Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019

Flexer, Arthur; Peeters, Geoffroy; Urbano, Julián; Volk, Anja

Publication date
2019

Document Version
Final published version

Citation (APA)
Flexer, A., Peeters, G., Urbano, J., & Volk, A. (Eds.) (2019). *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*. (Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019). International Society for Music Information Retrieval (ISMIR). https://archives.ismir.net/ismir2019/2019_Proceedings_ISMIR.pdf

20th International Society for Music Information Retrieval Conference

Across the bridge
4-8 November 2019
Delft, The Netherlands

ISMIR 2019

Proceedings of the 20th International Society for Music Information Retrieval Conference

November 4 - 8, 2019
Delft, The Netherlands

Edited by
Arthur Flexer, Geoffroy Peeters, Julián Urbano, Anja Volk

ISMIR 2019 is organized by Delft University of Technology and the International Society for Music Information Retrieval.

Website: http://ismir2019.ismir.net/

ISMIR 2019 Logo Designed by Saskia de Been
Proceedings Cover Designed by: Cynthia Liem

Edited by:
Arthur Flexer (Austrian Research Institute for Artificial Intelligence, Vienna, Austria)
Geoffroy Peeters (Telecom ParisTech, Paris, France)
Julián Urbano (Delft University of Technology, Delft, The Netherlands)
Anja Volk (Utrecht University, Utrecht, The Netherlands)

ISBN: 978-1-7327299-1-9

Title: Proceedings of the 20th International Society for Music Information Retrieval Conference

Organizers

Sponsors

Platinum Sponsors

Gold Sponsors

pandora®

DEEZER

chordify

Google

gracenote
A NIELSEN COMPANY

Silver Sponsors

ACRCloud

steinberg

Adobe

iZOTOPE

Kobalt

SONY

Bronze Sponsors

Peachnote

cochl.

antescofo

facebook

WiMIR Sponsors

WiMIR Benefactor

WiMIR Contributor

WiMIR Supporter

Conference Organizing Committee

General Chairs
Cynthia C. S. Liem, Delft University of Technology, Delft, The Netherlands
Emilia Gómez, Universitat Pompeu Fabra, Barcelona, Spain / Joint Research Centre, European Commision, Seville, Spain

Program Chairs
Arthur Flexer, Austrian Research Institute for Artificial Intelligence, Vienna, Austria
Geoffroy Peeters, Telecom ParisTech, Paris, France
Julián Urbano, Delft University of Technology, Delft, The Netherlands
Anja Volk, Utrecht University, Utrecht, The Netherlands

Tutorial Chairs
Eva Zangerle, University of Innsbruck, Innsbruck, Austria
Masataka Goto, AIST, Tsukuba, Japan

Proceedings and Archival Chairs
Alastair Porter, Universitat Pompeu Fabra, Barcelona, Spain
Eric J. Humphrey, Spotify, Boston MA, USA
Stefan Balke, Institute of Computational Perception, Johannes Kepler University Linz, Austria

Publicity Chairs
Amélie Anglade, Arcona.ai, Berlin, Germany
Tejaswinee Kelkar, University of Oslo, Oslo, Norway

Fringe programming Chair
Andrew Demetriou, Delft University of Technology, Delft, The Netherlands

Music
Martin Gasser, Austrian Research Institute for Artificial Intelligence, Vienna, Austria
Christof Weiß, International Audio Laboratories Erlangen, Erlangen, Germany

Late Breaking News and Demos Chairs
Jaeyoung Choi, Delft University of Technology, Delft, The Netherlands / International Computer Science Institute, Berkley, CA, USA
Mohamed Sordo, Pandora Media, Inc. Oakland, CA, USA

Unconference
Peter van Kranenburg, Meertens Institute, Amsterdam, The Netherlands

Satellite Events Chair
Jaehun Kim, Delft University of Technology, Delft, The Netherlands

Newcomer Initiatives Chairs
Iris Yuping Ren, Utrecht University, Utrecht, The Netherlands
Sandy Manolios, Delft University of Technology, Delft, The Netherlands

Sponsorship Chairs
Steve Tjoa, Google, Inc. CA, USA
Stefan Sullivan, Smule, Inc. San Francisco, CA, USA

Web Chair
Bart Vastenhouw, Delft University of Technology, Delft, The Netherlands

Finances & local arrangements
TU Delft Event Solutions, Delft University of Technology, Delft, The Netherlands
Saskia Peters, Delft University of Technology, Delft, The Netherlands

Advisor
Blair Kaneshiro, Stanford University, Stanford, CA, USA
Meinard Müller, International Audio Laboratories Erlangen, Erlangen, Germany
Frans Wiering, Utrecht University, Utrecht, The Netherlands

Best Paper Awards Panel
Tom Collins, University of York; MAIA, Inc.
Zhiyao Duan, University of Rochester
Olivier Lartillot, RITMO, University of Oslo
Florence Levé, Université de Picardie Jules Verne - Lab. MIS - Algomus
Oriol Nieto, Pandora
Alexander Schindler, Austrian Institute of Technology
Peter Van Kranenburg, Meertens Insitute; Utrecht University

Program Committee

Meta-reviewers

Andreas Arzt, Johannes Kepler University
David Bainbridge, Waikato University
Emmanouil Benetos, Queen Mary University of London
Rachel Bittner, Spotify
Sebastian Böck, Austrian Research Institute for Artificial Intelligence (OFAI)
Dmitry Bogdanov, Universitat Pompeu Fabra
John Ashley Burgoyne, University of Amsterdam
Ching-Hua Chuan, University of Miami
Tom Collins, University of York; MAIA, Inc.
Tim Crawford, Department of Computing, Goldsmiths, University of London, UK
Helene-Camille Crayencour, CNRS
Roger Dannenberg, School of Computer Science, Carnegie Mellon University
Matthew Davies, INESC TEC
Johanna Devaney, Brooklyn College
Christian Dittmar, Fraunhofer IIS
Simon Dixon, Queen Mary University of London
Stephen Downie, University of Illinois at Urbana-Champaign
Zhiyao Duan, University of Rochester
Andreas Ehmann, Pandora
Sebastian Ewert, Spotify
Gyorgy Fazekas, Queen Mary University of London
Ichiro Fujinaga, McGill University
Masataka Goto, National Institute of Advanced Industrial Science and Technology (AIST)
Fabien Gouyon, Pandora
Maarten Grachten, Machine Learning Consultant
Dorien Herremans, Singapore University of Technology and Design
Andre Holzapfel, KTH Royal Institute of Technology in Stockholm
Xiao Hu, The University of Hong Kong
Eric J. Humphrey, Spotify
Ozgur Izmirli, Connecticut College
Blair Kaneshiro, Stanford University
Katherine M. Kinnaird, Smith College
Peter Knees, TU Wien
Audrey Laplante, Université de Montréal
Olivier Lartillot, RITMO, University of Oslo
Jin Ha Lee, University of Washington
Alexander Lerch, Georgia Tech Center for Music Technology
Florence Levé, Université de Picardie Jules Verne - Lab. MIS - Algomus
Michael Mandel, Brooklyn College, CUNY
Brian McFee, New York University
Cory McKay, Marianopolis College
Matt McVicar, Apple
Meinard Müller, International Audio Laboratories Erlangen
Oriol Nieto, Pandora
Kevin R Page, University of Oxford
Johan Pauwels, Queen Mary University of London

Matthew Prockup, Pandora
Marcelo Queiroz, University of São Paulo
Preeti Rao, IIT Bombay
Christopher Raphael, Indiana University
Andreas Rauber, TU Wien
Justin Salamon, Adobe Research
Markus Schedl, Johannes Kepler University Linz
Alexander Schindler, Austrian Institute of Technology
Xavier Serra, Universitat Pompeu Fabra
Jordan B. L. Smith, National Institute of Advanced Industrial Science and Technology (AIST)
Mohamed Sordo, Pandora
Bob L. T. Sturm, KTH Royal Institute of Technology
Li Su, Academia Sinica
Douglas Turnbull, Ithaca College
George Tzanetakis, University of Victoria
Peter Van Kranenburg, Meertens Insitute; Utrecht University
Ye Wang, National University of Singapore
Geraint A. Wiggins, Vrije Universiteit Brussel
Kazuyoshi Yoshii, Kyoto University
Eva Zangerle, University of Innsbruck

Reviewers

Samer Abdallah
Jakob Abeßer
Stefanie Acevedo
Kat Agres
Islah Ali-Maclachlan
Anna Aljanaki
Nazareno Andrade
Tom Arjannikov
Claire Arthur
Stefan Balke
Isabel Barbancho
Dogac Basaran
Christine Bauer
Alejandro Bellogin
Brian Bemman
Oded Ben-Tal
Gilberto Bernardes
Louis Bigo
Ciril Bohak
Juanjo Bosch
Baris Bozkurt
Paul Brossier
Razvan Bunescu
Jorge Calvo-Zaragoza
Carlos Eduardo Cancino-Chacón
Estefania Cano
Julio Carabias
Rafael Caro Repetto
Benjamin Carterette
Pritish Chandna
Ning Chen
Tian Cheng
Srikanth Cherla
Kahyun Choi
Keunwoo Choi
Chia-Hao Chung
Nathaniel Condit-Schultz
Albin Correya
Emanuele Coviello
Helena Cuesta
Reinier de Valk
Andrew Demetriou
Junqi Deng
Chris Donahue
Jonathan Driedger
Hamid Eghbal-zadeh
Anders Elowsson
Jesse Engel
Philippe Esling
Jianyu Fan
Zhe-Cheng Fan
Xavier Favory
Andres Ferraro
Flavio Figueiredo
Frederic Font
José Fornari
Klaus Frieler
Magdalena Fuentes
Satoru Fukayama
Nick Gang
Martin Gasser
Roman Gebhardt
Mathieu Giraud
Aggelos Gkiokas
Rong Gong
Mark Gotham
Ryan Groves
David Grunberg
Carlos Guedes
Sankalp Gulati
Chitralekha Gupta
Siddharth Gururani
Yoonchang Han
Yun Hao
Peter Harrison
Florian Henkel
Jason Hockman

Yu-Hui Huang
Chris Hubbles
Karim M. Ibrahim
Jose M. Inesta
Charles Inskip
Katsutoshi Itoyama
Florent Jacquemard
Berit Janssen
Andreas Jansson
Tristan Jehan
Il-Young Jeong
Maximos Kaliakatsos-Papakostas
Andreas Katsiavalos
Tejaswinee Kelkar
Rainer Kelz
Jaehun Kim
Jong Wook Kim
Minje Kim
Rainer Kleinertz
Filip Korzeniowski
Katerina Kosta
Amanda E Krause
Lun-Wei Ku
Frank Kurth
Mathieu Lagrange
Robin Laney
Stefan Lattner
Kyogu Lee
Mark Levy
David Lewis
Bochen Li
Peter Li
Shengchen Li
Elad Liebman
Yuan-Pin Lin
Adam Liska
Meijun Liu
Yi-Wen Liu
Antoine Liutkus
Patricio López-Serrano
Simon Lui
Athanasios Lykartsis
Alexis D MacIntyre
Akira Maezawa
José Pedro Magalhães
Lucas S Maia
Leandro Balby Marinho
Matija Marolt
David Martins de Matos
Matthias Mauch
Matthew C McCallum
Andrew McLeod
David Meredith
Marius Miron
Nicola Montecchio
Hema A Murthy
Hidehisa Nagano
Eita Nakamura
Tomoyasu Nakano
Juhan Nam
Maria Navarro
Eric Nichols
Ken O'Hanlon
Mitsunori Ogihara
Sergio Oramas
Emilia Parada-Cabaleiro
Ashis Pati
Antonio Pertusa
Pedro D. Pestana
Aggelos Pikrakis
Pedro J. Ponce de León
Jordi Pons
Phillip Popp
Alastair Porter
Zafar Rafii
Antonio Ramires
Gang Ren
Iris Yuping Ren
David Rizo
Francisco Rodriguez Algarra
Gerard Roma
Robert Rowe
Patrick E. Savage
Flávio Luiz Schiavoni
Jan Schlüter
Rodrigo Schramm
Hendrik Schreiber
Jeffrey Scott
David Sears
Eleanor Selfridge Field
Sertan Şentürk
Zhengshan Shi
Siddharth Sigtia
George Sioros
Joren Six
Olga Slizovskaia
Lloyd A Smith
Carl Southall
Ajay Srinivasamurthy
Daniel Stoller
Fabian-Robert Stöter
Vinod Subramanian
Tiago Tavares
Florian Thalmann
Mi Tian
Marko Tkalcic
Petri Toiviainen
George Tourtellot
Christopher J Tralie
David Trevelyan
Timothy Tsai
Kosetsu Tsukuda
Amruta Vidwans
Gabriel Vigliensoni
Richard Vogl
Michael Vötter
Sanna Wager
Cheng-i Wang
Hsin-Min Wang
Xinxi Wang
David M. Weigl
Christof Weiss
Ron Weiss
Tillman Weyde
Daniel Wolff
Minz Won
Chih-Wei Wu
Anna Xambó
Karthik Yadati
Yujia Yan
Luwei Yang
Adrien Ycart
Asteris Zacharakis
Frank Zalkow
Yichi Zhang

Preface

With great pleasure, we welcome you to ISMIR 2019, the 20th International Society for Music Information Retrieval Conference. ISMIR is the world's leading research forum on processing, searching, organizing and accessing music-related data. For the 20th anniversary of the conference, we are honored to welcome you to the beautiful city of Delft, The Netherlands, where the conference will take place from November 4-8, 2019.

The tagline for this year's conference is 'Across the Bridge'. Our community reflects a diversity of scientific disciplines, seniority levels, professional affiliations, and cultural backgrounds. It always has been explicitly interested in fostering and stimulating this diversity, leading to better science and better music services. For this anniversary edition of ISMIR, through various actions and program elements, we have tried to explicitly encourage the community to take this a step further, and actively connect across the bridges between our backgrounds.

Scientific program

We are excited to present this year's program. A total of 308 abstracts were registered, of which 246 finally turned into complete and well-formatted papers to enter the review process. The PC insisted on a strict adherence to ISMIR's submission policy, e.g. not accepting any paper simultaneously under review at another venue. Special care was taken to assemble an experienced and interdisciplinary review panel reflecting our community's diversity of scientific disciplines and cultural backgrounds in line with this year's conference tagline 'Across the Bridge'. As in previous years, reviews were double-blind (i.e., both the authors and the reviewers were anonymous) with a two-tier review model involving a pool of 212 reviewers, plus 66 meta-reviewers. Each paper was assigned to 1 meta-reviewer and 3 reviewers. The meta-reviewers' assignments were based on topic preferences and bids on papers. The reviewers' assignments were based on these criteria and meta-reviewer suggestions. Handling at most 4 submissions, each meta-reviewer was asked to provide a full review themselves and to adopt an active role in the discussion phase with the other reviewers, ultimately providing a final summarizing meta-review. Of the 246 reviewed papers, 110 were accepted, resulting in an acceptance rate of 44.7%. Among these 110 papers, 66 have a student as the first author.

Authors could assign one or more subject areas to their paper. The following table indicates the distribution (in percent of the total number of papers) of subject areas. We indicate this distribution over the submitted (246) and the accepted works (110).

Subject Area	% of accepted	% of submitted
Domain knowledge: machine learning/artificial intelligence for music	40.0	38.6
MIR data and fundamentals: symbolic music processing	23.6	15.9
Domain knowledge: representations of music	20.9	16.3
MIR data and fundamentals: music signal processing	19.1	16.7
Applications: music retrieval systems	18.2	16.3
Music processing: pattern matching and detection	16.4	13.0
Music processing: automatic classification	16.4	18.3
Music processing: similarity metrics	16.4	13.8
Domain knowledge: computational music theory and musicology	16.4	12.6

Music processing: music transcription and annotation	13.6	12.2
Applications: music composition, performance and production	13.6	12.6
Evaluation and Methodology: MIR tasks, datasets and annotation protocols	13.6	11.4
Musical features and properties: harmony, chords and tonality	13.6	10.6
Musical features and properties: rhythm, beat, tempo	11.8	8.9
Musical features and properties: expression and performative aspects of music	10.9	8.5
Musical features and properties: melody and motives	10.0	8.9
Musical features and properties: structure, segmentation and form	10.0	9.8
User-centered MIR: human-computer interaction and interfaces	10.0	8.5
Applications: digital libraries and archives	9.1	5.3

To commemorate the 20 year anniversary of the ISMIR conference, authors were also invited to submit a contribution to a "20th anniversary papers" track. Four papers (one of these having seven undergraduate student joint first authors) were chosen from 12 submissions. This took place in a separate process involving only meta-reviewers. Every anniversary paper was reviewed and discussed by 4 meta-reviewers and 1 senior meta-reviewer.

This year we introduced several changes to the submission and review processes, such as: 1) the use of Microsoft CMT as submission system, which offers more flexibility, more features, and can be used without cost; 2) the submission form was re-designed to let authors upload supplementary material (e.g., code, audio samples) and provide the takeaway message from their contribution; 3) the review form was also re-designed to reduce problematic borderline decisions and focus on the re-usable insights of the papers; 4) the review process was 100% anonymous, that is, no author, reviewer or meta-reviewer knew the identity of other author, reviewer or meta-reviewer; and 5) the attribution of best-paper awards, moving from the choice of a single best contribution to a set of best contributions.

Based on meta-reviewers' recommendations for best paper awards, a panel of 7 experts reread all 8 recommended papers and delivered a ranking of the papers plus verbatim explanations of their reasoning. With this information at hand, the Scientific Program Chairs made the final choice to award 4 best papers. Next to this, based on rating feedback given by meta-reviewers, the choice was made to award 4 best reviewers.

The table shown below summarizes the ISMIR publication statistics over the history of the conference. A very special thanks goes to our Scientific Program Chairs for their extensive efforts to ensure a fair and efficient paper selection.

Following the successful presentation format that was first piloted at ISMIR 2018 in Paris, all regular papers both have been assigned a short plenary oral presentation, as well as a poster presentation. With anniversary papers being more focused on community reflection, these papers are presented in longer oral slots, with plenary Q&A.

To further emphasize equality of all regular papers—regardless of their subject area—the presentation order of papers has been randomized. In doing this, authors of thematically related work also have lower odds of having to present their posters in parallel.

To allow for longer exchanges about the scientific work presented at ISMIR, and more public exposure of ISMIR's output (given various public outreach efforts, as mentioned below), the choice also was made to have all posters exhibited during the full conference.

Year	Location	Oral	Poster	Total Papers	Total Pages	Total Authors	Unique Authors	Pages/ Paper	Authors/ Paper	Unique Authors/ Paper
2000	Plymouth	19	16	35	155	68	63	4.4	1.9	1.8
2001	Indiana	25	16	41	222	100	86	5.4	2.4	2.1
2002	Paris	35	22	57	300	129	117	5.3	2.3	2.1
2003	Baltimore	26	24	50	209	132	111	4.2	2.6	2.2
2004	Barcelona	61	44	105	582	252	214	5.5	2.4	2
2005	London	57	57	114	697	316	233	6.1	2.8	2
2006	Victoria	59	36	95	397	246	198	4.2	2.6	2.1
2007	Vienna	62	65	127	486	361	267	3.8	2.8	2.1
2008	Philadelphia	24	105	105	630	296	253	6	2.8	2.4
2009	Kobe	38	85	123	729	375	292	5.9	3	2.4
2010	Utrecht	24	86	110	656	314	263	6	2	2.4
2011	Miami	36	97	133	792	395	322	6	3	2.4
2012	Porto	36	65	101	606	324	264	6	3.2	2.6
2013	Curitiba	31	67	98	587	395	236	5.9	3	2.4
2014	Taipei	33	73	106	635	343	271	6	3.2	2.6
2015	Málaga	24	90	114	792	370	296	7	3.2	2.6
2016	New York	25	88	113	781	341	270	6.9	3.0	2.4
2017	Suzhou	24	73	97	716	324	248	7.4	3.3	2.6
2018	Paris	104		104	786	337	265	7.5	3.2	2.5
2019	**Delft**	**114**		**114**	**889**	**390**	**315**	**7.8**	**3.4**	**2.8**

WiMIR

Women in MIR (WiMIR) is a group of people in the MIR community dedicated to promoting the role of, and increasing opportunities for, women in the field. We meet to network, share information, and discuss in an informal setting the goal of building a community that supports women—and more broadly, diversity—in the field of MIR. WiMIR has held annual meetings at the ISMIR conference since 2012, garnering a high turnout of both female and male attendees. Since 2016, WiMIR has also organized a mentoring program connecting female students, postdocs, and early-stage researchers to more senior females and male allies in the field.

In contrast to past ISMIRs, for ISMIR 2019, the choice was made to not designate dedicated WiMIR chairs to the conference, and to explictly not restrict topics of interest to WiMIR to a WiMIR reception and session only. Instead, efforts were taken to make WiMIR's themes of interest naturally integrated, central themes to the conference, as part of the tasks of the General Chairs.

This year, the WiMIR workshop, that offers an accessible entry and networking opportunity to people interested in ISMIR themes, was scheduled before the ISMIR main conference. This was done to encourage for newly made connections at this workshop to further be extended throughout the main conference.

The WiMIR bingo, that normally would be held at a dedicated WiMIR reception or cocktail, has this year been included as an element in the official Welcome Reception of the conference, still bearing the WiMIR brand. Again, this has been done to emphasize that networking and inclusion are important to ISMIR as a whole.

Finally, the scope of the WiMIR grants was broadened and formulated more inclusively. For the first time, the WiMIR grant management was also taken up by a male ally.

Diversity & Inclusion

Diversity and inclusion are key themes to ISMIR 2019. Beyond thematic/disciplinary diversity (reflected in the scientific program) and gender diversity (reflected through WiMIR actions), we have actively tried to also stimulate other forms of diversity, and explicitly strive for inclusion.

With regard to other forms of diversity: as for diversity regarding types of positions and affiliations, through our Meetup with Industry, we have offered an explicit platform to our industry partners to present their recent work to the local community, beyond traditional conference presentation forms.

As for diversity regarding seniority levels, this year's ISMIR also has strived to actively include and encourage newcomers, and to make ISMIR accessible for the young generation, as well as those who may not have natural support and opportunity to afford attendance of ISMIR.

To this end, dedicated newcomer inclusion and accessibility strategies were followed. Through multiple student author grants, also partially supported by the ISMIR board, student authors got financial support to be able to join the conference.

Besides this grant, partially out of the anniversary donation of the ISMIR board to the ISMIR 2019 organizers, a new type of grant was established for ISMIR 2019: the Community Grant. This grant was meant to support several individuals who would like to attend ISMIR, but who are not in the capacity to actively participate as contributors to the conference yet (or anymore). Existing ISMIR members were encouraged to suggest and champion Community Grant applicants. Thanks to this grant, we both have managed getting several past ISMIR members to re-engage with the community, but also have supported multiple young people, e.g. from different disciplines, and from labs that do not have strong MIR presence or priorities at this moment, to join our community.

Also in terms of registration pricing, priority was given to make rates as accessible as possible to students. Following all these actions, we indeed have seen a substantial share of participants to ISMIR 2019 being newcomers to the conference (as much as 40% of registered full conference participants). To explicitly include these newcomers in the community, a dedicated Newcomer Chair focuses throughout the conference on making our newest community members indeed feel welcomed.

Sustainability

When collecting community input for ISMIR 2019, one request coming up was finding ways to reduce the ecological footprint of ISMIR. To this end, this year, proceedings are only distributed in digital form, and not on USB drives. We further chose not to produce a formal hardcopy program booklet, but instead offer hardcopies of paper listings on demand only, while including hyperlinks to papers and supplementary material in the digital file bearing these listings. To further increase the attractiveness of a digital program, and make it more interactive, we have chosen to make use of Introwise, a digital event platform and networking tool, suggested to us by ISMIR community member Anna Aljanaki. We are curious to see in the conference whether this will indeed allow for a conference experience that is at least as good as former conference experiences have been.

In terms of digital sustainability, we further have worked hard to comply to the ISMIR society standards on digital archivability, now also including the LBD submissions as non-proceedings, yet ISMIR-archived contributions.

Public visibility

ISMIR 2019 takes place at a location that does not have a large, long-time established MIR lab; at the same time, Delft does have an engaged community of students and citizens interested in learning about recent scientific and technological advances in the field. Taking advantage of the Aula Conference Centre having sufficient physical capacity, while the town of Delft is relatively small, we have actively invested in furthering ISMIR's open culture, by sharing as much of ISMIR as possible with the public.

Therefore, ISMIR's evening keynotes, the Youth Music Challenge, as well as the Meetup with Industry have been made free-of-charge ISMIR elements, explicitly accessible to the general public.

Keynote speakers

For ISMIR 2019, we are very honored to host four esteemed keynote speakers:

- Henkjan Honing, Professor in Music Cognition, Faculty of Humanities and the Faculty of Science University of Amsterdam: "What makes us musical animals". Henkjan has been a major role model in performing interdisciplinary scientific research on music, and disseminating science to the general public. We are looking forward to an inspiring talk on musicality: a topic not central to the MIR field, but personally recognizable for many of us.
- Georgina Born, Professor of Music and Anthropology, Professorial Fellow, Mansfield College, University of Oxford: "MIR redux: Knowledge and real-world challenges, and new interdisciplinary futures". Linking to our 'Across the Bridge' tagline, we are grateful to have Georgina shedding light on our field, from disciplinary viewpoints that we would not naturally encounter at ISMIR at large.
- Jeremy Pickens, Principal Data Scientist, OpenText: "Music as Investigation". Jeremy was an active member of our community in the early years of ISMIR; while he now formally works in other domains, he has kept on following what happened in ISMIR, and we are delighted to have him back at this anniversary edition.
- Henriette Cramer, Principal Research Scientist & PM, Spotify: "Music and algorithmic responsibilities in practice". Henriette is the expert on fair and responsible recommendation in music. With this being a topic that has rapidly emerged as a critical focus point for ISMIR, we look very much forward to learning from her.

Three of the keynotes (Henkjan Honing, Georgina Born, Jeremy Pickens) have been scheduled in early evening slots, such that a broader audience can be reached. The fourth keynote (Henriette Cramer) is scheduled as part of the WiMIR session, to emphasize the relation between the keynote's theme and key interests of WiMIR.

Tutorials

Our tutorial chairs have selected six tutorials, based on relevance (i.e. being of interest to the whole community or only a segment thereof); suitability to generally knowledgeable students in MIR rather than specialists; quality; comprehensiveness / coverage; originality of topics with respect to previous ISMIR tutorials; novelty of authors with respect to previous ISMIR tutorials; expertise of authors; foster complementarity of tutorials; and foster open source and free software tools/data and reproducible research.

This has led to three tutorials in the morning, and three tutorials in the afternoon of Monday, November 4:

Morning

- [T1] Fundamentals of Music Processing: An Introduction using Python and Jupyter Notebooks by Meinard Müller and Frank Zalkow;
- [T2] Generating Music with GANs: An Overview and Case Studies by Hao-Wen Dong and Yi-Hsuan;
- [T3] Audiovisual Music Processing by Zhiyao Duan, Slim Essid, Bochen Li and Sanjeel Parekh.

Afternoon

- [T4] Computational Modeling of Musical Expression: Perspectives, Datasets, Analysis and Generation by Carlos Cancino-Chaćon, Katerina Kosta and Maarten Grachten;
- [T5] Waveform-based music processing with deep learning by Sander Dieleman, Jordi Pons and Jongpil Lee;
- [T6] Fairness, Accountability and Transparency in Music Information Research (FAT-MIR) by Emilia Gómez, Andre Holzapfel, Marius Miron and Bob L. Sturm.

Unconference

As in previous years, there will be an Unconference session during ISMIR 2019. This session will offer a platform to discuss MIR-related topics as proposed by the participants in a bottom-up fashion. With plenty of meeting rooms and a central hall, the conference venue provides an excellent infrastructure for the Unconference. We will start Friday 8th of November, 15:30 with a short plenary meeting in the Frans van Hasseltzaal, in which we will establish the topics. In principle, any topic that generates sufficient interest among the participants will be facilitated, limited by the amount of groups and time-slots available. To speed-up the topic selection process, the Introwise tool will be used to prepare topics and do the voting.

Late Breaking/Demo Session

The Late Breaking/Demo (LBD) Session features prototype systems, initial concepts, and early results which have not yet fully matured, but are of interest to the Music-IR community. Following the convention of past years, a light review was performed on submissions by our LBD chairs, and a maximum of 50 submissions has been admitted for presentation at the conference.

To further emphasize that LBD contributions are not part of the proceedings of ISMIR, but rather should be seen as non-refereed works, a new paper template has been used that makes this status more explicit. To allow for those in need of visas to apply in time, LBD acceptance was performed in two stages, with the first stage explicitly allowing for feasible timelines towards visa applications.

Youth Music Challenge

Many people, regardless of their backgrounds or interests, love music and consume it daily. These days, this consumption will mostly be focused on digital music, which users discover and access through online services. As such, many people in the world implicitly engage with MIR technology, and our research field has considerable potential for public outreach. As discussed during the WiMIR session of ISMIR 2018, this could be an opportunity to try engaging a more diverse young audience for STE(A)M topics.

On the occasion of ISMIR 2019, we have partnered with Roem, a Delft-based Campaign Agency for the Greater Good, to organize a Youth Music Challenge. In the months between ISMIR 2018 and ISMIR 2019, we visited various (platforms for) STEM teachers in Dutch secondary schools for advice, and developed workshops and a challenge for youngsters aged 12-18, asking for them to propose, prototype and present ideas for 'the music service of the future'.

The first edition of the Youth Music Challenge was run at the Christelijk Lyceum Delft, a 'Technasium' school which explicitly includes STEM design challenges in the curriculum. Five groups of middle and high school students (aged 12-13 and 15-16) have participated in the challenge, and will pitch their ideas on the night of November 5, 2019, at OPEN, the public library and music school of Delft. These outcomes will subsequently become part of a temporary public exhibit at OPEN, showcasing our youth's ideas to both the ISMIR community and the general public.

The students will receive feedback by a jury consisting of:

- Cynthia Liem, general co-chair of ISMIR 2019, assistant professor at Delft University of Technology and initiator of the Youth Music Challenge;
- Romain Hennequin, lead scientist at Deezer;
- Casper Karreman, senior developer at Muziekweb;
- Peter Sobot, staff software engineer at Spotify.

Our special thanks go to:

- Martijn Kirsten from Roem, for the production of the Youth Music Challenge;
- Alexander Ettema from Studio Alex, for creative and media support to the challenge;
- John Schmitz, dean of the Faculty of Electrical Engineering, Mathematics and Computer Science of Delft University of Technology, for having agreed to financially support the production of the Youth Music Challenge, as an effort to increase future diversity in our professional field;
- students and teachers from Christelijk Lyceum Delft, for having participated in the challenge.

The intention is to keep running the Youth Music Challenge in coming years, and expand its scope to a truly national one.

Meetup with Industry

Following an initiative first started at ISMIR 2018, after the main conference, a Meetup with Industry will take place on Friday, November 8 in the late afternoon and evening. With ISMIR being the world's leading R&D forum for music tech since 2000, and many colleagues having taken industry positions, the Meetup with Industry offers a networking platform and exposure opportunity in which participants can:

- Discover and shape the future of music technology;
- Connect with and recruit top talent from the MIR community and local universities;
- Meet world-class experts and explore the current state-of-the-art in music, creative and data science tech applications;
- Connect with and learn from leading music, creative and data science tech companies;
- Discover startups leveraging latest technologies in a variety of music, creative and data science applications;
- Exchange with other players and startups in the music, creative and data science industry;
- Showcase and market latest products and services.

We have strived to make the Meetup as accessible as possible, and to encourage the participation of interested parties who may not be in the ISMIR community, with special focus on students. Therefore, the choice was made to make access to the Meetup free of charge, including first drinks and bites, to explicitly campaign for it towards students at Delft University of Technology, and to organize it close to the campus, at the very end of the students' exam week.

Exhibitors present at the Meetup include Spotify, Tencent Music Entertainment, Pandora, Chordify, Google, Gracenote, Peachnote, AI Music, Universal Music Group and Dolby.

Social Events

ISMIR 2019 has two official social events.

Welcome reception

The ISMIR 2019 Welcome Reception will take place on Monday, November 4 at the Bierfabriek in Delft. In a building that was formerly the local discotheque of Delft, the Bierfabriek now hosts a large informal pub and restaurant, serving various types of home-brewed beer and comfort food.

Conference Banquet & Pandora Jam Session

The ISMIR 2019 Conference Banquet & Pandora Jam Session will take place on Thursday, November 7 at the Lijm & Cultuur complex in Delft. Formerly a glue factory, Lijm & Cultuur now is a multi-functional creative lab, mostly used as a cultural and festival location. Following dinner, the community will transition into its now-traditional jam session. We are grateful to our Gold Sponsor Pandora for having financially supported this jam, and look forward to seeing many ISMIR attendees playing.

Beyond these official social events, the city of Delft offers multiple opportunities for local social experiences in smaller groups. Several local cafes known for their live music agreed to allow ISMIR attendees to play during non-booked late nights, and we look forward to hearing them.

Music

To commemorate ISMIR's 20th anniversary, the Call for Music for ISMIR 2019 was targeted at creating a joint musical-technical performance, taking up the spirit of ISMIR conferences, jam sessions, late-breaking/demos, and hackathons.

To include as many people as possible, a video performance entitled "Variations on ISMIR" has been compiled with Mozart's famous Variations on the French song "Ah, vous dirai-je maman" (K. 265) as basis. The variation theme is also well-known as a children's or Christmas song ("Twinkle, twinkle little star"...). These variations have had significance to the first ISMIR in 2000, and a score excerpt of this piece has served as the first ISMIR logo for many years.

The ISMIR community was encouraged to contribute short variations on this theme (maximally 60 seconds), showing both ISMIR people and MIR technology in action. In response to the Call for Music, multiple contributions were sent in from across the world. The final result will be shown in a plenary music intermezzo, and also will be made available online.

Satellite Events

Surrounding the ISMIR 2019 conference, participants have the opportunity to attend several satellite events:

- **2nd International Workshop on Reading Music Systems (WoRMS)**, held on November 2 as a half-day workshop at the Aula Conference Centre of Delft University of Technology;
- **1st Workshop on Designing Human-Centric MIR Systems**, held on November 2 as a half-day workshop at the Aula Conference Centre of Delft University of Technology;
- **2nd Women in MIR Workshop**, held on November 3 as a full-day workshop at the Aula Conference Centre of Delft University of Technology;
- **6th International Conference on Digital Libraries for Musicology (DLfM)**, held on November 9 as a full-day conference at the National Library of The Netherlands, The Hague.

Host City

Delft is a compact, historic town between Rotterdam and The Hague in the province of South-Holland, The Netherlands. It forms part of the 'Randstad', the urban agglomeration in the western part of the Netherlands, and is one of the nation's main educational and research centers.

Delft is more than 900 years old. The city owes its name to the word 'delven' (digging). Around 1100 AD, a local river was widened by hand to enable better drainage of the surrounding farmland, leading to the oldest (and still existing) canal called the Oude Delft (Old Delft). In 1246, Delft received its city franchise from the Dutch Earl Willem II. Delft flourished and new neighborhoods were added. As early as 1355, the city reached the size it would remain until the 19th century.

During the country's war of independence against Spain in the 16th and 17th centuries, Delft played an important role as the residence of William of Orange, known as the Father of the Nation who initiated the establishment of the state of The Netherlands. Upon his assassination at his Prinsenhof residence, he was buried in the New Church of Delft. His descendants formed the House of Orange-Nassau, which ultimately would become the Dutch Royal Family. The Royal Family still holds a special connection to Delft, and the New Church has now become the Royal Crypt in which former kings and queens are buried.

Many world-renowned painters like Johannes Vermeer, Jan Steen and Karel Fabritius lived and worked in Delft. Delft was also the hometown of scientist Antoni van Leeuwenhoek, inventor of the microscope. Furthermore, the city became famous for its Delft Blue pottery.

The establishment of the Royal Academy for the Training of Civil Engineers in 1842 (which later would become Delft University of Technology) was a strong stimulus to the revival of the industry (and thus the importance of Delft) in the 19th century. New neighborhoods were built, and university buildings and faculties were relocated from the historic center of the city to a new quarter dedicated to the university.

Despite wars and rapid population growth, the historic center of Delft has remained almost completely intact. Today, Delft is a beloved tourist location, an important high-tech hub, and a pleasant home to almost 100,000 inhabitants, many of whom are affiliated to the university.

Acknowledgements

We are very proud to present to you the proceedings of ISMIR 2019. The conference program was made possible thanks to the hard work of many people, including the members of the organizing committee, and the many reviewers and meta-reviewers from the program committee.

We would like to thank our sponsors, whose contributions have been indispensable for us to be able to support the multiple grants of ISMIR 2019:

Platinum sponsors

- Spotify
- Tencent Music Entertainment

Gold sponsors

- Pandora
- Deezer
- Chordify
- Google
- Gracenote

Silver sponsors

- ACRCloud
- Steinberg
- Adobe
- iZotope
- Kobalt Music Group
- Sony

Bronze sponsors

- Peachnote
- Cochlear.ai
- Antescofo
- Facebook

Besides, we would like to thank the sponsors that explicitly chose to sponsor WiMIR, its grants and its initiatives:

WiMIR Benefactor

- Spotify

WiMIR Contributor

- Smule

WiMIR Supporter

- iZotope
- Peachnote
- Native Instruments
- Steinberg
- Kobalt Music Group

We thank the ISMIR board, for having both financially and morally supported the Student Author and Community Grants of ISMIR 2019. We also are very grateful for the board

having given us both trust and freedom to pro-actively innovate many elements in ISMIR 2019, striving to make ISMIR's conferences even better than they already are.

We thank both the leadership and support staff of Delft University of Technology for having enabled this ISMIR. Implicitly, when counting in-kind support as contributions too, TU Delft has by far been the largest sponsor to this conference. We are extremely grateful we could make use of many in-house facilities, thus keeping this ISMIR financially manageable, despite us having taken greater financial risks when giving out grants and lowering our student rates.

We especially want to thank:

- Ginny Ruiter and Natascha Voskuijl from TU Delft Event Solutions, for having supported the logistics of ISMIR 2019;
- The event managers & tech supporters of the Aula Conference Centre, for making our conference experience comfortable, smooth and stable;
- TU Delft Media Solutions, in particular Saskia de Been and Debby van Vondelen, for having made the visual designs for ISMIR 2019;
- Saskia Peters, Management Assistant at the Multimedia Computing Group of Delft University of Technology, for having supported a considerable share of contractual and financial administrative work surrounding ISMIR 2019;
- Andrew Demetriou, Jaehun Kim and Sandy Manolios, PhD students at the Multimedia Computing Group of Delft University of Technology, for having taken important middle-management tasks throughout ISMIR;
- Francesca Lucas, for further practical and last-minute assistance on ISMIR matters;
- The many student volunteers, who have helped us in making ISMIR 2019 run smoothly.

We also thank you, the community: authors, reviewers, researchers, and participants of this conference. Thanks so much for making and being ISMIR. Let's make this a productive, memorable, and inspiring conference together. Let's consciously foster the welcoming and open culture of ISMIR, and proudly share it with those around us, within and beyond the ISMIR community. And here's to another 20 years of ISMIR, and beyond.

Arthur Flexer
Geoffroy Peeters
Julián Urbano
Anja Volk
Scientific Program Chairs

Cynthia C. S. Liem
Emilia Gómez
General Chairs

Contents

Keynote Talks

Keynote Talk 1

What makes us musical animals

Henkjan Honing

Professor in Music Cognition
Faculty of Humanities and the Faculty of Science
University of Amsterdam

Abstract

We are all born with a predisposition for music, a predisposition that develops spontaneously and is refined by listening to music. Nearly everyone possesses the musical skills essential to experiencing and appreciating music. Think of "relative pitch," recognizing a melody separately from the exact pitch or tempo at which it is sung, and "beat perception," hearing regularity in a varying rhythm. Research shows that all humans possess the trait of musicality. We are a musical species—but are we the only musical species? Can there be musical machines? In his presentation, Henkjan Honing embarks upon the quest to discover the cognitive and biological mechanisms that underpin musicality.

Biography

Henkjan Honing is a professor of Music Cognition at both the Faculty of Humanities and the Faculty of Science of the University of Amsterdam (UvA). He studies what musicality is or can be and to what extent human beings share musicality with other animals. His aim is to define the cognitive and biological mechanisms that underpin musicality. In addition to a research agenda (The Origins of Musicality, 2018, MIT Press), Honing has published several books for the general public, including the English-language publications Musical Cognition and The Evolving Animal Orchestra. Honing's books and lectures are popular with a broad audience and are appreciated both inside and outside the scientific world.

Keynote Talk 2

MIR redux: Knowledge and real-world challenges, and new interdisciplinary futures

Georgina Born
Professor of Music and Anthropology
Professorial Fellow, Mansfield College
University of Oxford

Abstract

How can MIR refresh itself and its endeavors, scholarly and real world? I speak as an outsider, and it is foolhardy to advise scientist colleagues whose methodologies one would be hard pressed to follow! Nonetheless, my question points in two directions: first, to two areas of auto-critique that have emerged within the MIR community – to do with the status of the knowledge produced, and ethical and social concerns. One theme that unites them is interdisciplinarity: how MIR would gain from closer dialogues with musicology, ethnomusicology, music sociology, and science and technology studies in music. Second, the 'refresh' might address MIR's pursuit of scientific research oriented to technological innovation, itself invariably tied to the drive for economic growth. The burgeoning criticisms of the FAANG corporations and attendant concerns about sustainable economies remind us of the urgent need for other values to guide science and engineering. We might ask: what would computational genre recognition or music recommendation look like if, under public-cultural or non-profit imperatives, the incentives driving them aimed to optimise imaginative and cultural self- and/or group development, adhering not to a logic of 'similarity' but diversity, or explored the socio-musical potentials of music discovery, linked to goals of human flourishing (Nussbaum 2003, Hesmondhalgh 2013)? The time is ripe for intensive and sustained interdisciplinary engagements in ways previously unseen. My keynote ends by inviting action: a think tank to take this forward.

Biography

Georgina Born OBE FBA is Professor of Music and Anthropology at the University of Oxford, was the bass player with Henry Cow in the late 70s, played improvised cello in the 80s, wrote an ethnography of IRCAM in the 90s and an ethnography of the BBC in the 2000s, and is a leading interdisciplinary scholar writing on the mediation of music, especially its social forms. She ran an ERC-funded research group (2010-15) studying ethnographically how digitization and the internet have affected music worldwide.

Keynote Talk 3

Music as Investigation

Jeremy Pickens
Principal Data Scientist
OpenText

Abstract

The music information retrieval landscape has changed dramatically in the past two decades since ISMIR's inception. Music and user data exist at web scale and systems and algorithms have evolved to take advantage of it. Yet there remain classes of problems and information needs for which scale data will never be available. The early ISMIR community, perhaps if only out of necessity, responded to such challenges by adopting various mindsets: exploratory, investigatory, niche. The benefits of these mindsets extend far beyond music into other individual-scale, task-oriented domains such as law, where popularity and prevalence do not provide easily-distillable answers. Do the benefits run both ways?

Biography

Jeremy Pickens is a Principal Data Scientist at OpenText (né Catalyst Repository Systems), a leader in enterprise software and legal technology. His research in information retrieval has spanned a number of domains from music to images and video to various legal applications. The common thread among these disparate areas has been an emphasis on recall-oriented, comprehensive, and holistic views of relevance. Jeremy is a pioneer in the field of collaborative exploratory search, a form of information seeking in which a group of people who share a complex information need actively collaborate to achieve it. His ongoing research focuses on methods for continuous learning. Jeremy holds a PhD in Computer Science from the University of Massachusetts, Amherst, Center for Intelligent Information Retrieval. He conducted his post-doctoral work at King's College, London. Before joining Catalyst and OpenText, he spent five years as a research scientist at FX Palo Alto Lab, Inc. He was also a member of many of the early (2000-2005) ISMIR organizing committees.

WiMIR Keynote

Music and algorithmic responsibilities in practice

Henriette Cramer
Principal Research Scientist & PM
Spotify

Abstract

Music is deeply personal, and influences our moods and motivation. Music creates communities, shapes subcultures, and powers an enormous industry. What music is available, what can be found, what is recommended, matters. Who gets to learn, to play, to record, matters.
And, crucially, how the music recommendation and retrieval community defines success determines who gets amplified.

We're at a moment in tech where, alongside its successes, machine learning's failures and biases have gained much attention. There is an overwhelming number of calls-to-action but still relatively few standard practices for industry practitioners. This means we have a responsibility, especially as an ISMIR community. But what does that mean, practically?

This talk will outline challenges encountered in practice and at scale, specific to music streaming. We'll briefly travel through time, and take you from early 1900's magic lantern slides' music promotion to the current zeitgeist where new guidelines for algorithmic accountability are clamoring for attention themselves. We'll discuss how new UIs (like voice) can make certain creators inaccessible, female creators' representation in streaming, and optimizing for more than just engagement. We'll share technical and organizational lessons learned, pitfalls, and tensions in assessing decisions' potential impact.

Biography

Henriette Cramer is a principal researcher at Spotify Research, and product manages Spotify's Algorithmic Responsibility effort. She is particularly interested in the impact that teams' design, data and organizational decisions have on algorithmic outcomes. Prior, she set up Spotify's 'human side of Machine Learning' Hai lab, and led data research for Spotify's voice platform. She has worked on recommendations, ad quality, and conversational interactions at Yahoo, and on location-based data, perceptions of place, and human-robot interaction at the Swedish Institute of Computer Science. She holds a PhD from the University of Amsterdam focused on people's responses to autonomous systems. More at: `http://henriettecramer.com`

Tutorials

Tutorial 1

Fundamentals of Music Processing: An Introduction using Python and Jupyter Notebooks

Meinard Müller and Frank Zalkow

Abstract

This tutorial will give an easy-to-understand introduction to music processing with a particular focus on audio-related analysis and retrieval tasks. In particular, the tutorial is aimed at non-experts and researchers who are new to the field. Based on well-established topics in Music Information Retrieval (MIR) as motivating application scenarios, we present fundamental techniques and algorithms that apply to a wide range of analysis and retrieval problems. We intend to explain the main ideas and techniques in an intuitive fashion using various figures and sound examples. Besides the theory, we also show how these techniques can be implemented going through specific Python code examples. All material, including the introduction of MIR scenarios, illustrations, sound examples, technical concepts, mathematical details, and code examples, are integrated into a comprehensive framework based on Jupyter notebooks. The notebooks are organized along with the eight chapters of the textbook on Fundamentals of Music Processing (FMP) (Springer 2015, `http://www.music-processing.de`). Another important goal of this tutorial is to show how the notebooks can be used to generate educational material for lectures and presentations. The notebooks (as well as HTML exports and multimedia examples) can be accessed via `https://www.audiolabs-erlangen.de/FMP`.

Reference

Meinard Müller
Fundamentals of Music Processing — Audio, Analysis, Algorithms, Applications
Springer Verlag, ISBN: 978-3-319-21944-8, 2015.

Meinard Müller studied mathematics (Diplom) and computer science (Ph.D.) at the University of Bonn, Germany. In 2002/2003, he conducted postdoctoral research in combinatorics at the Mathematical Department of Keio University, Japan. In 2007, he finished his Habilitation at Bonn University in the field of multimedia retrieval. From 2007 to 2012, he was a member of the Saarland University and the Max-Planck Institut für Informatik. Since September 2012, Meinard Müller holds a professorship for Semantic Audio Processing at the International Audio Laboratories Erlangen, which is a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Fraunhofer-Institut für Integrierte Schaltungen IIS. His recent research interests include music processing, music information retrieval, audio signal processing, and motion processing. Meinard Müller has been a member of the IEEE Audio and Acoustic Signal Processing Technical Committee from 2010 to 2015 and is a member of the Board of Directors of the International Society for Music Information Retrieval (ISMIR) since 2009. He wrote a monograph titled "Information Retrieval for Music and Motion" (Springer, 2007) as well as a textbook titled "Fundamentals of Music Processing" (Springer, 2015, `http://www.music-processing.de`).

Frank Zalkow studied Music Informatics and Musicology (Bachelor) and Music Informatics (Master) at the University of Music Karlsruhe, Germany. Since 2016, he has been working towards his Ph.D. degree in the Semantic Audio Processing Group headed by Meinard Müller at the International Audio Laboratories Erlangen. Previously, he worked for the Max-Reger-Institute Karlsruhe (2008–15) as well as Institute for Musicology at Saarland University (2015–16). His research interests include music retrieval, machine learning, as well as cross-connections between musicology and music information retrieval.

Tutorial 2

Generating Music with GANs: An Overview and Case Studies

Hao-Wen Dong and Yi-Hsuan Yang

Abstract

This tutorial aims to provide an overview of generative adversarial networks (GANs) and their use in generating music. The format of the tutorial will include lectures, demonstration of sample systems and technical results with illustrative musical examples.

- We will start by discussing the scope of music generation and introduce various tasks that can broadly be regarded as music generation. For each task, we will then discuss its challenges, commonly used approaches and some notable systems proposed in the literature.
- In the second part, we will explain the machine learning fundamentals for GANs. We will also present some interesting applications of GANs in other fields to showcase their potentials.
- The following section will contain the case studies of four different tasks---symbolic melody generation, symbolic arrangement generation, symbolic musical style transfer and musical audio generation. In each part, we will first provide an overview of the task and then introduce several models proposed in the literature as examples.
- We will conclude the tutorial by discussing the current limitations of GAN-based models and suggesting some possible future research directions.

In addition to lectures, we will go through some demo projects using Google Colab. These demo projects are designed to provide participants with hands-on experience and deeper understanding of the training of GANs. We will also cover topics such as data representation, processing, I/O, visualization and evaluation.

The tutorial is targeted to students and newcomers who are interested in or working on music generation research, and also machine learning specialists who want to see how GANs can be applied to music generation.

Tutorial website: `https://salu133445.github.io/ismir2019tutorial/`

Hao-Wen Dong is currently a research internship in the Research and Development Division at Yamaha Corporation. He will be starting a Ph.D. this fall in Electrical and Computer Engineering at University of California, San Diego. Previously, he was a research assistant under the supervision of Dr. Yi-Hsuan Yang in the Music and AI Lab at Academia Sinica. He received his bachelor's degree in Electrical Engineering at National Taiwan University. His research interests lie at the intersection of machine learning and music.

Yi-Hsuan Yang is an Associate Research Fellow with Academia Sinica, where he leads a research lab called the Music and AI Lab. He received his Ph.D. degree in communication engineering from National Taiwan University in 2010. He is also a Joint-Appointment Associate Professor with the National Cheng Kung University. His research interests include

music information retrieval, affective computing, and machine learning. Dr. Yang was a recipient of the 2011 IEEE Signal Processing Society Young Author Best Paper Award, the 2012 ACM Multimedia Grand Challenge First Prize, and the 2015 Best Conference Paper Award of the IEEE Multimedia Communications Technical Committee. In 2014, he served as a Technical Program Chair of the International Society for Music Information Retrieval Conference (ISMIR). He gave a tutorial on “Music Affect Recognition: The State-of-the-art and Lessons Learned” in ISMIR 2012. He was an Associate Editor for the IEEE Transactions on Affective Computing and the IEEE Transactions on Multimedia in 2016-2019. He is currently on a sabbatical leave to work with a privately funded research organization in Taipei called the Taiwan AI Labs.

Tutorial 3

Audiovisual Music Processing

Zhiyao Duan, Slim Essid, Bochen Li, and Sanjeel Parekh

Abstract

Music is a multimodal art form. While sound plays a key role, other modalities, especially visual, are also critical to enhancing the musical experience. Recently, the MIR field has witnessed a rapid growth of interest in audiovisual processing of music.

This tutorial is intended to introduce this emerging research direction to the broader MIR community. It extends a recently published overview article on audiovisual analysis of music performances [1] into general audiovisual music processing. Specifically, it provides a comprehensive overview of state-of-the-art research in different aspects of audiovisual music processing, including music performance analysis, content-based retrieval, and music creation. It summarizes datasets, tools and other resources in this field, and articulates challenges and opportunities for future research. An interesting aspect of this tutorial is that it contains two hands-on case studies (30 min each) for the audience to personally experience audiovisual research. Instructions of software environments and starter code will be provided prior to the tutorial for preparation.

To our best knowledge, this is the very first tutorial on audiovisual processing at ISMIR. This tutorial is designed for students and researchers who have general knowledge of music information retrieval and who are interested in learning the state of the art and gaining hands-on experience of audiovisual music processing research. The comprehensive overview and categorization of different aspects of this field will help the audience gain a global view of the research problems, methods, tools, challenges, and opportunities. The hands-on case studies will provide the audience a first-hand experience of the research, helping them quickly arrive at the research frontier. We especially look forward to ideas and inspirations that the MIR community has to offer through this interactive and hands-on tutorial.

[1] Zhiyao Duan*, Slim Essid*, Cynthia C. S. Liem*, Gaël Richard*, and Gaurav Sharma*, “Audiovisual analysis of music performances: overview of an emerging field,” IEEE Signal Processing Magazine, vol. 36, no. 1, pp. 63-73, 2019. (* authors in alphabetical order)

Zhiyao Duan is an assistant professor in Electrical and Computer Engineering, Computer Science and Data Science at the University of Rochester. He received his B.S. in Automation and M.S. in Control Science and Engineering from Tsinghua University, China, in 2004 and 2008, respectively, and received his Ph.D. in Computer Science from Northwestern University in 2013. His research interest is in the broad area of computer audition, i.e., designing computational systems that are capable of understanding sounds, including music, speech, and environmental sounds. He is also interested in the connections between computer audition and computer vision, natural language processing, and augmented and virtual reality. He co-presented a tutorial on Automatic Music Transcription at ISMIR 2015. He received a best paper award at the 2017 Sound and Music Computing (SMC) conference, a best paper

nomination at the 2017 International Society for Music Information Retrieval (ISMIR) conference, and a CAREER award from the National Science Foundation.

Slim Essid received his state engineering degree from the École Nationale d'Ingénieurs de Tunis, Tunisia, in 2001, his M.Sc. (D.E.A.) degree in digital communication systems from the École Nationale Supérieure des Télécommunications, Paris, France, in 2002, his Ph.D. degree from the Université Pierre et Marie Curie (UPMC), Paris, France, in 2005, and his Habilitation à Diriger des Recherches degree from UPMC in 2015. He is a professor in Telecom ParisTech's Department of Images, Data, and Signals and the head of the Audio Data Analysis and Signal Processing team. His research interests are machine learning for audio and multimodal data analysis. He has been involved in various collaborative French and European research projects, among them Quaero, Networks of Excellence FP6-Kspace, FP7-3DLife, FP7-REVERIE, and FP-7 LASIE. He has published over 100 peer-reviewed conference and journal papers, with more than 100 distinct coauthors. On a regular basis, he serves as a reviewer for various machine-learning, signal processing, audio, and multimedia conferences and journals, e.g., a number of IEEE transactions, and as an expert for research funding agencies.

Bochen Li received his B.S. from University of Science and Technology of China in 2014. He is currently pursuing a Ph.D. degree in the Department of Electrical and Computer Engineering at the University of Rochester in the USA, under the supervision of Professor Zhiyao Duan. His research interests lie primarily in the inter-disciplinary area of audio signal processing, machine learning, and computer vision towards multimodal analysis of music performances, such as video-informed multipitch estimation and streaming, source separation and association, and expressive performance modeling and generation.

Sanjeel Parekh received B. Tech (hons.) degree in Electronics and Communication engineering from LNM Institute of Information Technology, India in 2014 and M.S. in Sound and Music Computing from Universitat Pompeu Fabra (UPF), Spain in 2015. His Ph.D. thesis titled 'Learning representations for robust audio-visual scene analysis' was completed in collaboration with Technicolor R&D and Telecom ParisTech, France between 2016-19. His research focusses on developing and applying machine learning techniques to problems in audio and visual domains. Currently, he is with LTCI lab at Telecom ParisTech, France.

Tutorial 4

Computational Modeling of Musical Expression: Perspectives, Datasets, Analysis and Generation

Carlos Cancino-Chacón, Katerina Kosta, and Maarten Grachten

Abstract

The aim of this tutorial is to introduce the theory and practice of music performance research to a broad MIR audience. A music performance—an acoustic or audio-visual rendering of it—provides a much richer musical experience than the symbolic or notated representation of the performed music. This richness is arguably an important aspect of our engagement with music and is shaped by the musician's interpretation of the intentions of the music, as conveyed through their performance. The means of expressing these intentions vary from one instrument to another, and can include tempo, timing, dynamics, articulation, timbre, and intonation, among others.

In this tutorial we will give a brief overview of the music performance literature and highlight how expressive dimensions affect the perception and the creation of music. Furthermore we will showcase some state-of-the-art computational methods for both analysis and synthesis of expressive piano performances. We include a hands-on part in which we share easily operated and adaptable code written in Python using Jupyter iPython notebook for demonstrating how to get started with computational analysis and synthesis of musical expression.

This work is partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 670035 (project "Con Espressione").

Carlos Cancino-Chacón is a postdoctoral researcher at the Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria. His research focuses on studying expressive music performance, music cognition and music theory with machine learning methods. He pursued a doctoral degree on computational models of expressive performance at the Institute of Computational Perception of the Johannes Kepler University Linz, Austria. He received an M.Sc. degree in Electrical Engineering and Audio Engineering from the Graz University of Technology, a degree in Physics from the National Autonomous University of Mexico and a degree in Piano Performance from the National Conservatory of Music of Mexico.

Katerina Kosta is a senior machine learning researcher at ByteDance AI lab. She pursued her Ph.D. from the Centre for Digital Music at the Computer Science and Electronic Engineering department of Queen Mary University of London, conducting research on computational modelling and quantitative analysis of expressive changes of dynamics in performed music. Research interests during her studies included time series analysis, custom data structures, pattern recognition, audio processing, and machine learning for music synthesis and analysis of perceived emotion in music audio. She received degrees from University of Athens (Mathematics) and Filippos Nakas Conservatory, Athens (Piano), and a

Sound and Music Computing Masters from the Music Technology Group at Universitat Pompeu Fabra, Barcelona.

Maarten Grachten is a senior researcher in machine learning for music and sound technology, currently active as an independent machine learning consultant. He holds an M.Sc. in Artificial Intelligence from University of Groningen (The Netherlands, 2001), and a Ph.D. in Computer Science and Digital Communication from Pompeu Fabra University (Spain, 2007). He has worked on computational modeling of musical expression in jazz and classical music since 2001. His work has been funded by European and national research grants at research institutions including the Austrian Research Institute for Artificial Intelligence (OFAI) and Johannes Kepler University (Austria), and has been published in international peer-reviewed conferences and journals.

Tutorial 5

Waveform-based music processing with deep learning

Jongpil Lee, Jordi Pons, and Sander Dieleman

Abstract

A common practice when processing music signals with deep learning is to transform the raw waveform input into a time-frequency representation. This pre-processing step allows having less variable and more interpretable input signals. However, along that process, one can limit the model's learning capabilities since potentially useful information (like the phase or high frequencies) is discarded. In order to overcome the potential limitations associated with such pre-processing, researchers have been exploring waveform-level music processing techniques, and many advances have been made with the recent advent of deep learning.

In this tutorial, we introduce three main research areas where waveform-based music processing can have a substantial impact:

1) Classification: waveform-based music classifiers have the potential to simplify production and research pipelines.

2) Source separation: making possible waveform-based music source separation would allow overcoming some historical challenges associated with discarding the phase.

3) Generation: waveform-level music generation would enable, e.g., to directly synthesize expressive music.

Jongpil Lee received the B.S. degree in electrical engineering from Hanyang University, Seoul, South Korea, in 2015, the M.S. degree, in 2017, from the Graduate School of Culture Technology, Korea Advanced Institute of Science and Technology, Daejeon, South Korea, where he is currently working toward the Ph.D. degree. He interned at Naver Clova Artificial Intelligence Research in the summer of 2017 and at Adobe Audio Research Group in the summer of 2019. His current research interests include machine learning and signal processing applied to audio and music applications.

Jordi Pons is a researcher at Dolby Laboratories. He is finishing a PhD in music technology, large-scale audio collections, and deep learning at the Music Technology Group (Universitat Pompeu Fabra, Barcelona). Previously, he received a MSc in sound and music computing (Universitat Pompeu Fabra, Barcelona), and his BSc was in telecommunications engineering (Universitat Politècnica de Catalunya, Barcelona). He also interned at IRCAM (Paris), at the German Hearing Center (Hannover), at Pandora Radio (USA, Bay Area), and at Telefónica Research (Barcelona).

Sander Dieleman is a Research Scientist at DeepMind in London, UK, where he has worked on the development of AlphaGo and WaveNet. His current research interest is large-scale generative modeling of perceptual signals (images, audio, video). He was previously a PhD student at Ghent University, where he conducted research on feature learning and deep learning techniques for learning hierarchical representations of musical audio signals. In the summer of 2014, he interned at Spotify in New York, where he worked on implementing audio-based music recommendation using deep learning on an industrial scale.

Tutorial 6

Fairness, Accountability and Transparency in Music Information Research (FAT-MIR)

Andre Holzapfel, Marius Miron, Bob L. Sturm, Emilia Gómez

Abstract

This tutorial focuses on the timely issues of ethics, fairness, accountability and transparency, with particular attention paid to research in applications in music information research. These topics arise from a broader consideration of ethics in the field – related work of which was recently published in TISMIR: https://transactions.ismir.net/articles/10.5334/tismir.13. These topics are also receiving attention in the broader domain of machine learning and data science, e.g., the FAT-Machine Learning (ML) conference 2014-2018, Explainable AI workshops 2017-2018, Interpretable Machine Learning workshops, and in the context of the HUMAINT project and winter school on ethical, legal, social and economic impact of Artificial Intelligence (https://ec.europa.eu/jrc/communities/en/community/humaint). This tutorial is suitable for researchers and students in MIR working in any domain, as these issues are relevant for all MIR tasks, from low- to high-level, from system to user-centered research. There are no prerequisites for taking this tutorial.

Andre Holzapfel received M.Sc. and Ph.D. degrees in computer science from the University of Crete, Greece, and a second Ph.D. degree in music from the Centre of Advanced Music Studies (MIAM) in Istanbul, Turkey. He worked at several leading institutes in computer engineering as postdoctoral researcher, with a focus on rhythm analysis in music information retrieval. His field work in ethnomusicology was mainly conducted in Greece, with Cretan dance being the subject of his second dissertation. In 2016, he became Assistant Professor in Media Technology at the KTH Royal Institute of Technology in Stockholm, Sweden. Since then, his research subjects incorporate the computational analysis of human rhythmic behavior by means of sensor technology, and the investigation of ethical aspects of computational approaches to music.

Marius Miron is a Postdoctoral researcher for European Commission's Joint Research Centre within the project HUMAINT, working on fairness and interpretable machine learning and on assessing the influence of artificial intelligence on humans. He has a PhD (2018) in Computer Science (Audio Signal Processing and Machine Learning) from Pompeu Fabra University, Barcelona. His PhD thesis concerned separating the audio corresponding to the instruments in an orchestral music mixture. He has completed internships at Computational Perception Group, Johannes Kepler University, Linz where he worked on deep learning for source separation, and at Telefonica Research, Barcelona, where he worked on catastrophic forgetting in machine learning. During 2011-2013 he was a research engineer for the research institute INESC in Porto for a project aiming at modelling groove in music.

Bob L. Sturm received the B.A. degree in physics from University of Colorado, Boulder in 1998, the M.A. degree in Music, Science, and Technology, at Stanford University, in 1999, the M.S. degree in multimedia engineering in the Media Arts and Technology program at University of California, Santa Barbara (UCSB), in 2004, and the M.S. and Ph.D. degrees in Electrical and Computer Engineering at UCSB, in 2007 and 2009. In Dec. 2014, he became a

Lecturer at the Centre for Digital Music at Queen Mary University of London. In July 2018 he became an associate professor of computer science at the Royal Institute of Technology KTH in Stockholm Sweden.

Emilia Gómez leads the HUMAINT (HUman and MAchine INTelligence) team at the Centre for Advanced Studies, Joint Research Centre (European Commission) and the MIR (Music Information Research) lab of the Music Technology Group, Universitat Pompeu Fabra in Barcelona. Her research background is in music information retrieval, where she has particularly focused on pitch, melody and tonal processing of music audio signals. She also researches more widely on the impact of artificial intelligence technologies on human behaviour. She is a Telecommunication Engineer (Universidad de Sevilla, Spain), DEA in Acoustics, Signal Processing and Computer Science applied to Music (IRCAM, Paris) and Ph.D. in Computer Science (Universitat Pompeu Fabra). Emilia Gómez has co-authored more than 130 scientific publications in peer-reviewed scientific journals and conferences, and contributed to several open datasets and software libraries. She is currently president of the International Society for Music Information Retrieval (ISMIR), and particularly interested in promoting diversity in the MIR field.

20th Anniversary Papers

DATA USAGE IN MIR: HISTORY & FUTURE RECOMMENDATIONS

Wenqin Chen[†] **Jessica Keast**[†] **Jordan Moody**[†] **Corinne Moriarty**[†]
Felicia Villalobos[†] **Virtue Winter**[†] **Xueqi Zhang**[†] **Xuanqi Lyu**
Elizabeth Freeman **Jessie Wang** **Sherry Cai** **Katherine M. Kinnaird**
Smith College, Northampton Massachusetts, USA
kkinnaird@smith.edu

ABSTRACT

The MIR community faces unique challenges in terms of data access, due in large part to country-specific copyright laws. As a result, there is an emerging divide in the MIR research community between labs that have access to music through large companies with abundant funds, and independent labs at smaller institutions who do not have such expansive access. This paper explores how independent researchers have worked to overcome limitations of access to music data without contributing to the crisis of reproducibility. Acknowledging that there is no single solution for every data access problem that smaller labs face, we propose a number of possibilities for how the MIR community can bridge the gap between advancements from large companies and those within academia. As MIR looks towards the next 20 years, democratizing and expanding access to MIR research and music data is critical. Future solutions could include a distributed MIREX system, an API designed for MIR researchers, and community-led advocacy to stakeholders.

1. INTRODUCTION

Since its very conception, the field of Music Information Retrieval (MIR) has struggled with data accessibility, due in part to the nature of music copyright. To deal with this, MIR researchers have developed methods for avoiding or circumventing copyright infringement. Said methods include relying upon public domain and/or Creative Commons music, attainment of certain licenses and/or permissions, use of private in-house data sets, and the access of music data without specifically accessing the audio itself. Each of these methods, however, possess inherent drawbacks such as cost, lack of diverse data, or challenges associated with acquiring required licenses. Public domain and Creative Commons music is often constrained to Western classical pieces; in-house data sets require financial investment; and music data without audio often fail to give a full picture. These drawbacks contribute to an overall disparity of access to data in the MIR community and add to the crisis of reproducibility. In an attempt to strengthen both MIR research and the MIR community as a whole, this paper will propose a number of potential solutions to the drawbacks in existing methods in the hopes that they may serve as a guiding future direction.

The paper is organized as follows. In Section 2, we report on the current concerns and constrictions to data access for MIR researchers. In Section 3, we discuss how datasets from the first ISMIR in 2000 differ from the 10th ISMIR in 2009, and how both compare to the most recent ISMIR in 2018. In Section 4, we outline three proposals for expanding and enhancing access to music data for researchers. Finally, in Section 5, we challenge the community to think urgently about the future and how we can support efforts to improve access to music data.

2. MOTIVATION AND BACKGROUND

Over the last 20 years, society's access to music has evolved. At the first ISMIR conference in 2000, we could only dream of a world with smartphones complete with numerous personalized music streaming applications that make music readily available. Platforms like Spotify and SoundCloud – that now allow for artists to directly share music with the world – had yet to be a pervasive reality.

While the rise of these technologies allows end users to enjoy music more easily and expands MIR research greatly in scope, data commonly used MIR research has not necessarily become more accessible. As with any rapidly growing field, careful attention needs to be paid to legal and scientific concerns with regards to data. In this section, we provide a historical overview of copyright in the United States as it relates to MIR research (though similar statements could be said for other countries). We also discuss the importance of reproducibility for MIR studies.

2.1 History of US Copyright and MIR research

As every MIR researcher is keenly aware, musical recordings are closely protected by copyright laws, which vary by country. In this paper, we treat copyright law in the

† Joint first authors with equal contribution

United States as a case study. The Copyright Act of 1790 marks the beginning of copyright in the US [26]. In 1998, just before the first ISMIR conference, one of the most extensive laws passed was the Digital Millennium Copyright Act (DMCA). This law dealt with issues that arose due to the advent of computers, including allowing data to be copied temporarily during computer maintenance and the ease of broadcast data over the Internet. It also facilitated the implementation of World Intellectual Property Organization (WIPO) Internet treaties, a set of international norms aimed at preventing unauthorized access to and use of creative works on the internet [26]. [1]

As technology has continued to evolve, so too have US laws. In 2018, the Music Modernization Act (MMA) was passed to establish a system of music licensing for digitally distributed music of today [4]. The MMA designates a mechanical licensing collective to oversee a database of musical works that is made publicly available, as well as ensure that artists are paid for their work through the administration of a blanket license. MMA also specifies what songs are in the public domain. With a clearer regulation on music copyright and establishment of a centralized database to track down songwriters, MMA helps reduce unnecessary US copyright related lawsuits while protecting the interests of the music creators. However, given the global nature of the MIR community, being compliant with US laws alone is not enough. As a result, researchers tend to opt for international open source licenses such as Creative Commons (see Section 3.2).

2.2 Reproducibility and Validity of Research

Two vital aspects of scientific research are reproducibility (which encodes reliability) and validity (which often is a proxy for generalizability). These concepts determine how "good" research is – a study that cannot be generalized or trusted is not worth doing.

As MIR tasks grow increasingly complex, it is urgent that the community address the crisis of reproducibility and how it affects the reliability of research. At ISMIR 2014, Raffel et. al. reported that differing evaluation implementations can produce deviations of 9-11% in commonly used metrics across diverse tasks including beat tracking, structural segmentation, and melody extraction [29]. The ability of researchers to verify each other's work is an integral step in the scientific process; without it, consensus on new findings is difficult to reach. That process, however, has been hindered by the use of copyright-protected data in MIR research. Privatized data cannot be legally shared between authors, thus preventing proper re-evaluation for reproducibility [24].

In addition, private and copyright-protected data can be expensive to procure, thus incentivizing and often limiting MIR researchers to using relatively small datasets. This is unfavorable when considering the external validity of a study. It is a general consensus, as Sturm puts eloquently, that "experimental power increases with the number of observations" [35]. Using larger datasets helps control for confounding variables and can sometimes reveal subtle patterns that smaller collections would not pick up on [6]. Smaller datasets carry none of these benefits in addition to having weaker experimental power, greater susceptibility to variation, and revealing only the more obvious patterns. Moreover, limited access to material leads to "restricted and biased subsets," which are difficult to generalize to larger music populations and thus bring into question the external validity of such studies [37].

[1] https://www.wipo.int/copyright/en/activities/internet_treaties.html

3. DATA USAGE & EVOLUTION

In this section, we explore data usage and data evolution as well as features of datasets commonly used today. We investigate different classifications of data explored in papers in three different ISMIR conferences. We then discuss the evolution of data creation and examine the limitations of popular datasets.

3.1 Dataset Classification

To illustrate how data usage has evolved over the last 20 years, this paper analyzes datasets used in published papers from ISMIR 2000 [1], 2009 [2], and 2018 [3]. This is done in order to answer three key questions. First, are these datasets diverse enough? Second, how were these datasets created and did the process get easier or harder over time? Third, how were these datasets released and what was their impact?

3.1.1 Genre Classification

To address the first question, we investigate how different music genres have been represented at ISMIR. To do so, we examined the published submissions from the first (in 2000), 10^{th} (in 2009), and most recent (in 2018) ISMIR conferences. Each paper in the respective proceedings was analyzed by three of the authors of this paper, and all music data used for analysis, training, or testing was classified by genre. Although most papers explicitly stated what type of music was used, any disagreements regarding dataset classification were discussed until there was consensus. The results of this survey are displayed in Figure 1. Datasets that used four or fewer genres were classified with each of the genres, those with five or more were given the genre label "various". [2] As many datasets contained multiple genres and many papers referenced multiple datasets, there may be more than one genre representing each paper. Any genre only used once in a given year was categorized as "Other," such as the analysis of opera singing in ISMIR 2018 by Parada-Cabaleiro et.al. [28].

We find that the proceedings in ISMIR 2018 [3] and ISMIR 2009 [2] used data from more music genres, and have a more equal distribution among these genres than those in 2000 [1]. The proportion of pop and rock songs used decreased over time as other genres were included. The proportion of papers that did not use music data or did not

[2] Datasets with at least one non-Western genre were labeled both "various" and "non-Western." Datasets with only Western genres were given no other genre labels

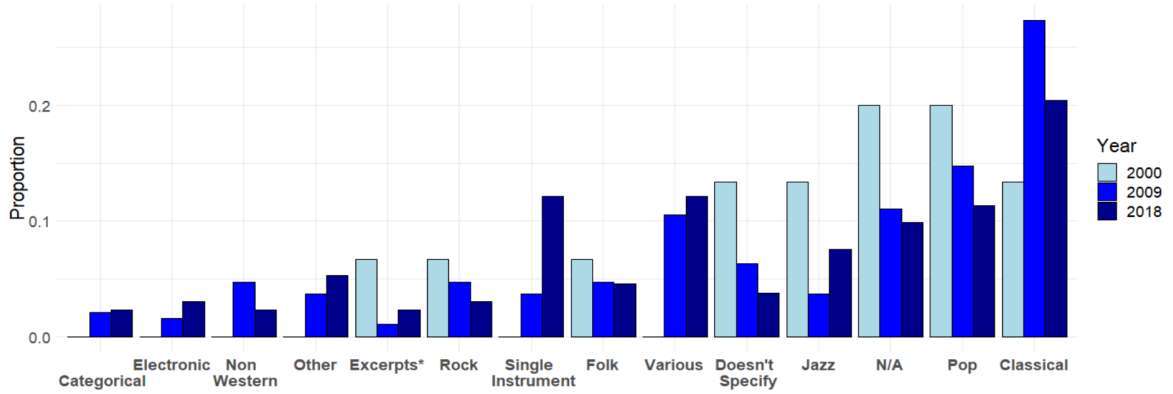


Figure 1: Proportion [3] of the number of papers that use different genres of data from first ISMIR conference in 2000 [1] to the 10th ISMIR in 2009 [2], to the 19th ISMIR in 2018 [3]. "Excerpts*" refers to music excerpts under 3 seconds, and "categorical" refers to music selected for a non-genre category such as mood. The "non-Western" category does not include genres such as J-pop and K-pop, which were classified as solely "pop".

specify the type of music they used have also decreased over time. The latter may be due to an increasing emphasis on reproducibility in MIR, researchers becoming more diligent in documenting their data sources so that others may accurately reproduce their work.

The diversity of MIR research has increased over time. For example, "electronic", "categorical", "non-Western", and "single instrument" music were studied in both 2009 and 2018, even though they had not been in 2000. There has been a substantial increase in papers that consider five or more different genres. However, non-Western genres remain comparatively under-studied, and "classical" continues to be one of the most-studied genres in MIR.

3.1.2 Dataset Collection and Release

To address the second and the third questions, four authors of this work determined the data classifications for datasets used in ISMIR 2018. The categories can be applied to both the dataset creation process and the dataset release process and are summarized in Table 1. [4]

- Creative Commons – CC – includes all datasets released under a Creative Commons license, such as MedleyDB [7]. [5]
- Public Domain – PD – includes all freely available, downloadable, open source datasets that are not licensed under Creative Commons, including datasets released on GitHub; for example, The Million Song Dataset (MSD) [6]. [6]
- Commercial datasets – COM – include those that are owned by for-profit companies such as Spotify and Deezer, and are therefore either not available for public use or available only for purchase.
- In-house datasets – IH – refers to unshareable datasets that are privately owned or contributed via personal connections. For example, some of MedleyDB's data are in-house, though the database is released under CC.
- Library held datasets – LIB – refer to all datasets that are owned and maintained by libraries, or research institutions, including archives dedicated to preserving historical audio records; for example, the University of Iowa Musical Instrument Samples. [7]
- Meta library – MET – refers to aggregates of several existing datasets; for example, the MuMu dataset combines info from MSD and Amazon Reviews [27].
- Permission needed category – PER – refers to any dataset that needs explicit permission to access, but are available for free or at a minimal cost so as to distinguish from the commercial category. An example would be the RWC database [18]. [8]
- Short clips category – SHO – consists of datasets that are made up of short audio samples (usually less than 30 seconds) or recordings of single notes and chords. The NSynth dataset is one example [16]. [9]

3.2 Evolution of dataset creation process: Creative Commons and open source practices

Creative Commons [34] is an international system that creators can use to offer certain usage rights to the public while reserving other rights. This licensing allows content to be distributed within the boundaries of copyright laws, ensuring that creators get credit for their work while allowing for non-commercial distribution and use of the

[3] Proportion out of all genres labels generated for music data used for each paper in that year's ISMIR. The authors found 15 instances of genre usage in ISMIR 2000, 190 in ISMIR 2009, and 132 in 2018.
[4] Note that one dataset can be put into multiple categories.
[5] https://medleydb.weebly.com/downloads.html
[6] http://millionsongdataset.com/
[7] http://theremin.music.uiowa.edu/MIS.html
[8] https://staff.aist.go.jp/m.goto/RWC-MDB/
[9] https://magenta.tensorflow.org/datasets/nsynth

Category	Creation Count	Release Count
CC	10	22
PD	16	32
COM	9	5
IH	18	5
LIB	9	9
MET	9	0
PER	4	9
SHO	12	12
unclear	17	12

Table 1: Classifications of datasets in ISMIR 2018 [3]. Creation count is the number of datasets in each form as gathered by dataset creator, while release count refers to the number of datasets in each form as released for use. There were a number of papers that were not clear on the dataset that they used.

copyrighted material [9, 19]. There are six types of Creative Commons licenses, all applicable worldwide [9, 19].

Before the emergence of Creative Commons in 2002 [8], early MIR research used small and idealized data sets [24]. This is evident in the proceedings of the first ISMIR conference held in 2000, which suffered from a lack of data. Out of the ten papers published in the proceedings, only six described the datasets used. Out of these six, just two papers – [21, 30] – used existing databases, while the other four had to gather data on their own. In comparison with ISMIR 2000, ISMIR 2018 has expanded not just in the scope and complexity of MIR research, but also in the number of datasets referenced, leveraging the resources licensed under Creative Commons or in the public domain.

3.3 Popular Databases Today

Some of the most influential datasets are *Million Song Dataset* (*MSD*), *RWC* and *MedleyDB*. As of June 2019, according to Semantic Scholar, these sets were cited 560, 364, and 124 times respectively. [10] Some CC and OPS datasets such as *Jamendo* are popular, but their influence is difficult to quantify because citations are often not explicitly required. [11] Both *MedleyDB* and *Jamendo* rely on Creative Commons for copyright-free distribution of music files [7, 20].

In ISMIR 2018 alone, the *MSD* was used in 11 different papers, making it one of the most frequently cited datasets in that conference. Released in 2011, the *MSD* is a freely available collection of derived Echo Nest features and metadata of one million contemporary Western commercial songs [6]. The dataset can be used along with `7digital` to fetch short samples of songs within certain limitations for free [6].

This lack of audio is not unusual; our investigations (described in Section 3.1) found that about one-fifth of all datasets used in ISMIR 2018 do not contain any audio files. Research on this kind of data is limited to the provided features (by Echo Nest in the case of *MSD*), and does not allow for the kind of in-depth investigation that audio data provides. However, the model of collaborating with free, community-driven datasets as well as commercial companies has served as a good solution for expanding the scale of MIR datasets without violating copyright law.

Due to copyright limitations, many MIR researchers also choose to use music databases that consist exclusively of public domain works, Creative Commons works, or those with limited features of the raw audio. The *Classical Music Archive*, for instance, is a large repository of classical music with each track existing in public domain [5]. The pervasiveness of classical music (and dirth of other genres) in the public domain leads to research being greatly skewed towards the classical genre with other genres being explored much less. Although this research remains important, it is worth pointing out that the vast majority of music produced today is not classical, and is not encompassed by these studies.

Other datasets avoid copyright infringement by omitting material subject to copyright. For instance, *AcousticBrainz* is a publicly available database that is entirely composed of features extracted from songs, rather than the songs themselves [5]. *AcousticBrainz* extracts features on the song level, accumulating the values researchers might want to use and associating them with songs without storing any audio [5]. Similarly, in 2006, *OMEN* was proposed as a system of feature extraction that would take place in libraries [22]. The proposal indicates a method of communication between researchers and libraries that would allow researchers to request specific features from a specific song that would then be extracted by librarians. Frameworks such as *OMEN* would help researchers circumvent copyright and the cost of in-house datasets while protecting content created by artists through the separation of music from its features to create more available music data.

3.4 Many Datasets; Yet Limited Data & Data Access

Despite the numerous cited datasets, an issue repeatedly raised within MIR is the existence of bias [12] in data commonly used in the field. Bias is frequently inevitable due to lack of resources. For example, the structural segmentation task in MIREX 2010 used two datasets with 397 songs in total almost exclusively biased towards Western popular music [15]. This was partially because the data were donated by a few universities and were therefore limited in quantity and diversity. A new annotated dataset with over 1,300 songs covering pop, jazz, classical and world music then became available in 2011 [15], and has been used since in the structural segmentation task of MIREX.

Community and partnership-driven data initiatives have provided working data access to further MIR research. However each member of the MIR community faces

[10] Accessed on June 23, 2019

[11] https://www.jamendo.com/

[12] Bias can exist in terms of geographic location, musical instruments, music genres, musical styles and musical fundamentals such as key, harmony, tempo and rhythm.

unique hindrances to accessing data. *MSD* and Creative Commons currently help researchers bypass these challenges with innovative solutions, but do not support a data access framework for the MIR community as a whole.

4. PROPOSALS

As concerns about access and reproducibilty mount, we offer proposals to increase data access that draw upon resources existing within the MIR community. These suggestions seek to address the crisis of reproducibility while working within current copyright law.

The three proposals below tackle these issues from three different parts of the MIR community. The first calls upon our academic partners to build a distributed MIREX. The second asks our industrial partners to leverage their existing infrastructures. The third asks the whole community to expand our support for dataset creators. Noting that MIR researchers around the world fall under different copyright laws given their location, some of these proposals focus on actions that could be taken in the United States (to the benefit of the worldwide community), while others provide potential global solutions.

4.1 Academic Proposal: Distributed MIREX

Our first proposal is a new, decentralized, distributed Music Information Retrieval Exchange (MIREX) system that would build on the success of the existing MIREX infrastructure, while leveraging previously untapped resources to support the development of such a system. Drawing on the strengths of the proposals in [11–13], this distributed MIREX seeks to be a middle ground between small, in-house datasets and large industrial ones – allowing researchers to test algorithms without violating copyright or relying solely on non-copyrighted music.

This proposed system would distribute the responsibilities and challenges associated with running MIREX among three institutions, by creating two new MIREX sites in addition to the current one run by IMIRSEL at University of Illinois Urbana-Champaign (UIUC). Though the startup cost of a new MIREX site is substantial (data, servers, and other infrastructure needs), creating several MIREX locations is a long-term investment in MIR research and future researchers. More MIREX locations will increase the community's opportunities to access music datasets and allow for additional comparisons between algorithms.

4.1.1 Current MIREX

MIREX was created by IMIRSEL as a method to make comparisons between algorithms. The system's process of submitting, running, and returning results ensures that the music data being used is in accordance with US copyright laws [10]. Though MIREX has been a successful tool for MIR research, it presents challenges of accessibility and efficiency. In response to these challenges, several MIR papers have called on the community for help and have proposed a distribution of MIREX responsibilities including: a web service system to give some algorithm execution responsibility to the user [13], a distributed task management [11], a breakdown of the MIR process across labs participating in the NEMA framework with inter-lab communication [12], and a proposal for a distributed computation model that leverages open source methods [23].

4.1.2 Logistics of Distributed MIREX

The first step in creating a distributed MIREX is to identify the additional two sites. We propose that the three sites would not operate in a coordinated fashion, as suggested in [11, 12], other than ensuring that the timings of the runs do not overlap. Undergraduate-focused institutions – common in the US – are ideal for selection as new MIREX locations, as these schools tend to have existing systems for supporting undergraduate student research and student work-study programs.

Within the MIR community, MegsRadio, under the direction of Douglas Turbull, is an example of an undergraduate student driven MIR project that thrived at Ithaca College [36]. MegsRadio ran throughout the year with students addressing technical issues as well as working on user-facing elements of their work [36]. Student involvement similar to that in MegsRadio is what we envision for the proposed distributed MIREX. Additionally, research has suggested that students who do undergraduate research in STEM may be more likely to pursue a graduate/professional degree in a STEM field [14]. It stands to reason that these findings could be extended to MIR, especially the field's technical aspects. As such, intentionally involving more undergraduates in MIR research will help broaden the field and provide some of the labor required to maintain a distributed MIREX system.

In addition to the potential of undergraduate workers, our distributed MIREX's ability to run algorithms on different datasets multiple times per year could allow researchers the the chance to test their work more than once a year. Furthermore, this system could lessen the individual infrastructure-related responsibility of handling the "interim data explosion" problem [12] caused by specific algorithms. More locations could allow for more storage of features created and generated by algorithms, which the current MIREX discards due to storage limitations [12]. The task of maintaining not only the data itself, but any features created by algorithms, would be distributed to various MIREX sites – thus addressing the infrastructure capacity issues in the current model. The current system entails engaging in several hours of intensive back and forth communication with participants about their algorithms as described in [11]. Instead, this distributed MIREX could increase access to more diverse and quality datasets over time as well as foster growth in the MIR community. Additionally, the creation of the distributed MIREX would compliment open source systems like those proposed in [23] at ISMIR 2016.

4.2 Industrial Proposal: Researcher API License

As the MIR field has grown, so have industrial research groups and public-facing technologies. To complement

our academic proposal, we advocate developing a researcher API license that leverages existing industrial infrastructure. This proposed new license would be designed with the workflow of MIR research practices in mind and would allow for actions that are completely disjointed from current developer licenses.

Developer licenses are designed with software developers in mind and possess inherent restrictions that make them unsuited for MIR research. For example, a streaming phone app that allows for streaming from a different platform would be created under a developer license. But actions such as feature extraction, harmonic-percussive separation, and sampling are in violation of most developer licenses (for example, see Section IV.1.a of [33]). These guidelines leave very little room for MIR research of any kind, and similar language exists on other APIs' terms of use, including Soundcloud's API terms of use that prohibits the using of any data from their platform (see [32]).

We propose that companies with existing developer licenses and API infrastructure (such as Spotify and SoundCloud) create a new license that explicitly allows for common MIR practices such as feature extraction and explicitly prohibits using algorithm outputs for commercial or listening purposes, but limits casual listening. To create appropriately narrow language, companies could consult with IMIRSEL who forged similar agreements to build the MIREX [10]. Such a license addresses concerns about data biased towards popular music, as smaller artists can directly share their work on these streaming platforms.

An immediate and real concern with such a license is the potential for abuse. To help address this concern, we propose the creation of an ethics training module similar to those taken by researchers conducting work using human and animal subjects (colloquially referred to as IRB courses). In this ethics training module, MIR researchers would be educated on the ethical, cultural, and financial issues at play in using and misusing music data. MIR researchers would also have to complete various assessments that certify their understanding of these nuanced issues.

This proposal has the potential for global reach as many companies already operate in several countries, and seeks to provide access to more diverse data.

4.3 Community Proposal: Usage and Advocacy

In addition to creating a decentralized MIREX system and an academic API license, the MIR community must also continue to support the current creators and maintainers of research datasets, not only through their use but through advocacy of laws that make them possible in the first place. The passing of laws such as the Music Modernization Act in the US (Section 2.1) affect the MIR community, but the community had no influence on the creation or implementation of this policy.

Since ISMIR is affected by these policy changes, it is important for the society to know about the current policies. It would be beneficial to be involved in the politics behind these laws to ensure that the voices of researchers are heard. Many research societies have created frameworks for government communication. The American Association for the Advancement of Science (AAAS) has the Center for Science Diplomacy, which connects science to policy and allows the strengthening of connections between diplomatic and science communities [17]. The American Mathematical Society (AMS) is a part of the Coalition for National Science Funding and the Task Force on American Innovation, both of which are alliances of professional, academic, and scientific organizations that advocate for issues that affect the members and their communities [31]. These alliances and committees ensure that these fields of research are not negatively affected by policy changes and that the voices of the societies are heard.

As stated on the ISMIR website, one of the purposes of the ISMIR Organization is "to cooperate with representatives of other organizations and disciplines toward the furtherance of music information retrieval" [25]. To achieve this goal, it is necessary to cooperate not only with other research societies but also with the government bodies that create the infrastructures that MIR researchers work under. The creation of an ad-hoc committee or some other system focused on current music policy changes and advocating for music research in policy would benefit the MIR community. Allying with other music research organizations such as the American Musicological Society or the International Musicological Society could also help facilitate change in music policy for the betterment of research. These systems would benefit both the society and the growth of the MIR field as a whole.

5. CONCLUSION: TOWARDS A STRONGER MIR COMMUNITY

We have proposed three possibilities for expanding researcher access to music data. While progress on any of these three proposals will better the field, simultaneous efforts will lead to a collective impact that is greater than the sum of each proposal's individual impacts. For example, as new MIREX locations begin, industrial researchers can consult on how to design and build appropriate infrastructure given the school's constraint. Similarly, in the US model, undergraduate students working in MIR research groups would make excellent testers for early versions of companys' researcher API licenses. Finally, with the creation of an ad-hoc committee (supported by the ISMIR board) representing the interests of MIR researchers to policy-makers and vice versa, the whole MIR community will be better informed about current laws concerning music. This increased awareness will help to continually improve both academic and industrial research efforts.

Unequal access to music data has led to field-wide issues including a crisis of reproducibility and concerns about access. Despite this, future directions and solutions do exist. By implementing the multi-pronged approach outlined in this paper, MIR research can adapt to overcome these obstacles and avoid the looming schism between corporate and independent labs. Moreover, the methods we have proposed will foster cooperation within the field and could lead to greater diversity within MIR.

6. ACKNOWLEDGEMENTS

The last author is the Clare Boothe Luce Assistant Professor of Computer Science and Statistical and Data Science at Smith College and as such, is supported by Henry Luce Foundation's Clare Boothe Luce Program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Luce Foundation.

7. REFERENCES

[1] ISMIR 2000: International Symposium on Music Information Retrieval. http://ismir2000.ismir.net/papers.html, 2000. [Online; accessed 3-April-2019].

[2] ISMIR 2009: 10th International Society for Music Information Retrieval conference. https://ismir2009.ismir.net/program.html, 2009. [Online; accessed 20-June-2019].

[3] ISMIR 2018: 19th International Society for Music Information Retrieval conference. http://ismir2018.ircam.fr/pages/events-main-program.html, 2018. [Online; accessed 3-April-2019].

[4] 115th Congress. H.R.5447 - Music Modernization Act. https://www.congress.gov/bill/115th-congress/house-bill/5447/text, 2018. [Online; accessed 10-April-2019].

[5] Y. Bayle, M. Robine, and P. Hanna. SATIN: a persistent musical database for Music Information Retrieval and a supporting deep learning experiment on song instrumental classification. *Multimedia Tools and Applications*, 78(3):2703–2718, 2019.

[6] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere. The Million Song Dataset. In *Proc. of* 10^{th} *ISMIR Conference*, pages 591–596, 2011.

[7] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *Proc. of* 15^{th} *ISMIR Conference*, pages 155–160, 2014.

[8] Creative Commons. The story of Creative Commons. https://certificates.creativecommons.org/cccertedu/chapter/1-1-the-story-of-creative-commons/. [Online; accessed 28-June-2019].

[9] Creative Commons. About the licenses. https://creativecommons.org/licenses/, 2018. [Online; accessed 10-April-2019].

[10] J. S. Downie, J. Futrelle, and D. K. Tcheng. The international Music Information Retrieval systems evaluation laboratory: Governance, access and security. In *Proc. of* 5^{th} *ISMIR Conference*, 2004.

[11] J. S. Downie, X. Hu, J. H. Lee, K. Choi, S. J. Cunningham, and Y. Hao. Ten years of MIREX: Reflections, challenges and opportunities. *Proc. of* 15^{th} *ISMIR Conference*, pages 27–31, 2014.

[12] J.S. Downie. The Music Information Retrieval Evaluation Exchange (2005-2007): A window into Music Information Retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[13] J.S. Downie, A. Ehmann, and J. Lee. The Music Information Retrieval Evaluation eXchange (MIREX): Community-led formal evaluations. *Digital Humanities (Conference)*, 29(4):239–241, 2008.

[14] M. K. Eagan Jr., S. Hurtado, M. J. Chang, G. A. Garcia, F. A. Herrera, and J. C. Garibay. Making a difference in science education: The impact of undergraduate research programs. *American Educational Research Journal*, 50(4):683–713, 2013. PMID: 25190821.

[15] A. F. Ehmann, M. Bay, J. S. Downie, I. Fujinaga, and D. De Roure. Music structure segmentation algorithm evaluation: Expanding on MIREX 2010 analyses and datasets. In *Proc. of* 12^{th} *ISMIR Conference*, pages 561–566, 2011.

[16] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan. Neural audio synthesis of musical notes with wavenet autoencoders. In *Proc. of the 34th International Conference on Machine Learning - Vol. 70*, ICML'17, pages 1068–1077. JMLR.org, 2017.

[17] The American Association for the Advancement of Science. Science diplomacy. https://www.aaas.org/focus-areas/science-diplomacy, 2019. [Online; accessed 11-April-2019].

[18] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Popular, classical and jazz music databases. In *Proc. of* 3^{rd} *ISMIR Conference*, pages 287–288, 2002.

[19] C. Green. Open licensing and open education licensing policy. In *Open: The Philosophy and Practices that are Revolutionizing Education and Science*, pages 29–42. Ubiquity Press, 2017.

[20] Jamendo. Jamendo music. https://www.jamendo.com/, 2019. [Online; accessed 8-April-2019].

[21] P. Jeremy. A comparison of language modeling and probabilistic text information retrieval approaches to monophonic music retrieval. In *Proc. of* 1^{st} *ISMIR Conference*, 2000.

[22] D. McEnnis, C. McKay, and I. Fujinaga. Overview of OMEN. In *Proc. of* 7^{th} *ISMIR Conference*, pages 7–12, 2006.

[23] B. McFee, E.J. Humphrey, and J. Urbano. A plan for sustainable MIR evaluation. In *17th International Society for Music Information Retrieval Conference*, ISMIR, 2016.

[24] B. McFee, J. W. Kim, M. Cartwright, J. Salamon, R. M. Bittner, and J. P. Bello. Open-source practices for music signal processing research: Recommendations for transparent, sustainable, and reproducible audio research. *IEEE Signal Processing Magazine*, 36(1):128–137, Jan. 2019.

[25] The International Society of Music Information Retrieval. About the society. `https://www.ismir.net/society.php`, 2017. [Online; accessed 11-April-2019].

[26] Association of Research Libraries. Copyright timeline: History of copyright in the United States. `https://www.arl.org/focus-areas/copyright-ip/2486-copyright-timeline`, 2019, continually updated. [Online; accessed 08-April-2019].

[27] S. Oramas, O. Nieto, F. Barbieri, and X. Serra. Multi-label music genre classification from audio, text and images using deep features. In *Proc. of* 18^{th} *ISMIR Conference*, pages 23–30.

[28] E. Parada-Cabaleiro, M. Schmitt, A. Batliner, S. Hantke, G. Costantini, K. Scherer, and B. W. Schuller. Identifying emotions in opera singing: Implications of adverse acoustic conditions. In *Proc. of* 19^{th} *ISMIR Conference*, pages 376–382, 2018.

[29] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis. `mir_eval`: a transparent implementation of common MIR metrics. *Proc. of* 15^{th} *ISMIR Conference*, 2014.

[30] C. G. Sayeed, D. Michael, D. Tim, F. Ichiro, and H. Brian. Optical music recognition system within a large-scale digitization project. In *Proc. of* 1^{st} *ISMIR Conference*, 2000.

[31] The American Mathematical Society. Office of government relations collaborations and position statements. `https://www.ams.org/government/dc-collaborations`, 2019. [Online; accessed 11-April-2019].

[32] Soundcloud. API terms of use. `https://developers.soundcloud.com/docs/api/terms-of-use`, 2016. [Online; accessed 5-April-2019].

[33] Spotify. Spotify developer terms of service. `https://developer.spotify.com/terms/`, 2018. [Online; accessed 3-April-2019].

[34] E. Steuer. Response to ASCAP's deceptive claims. `https://creativecommons.org/2010/06/30/response-to-ascaps-deceptive-claims/`, Jun. 2010. [Online; accessed 28-June-2019].

[35] B. L. Sturm. Revisiting priorities: Improving MIR evaluation practices. In *Proc. of* 17^{th} *ISMIR Conference*, pages 488–494.

[36] D. Turnbull, C. Perez, D. Dorsey, J. Burger, J. Menduni, E. Piech D. Akimchuk, A. Python, and T. Joachims. Developing a professional-grade personalized radio app in an academic setting. *Late breaking at* 17^{th} *ISMIR Conference*, 2017.

[37] J. Urbano, M. Schedl, and X. Serra. Evaluation in Music Information Retrieval. *Journal of Intelligent Information Systems*, 41:345–369.

MUSIC PERFORMANCE ANALYSIS: A SURVEY

Alexander Lerch **Claire Arthur** **Ashis Pati** **Siddharth Gururani**
Center for Music Technology, Georgia Institute of Technology, Atlanta, USA
{alexander.lerch,claire.arthur,ashis.pati,siddgururani}@gatech.edu

ABSTRACT

Music Information Retrieval (MIR) tends to focus on the analysis of audio signals. Often, a single music recording is used as representative of a "song" even though different performances of the same song may reveal different properties. A performance is distinct in many ways from a (arguably more abstract) representation of a "song," "piece," or musical score. The characteristics of the (recorded) performance —as opposed to the score or musical idea— can have a major impact on how a listener perceives music. The analysis of music performance, however, has been traditionally only a peripheral topic for the MIR research community. This paper surveys the field of Music Performance Analysis (MPA) from various perspectives, discusses its significance to the field of MIR, and points out opportunities for future research in this field.

1. INTRODUCTION

Music, as a performing art, requires a performer or group of performers to render a musical score into an acoustic realization [38]. This is also true for non-classical music: for example, the 'score' might be a lead sheet or only a structured sequence of musical ideas, a 'performer' could also be a computer rendering audio, and the acoustic realization might be represented by a recording. The performance plays a major role in how listeners perceive a piece of music: even if the score content is identical for different renditions, as is the case in western classical music, listeners may prefer one performance over another and appreciate different 'interpretations' of the same piece of music. These differences are the result of the performers actively or subconsciously interpreting, modifying, adding to, and dismissing score information.

Although the distinction between score and performance parameters is less obvious for other genres of western music, especially ones without clear separation between the composer and the performer, the concept of interpreting an underlying score is still very much present, be it as a live interpretation of a studio recording or a cover version of another artist's song. In these cases, the freedom of the performers' in modifying the score information is often much higher than it is for classical music — reinterpreting a jazz standard can, e.g., include the modification of content related to pitch, harmony, and rhythm.

Performance parameters can have a major impact on a listener's perception of the music [13]. Formally, performance parameters can be structured in the same basic categories that we use to describe audio in general: tempo and timing, dynamics, pitch, timbre [51]. While the importance of different parameters might vary from genre to genre, the following list introduces some mostly genre-agnostic examples to clarify the performance parameter categories:

- *Tempo and Timing* — the score specifies the general rhythmic content, just as it often contains a tempo indicator. While the rhythm is often only slightly modified by performers in terms of micro-timing, the tempo (both in terms of overall tempo as well as expressive tempo variation) is frequently seen only as a suggestion to the performer.
- *Dynamics* — in most cases, score information on dynamics is missing or only roughly defined. The performers will vary loudness based on their plan for phrasing, tension, importance of certain parts of the score, and highlight specific events with accents.
- *Pitch* — the score usually defines the general pitches to play, but pitch-based performance parameters include expressive techniques such as vibrato as well as conscious or unconscious choices for intonation.
- *Timbre* — as the least specific category of musical sound, scores encode timbre parameters often only implicitly (e.g., instrumentation) while performers can, e.g., change playing techniques or the choice of specific instrument configurations (such as the choice of organ registers).

Note that usually the performance to be analyzed is a *recording* and not a live performance; every recording contains processing choices and interventions by sound engineer and editor with potential impact on expressivity — these modifications cannot be separated from the musicians' creation and are thus an integral part of what is investigated [58].

The most intuitive form of Music Performance Analysis (MPA) —discussing, criticizing, and assessing a performance after a concert— has arguably taken place since music was first performed. Early attempts at systematic and empirical MPA can be traced back to the 1930s with vibrato and singing analysis by Seashore [90] and the examination of piano rolls by Hartmann [37]. In the past two decades, MPA has greatly benefited from the advances in audio analysis made by members of the Music Infor-

mation Retrieval (MIR) community, significantly extending the volume of empirical data by simplifying access to a continuously growing heritage of commercial audio recordings. However, while advances in audio content analysis have had clear impact on MPA, the opposite is less true. While there have been publications on performance analysis at ISMIR, the major MIR conference, their absolute number remains comparably small (compare [3, 10, 18, 41, 53, 54, 73, 84, 85, 98, 104, 105] with a title referring to music performance out of approximately 1800 ISMIR papers overall).

Historically, MIR researchers often do not distinguish between score-like information and performance information even if the research deals with audio recordings of performances. For instance, the goal of music transcription, a very popular MIR task, is usually to transcribe all pitches with their onset times [5]; that means that a successful transcription system transcribes two renditions of the same piece of music differently, although the ultimate goal is to detect the same score (note that this is not necessarily true for all genres). Therefore, we can identify a disconnect between MIR research and performance researchers that impedes both the evolution of MPA approaches and robust MIR algorithms, slows gaining new insights into music aesthetics, and hampers the development of practical applications such as new educational tools for music practice and assessment.

The remainder of this paper is structured as follows. The next Sect. 2 presents research on the objective description and visualization of the performance itself, identifying commonalities and differences between performances. The subsequent sections focus on studies taking these objective performance parameters and relating them to either the performer (Sect. 3) or the listener (Sects. 4 and 5). We conclude our overview with a summary on applications of MPA and final remarks in Sect. 6. Note that while performance research has been inclusive of various musical genres, such as the Jingju music of the Beijing opera [33, 118], traditional Indian music [15, 34, 65] and jazz music [1], the vast majority of studies have been concerned with Western classical music. Therefore, the remainder of the paper focuses primarily on Western music.

2. PERFORMANCE MEASUREMENT

A large body of work focuses on a descriptive approach to analyzing performance recordings. Such studies typically extract characteristics such as the tempo curve [69, 75, 77] or loudness curve [82, 90] from the audio and aim at either gaining general knowledge on performances or comparing attributes between different performances/performers based on trends observed in the extracted data.

Several researchers observed a close relationship between musical phrase structure and deviations in tempo and timing [71, 75, 91]. For example, tempo changes in the form of *ritardandi* tend to occur at phrase boundaries [50, 69]. A similar co-occurrence was observed between dynamic patterns and timing [50, 81]. Additionally, Dalla Bella found the overall tempo influences the overall loudness of a performance [16]. There are also indications that loudness can be linked to pitch height [81]. While the close relation of tempo and dynamics to structure has been repeatedly verified, Lerch did not succeed in finding similar relationships between structure and timbre properties in the case of string quartet recordings [50].

Pitch-based performance parameters have been analyzed mostly in the context of single-voiced instruments. Vibrato and its rate has, e.g., been studied for vocalists [17, 90] and violinists [22, 57]. Regarding intonation, Devaney et al. found significant differences between professional and non-professional vocalists in terms of the size of the interval between semi-tones [17].

Other studies use a multitude of performance parameters and aim to identify trends over time. For example, Ornoy and Cohen investigated violin performances of 19th century repertoire recorded in the past two decades [68]. They found a blend of stylistic approaches among violinists which questions the traditional distinction made between a historically informed and a mainstream performance.

The challenges in accessibility and interpretability of the extracted performance parameters have also led researchers to work on more intuitive or condensed forms of visualization that allow describing and comparing different performances beyond the traditional forms of visualization such as tempo curves [69, 75, 77] and scatter plots [50]. The "performance worm," for example, is a pseudo-3D visualization of the tempo-loudness space that allows the identification of specific gestures in that space [23, 48]. Sapp proposed so-called "Timescapes" and "Dynascapes" to visualize subsequence similarities of performance parameters [84, 85].

While most of the studies mentioned above make use of statistical methods to extract, visualize, and investigate patterns in the performance data, few studies make use of Machine Learning (ML) for MPA and performance modeling. However, ML-based approaches are useful for tasks such as composer classification, discovery of performance rules, or modeling performance characteristics. Widmer conducted studies that utilized ML to model expression in musical performance [109, 111, 112] and to learn simple rules from performance data with inductive learning [110]. He also applied ML to identify performers, showing that performer characteristics can be modeled by ML algorithms [112].

The studies presented in this section often follow an exploratory approach; extracting various parameters in order to identify commonalities or differences between performances. While this is, of course, of considerable interest, one of the main challenges is the interpretability of these results. Just because there is a timing difference between two performances does not necessarily mean that this difference is perceptually meaningful. Without this link, however, results can only provide limited insights into which parameters and parameter variations are ultimately important. Another typical challenge in such studies is the reliability of MIR algorithms for automatic annotations. While the accuracy of such algorithms has steadily improved over time, the fact that the majority of studies surveyed here continue to rely on manually-annotated data implies that the state-of-the-art algorithms for automatic annotation still

lack the required degree of accuracy for most tasks.As a result, most analyses are performed on small sample sizes possibly leading to poor generalizability of the studies.

3. PERFORMER

While most studies focus on the extraction of performance parameters or the mapping of these parameters to the listeners' reception (see Sects. 4 and 5), some investigate the capabilities, goals, and strategies of performers. A performance is usually based on an explicit or implicit performance plan with clear intentions [14]. There is, as Palmer verified, a clear relation between reported intentions and objective parameters related to phrasing and timing of the performance [69]. Similar relations between the intended emotionality and loudness and timing measures were reported by Juslin [42] and Dillon [19–21]. For example, projected emotions such as anger and sadness show significant correlations with high and low tempo and high and low overall sound level, respectively. Moreover, a performer's control of expressive variation has been shown to significantly improve the conveyance of emotion. For instance, a study by Vieillard et al. found that listeners were better able to perceive the presence of specific emotions in music when the performer played an expressive (versus mechanical) rendition of the composition [107]. This suggests that the performer plays a fairly large role in communicating an emotional "message" above and beyond what is communicated through the score alone [44]. In addition to the performance plan itself, there are other influences shaping the performance. Acoustic parameters of concert halls such as the early decay time have been shown to impact performance parameters such as tempo [55, 86, 87]. Another interesting area of research is performer error. Repp analyzed performers' mistakes and found that errors were concentrated in mostly unimportant parts of the score (e.g., middle voices) where they are harder to recognize [40], suggesting that performers consciously or unconsciously avoid salient mistakes [80].

There is a wealth of information about performances that can be learned from performers. The main challenge of this direction of inquiry is that such studies have to involve the performers themselves. This limits the amount of available data and possibly excludes well-known and famous artists, resulting in a possible lack of generalizability. Depending on the experimental design, the separation of possible confounding variables (e.g., motor skills, random variations, and the influence of common performance rules) from the scrutinized performance data can be a challenge.

4. LISTENER

Every performance will ultimately be received and processed by a listener. The listener's meaningful interpretation of the incoming musical information relies on a sophisticated network of parameters. These include not only external, or semi-objective parameters such as score or performance-based features, but also "internal" ones such as those shaped by the culture, training, and history of the listener. For this reason, listener-focused MPA remains one of the most challenging and elusive areas of research. However, to the extent that MPA research depends on purely perceptual information (e.g., expressiveness) or intends to deliver perceptually-relevant output (e.g., performance evaluation or reception, similarity), it is imperative to achieve a fuller understanding of the perceptual relevance of the manipulation and interaction of performance characteristics (e.g., tempo, dynamics, articulation).

4.1 Musical expression

When it comes to listener judgments of a performance, it remains poorly understood which aspects are most important, salient, or pertinent for the listener's sense of satisfaction. According to Schubert and Fabian [89], listeners are very concerned with the notion of "expressiveness" which is a complex, multifaceted construct. Discovering which performance characteristics contribute to an expressive performance thus requires dissecting what listeners deem "expressive" as well as understanding the relation and potential differences between measured and perceived performance features. For instance, expressiveness is style-dependent, meaning that the perceived *appropriate* level of expression in a Baroque piece will be different from that of a Romantic piece — something that has been referred to as "stylishness" [25, 45]. In addition, there is the perceived *amount* of expressiveness, which is considered independent of stylishness [88]. Finally, Schubert and Fabian distinguish a third "layer" of expressiveness which arises from a performer's manipulation of various features specifically to alter or enhance emotion [89]. This is distinct from *musical* expressiveness which more generally refers to the manipulation of compositional elements by the performer in order to be "expressive" without necessarily needing to express a specific emotion. Practically speaking, however, it may be difficult for listeners to separate these varieties of expressiveness [89, p.293]), and research has demonstrated that there are interactions between them (e.g., [107]).

4.2 Expressive variation

Several scholars have made significant advances in our understanding of the role of timing, tempo, and dynamics on listeners' perception of music. As noted in Sect. 2, the subtle variations in tempo and dynamics executed by a performer have been shown to play a large role in highlighting and segmenting musical structure. For instance, changes in timing and articulation within small structural units such as the measure, beat, or sub-beat appear to aid in the communication and emphasis of the metrical structure (e.g., [4, 29, 70, 93]), whereas changes across larger segments such as phrases, aid in the communication of formal structure. In fact, the communication of musical structure has been suggested as one of the primary roles or functions of a successful performer (see [43, 83]). For instance, an experiment by Sloboda found that listeners were better able to identify the meter of an ambiguous passage when performed by a more experienced performer [93]. Through measuring the differences in the performers' expressive

variations, Sloboda identified dynamics and articulation — in particular, a *tenuto* articulation— as the most important features for communicating which notes were accented.

The extent to which a performer's expressive variations align with a listener's musical expectations appears an important consideration. For example, because of the predictable relation between timing and structural segmentation, it has been demonstrated that listeners find it difficult to detect timing (and duration) deviations from a "metronomic" performance when the pattern and placement of those deviations are stylistically typical [66, 77, 78]. Likewise, Clarke [11] found pianists able to more accurately reproduce a performance when the timing profile was "normative" with regards to the musical structure, and also found listeners' aesthetic judgments to be highest for those performances with the original timing profiles compared with those that were inverted or altered.

In addition to communicating structural information to the listener, the role of performance features such as timing and dynamics have also been studied extensively for their role in shaping "expressive" performance (see [12, 30]). For instance, a factor analysis in [89] examined the features and qualities that may be related to perceived expressiveness, finding that dynamics had the highest impact on the factor labeled "emotional expressiveness." Gingras et al. studied the relation between musical structure, expressive variation, and listeners' ratings of musical tension. They found that variations in expressive timing were most predictive of listeners' tension ratings [31]. While the role of expressive variation in timbre and intonation have generally been less studied, there has been substantial attention given to the expressive qualities of the singing voice where these parameters are especially relevant (see [96]). For instance, Sundberg found that a sharpened intonation at a phrase climax contributed to increased expressiveness and perceived excitement [97], and Siegwart and Scherer found that listener preferences were correlated with certain spectral components such as the relative strength of the fundamental and higher spectral centroid [92].

The reason why expressive variation is so enjoyable for listeners remains largely an open research question. As mentioned above, its role appears to go beyond bolstering the communication of musical structure. As pointed out by Repp, even a computerized or metronomic performance will contain grouping cues [83]. However, one prominent theory suggests that systematic performance deviations (such as tempo) may generate aesthetically pleasing expressive performances in part due to their exhibiting characteristics that mimic "natural motion" in the physical world [32, 49, 79, 102, 103] or human movements or gestures [8, 66]. For instance, Friberg and Sundberg suggested that the shape of final ritardandi matched the the velocity of runners coming to a stop [27], and Juslin includes "motion principles" in his model of performance expression [43].

4.3 Mapping and Predicting Listener Judgments

In order to isolate listeners' perception of features that are strictly performance-related, several scholars have investigated listeners' judgments across multiple performances of the same excerpt of music (e.g., [24, 77]). A less-common technique relies on synthesized constructions or manipulations of performances, typically using some kind of rule-based system to manipulate certain musical parameters (e.g., [11, 76, 83, 95]), and frequently making use of continuous data collection measures (e.g., [89]).

From these studies, it appears that listeners (especially "trained" listeners) are capable not only of identifying performance characteristics such as phrasing, articulation, and vibrato, but that they are frequently able to identify them in a manner that is aligned with the performer's intentions (e.g., [26, 63]). However, while listeners may be able to identify performers' intentions, they may not have the perceptual acuity to identify certain features with the same precision allowed by acoustic measures. For instance, a study by Howes [39] showed there was no correlation between measured and perceived vibrato onset times. This suggests that there are some measurable performance parameters that may not map well to human perception. For example, an objectively measurable difference between a "deadpan" and "expressive" performance does not necessarily translate to a *perceived* "expressive" performance, especially if the changes in measured performance parameters are structurally normative, as discussed in Sect. 4.2.

Given a weak relation between a measured parameter and listeners' perception of that parameter, an important question arises: is the parameter itself not useful in modeling human perception, or is the metric simply inappropriate? For example, there are many aspects of music perception that are known to be categorical (e.g., pitch) in which case a continuous metric would not work well in a model designed to predict human ratings. Similarly, there is the consideration of the role of the *representation* and transformation of a measured parameter for predicting perceptual ratings. This question was raised by Timmers, who examined the representation of tempo and dynamics that best predicted listener judgments of musical similarity [101]. This study found that, while most existing models rely on normalized variations of tempo and dynamics, the absolute tempo and the interaction of tempo and loudness were better predictors.

Clearly, the execution of several performance parameters are important for the perception of both fine-grained and large-grained musical structures, and appear to have a large influence over listeners' perception and experience of the emotional and expressive aspects of a performance. Since the latter appears to carry great significance for both MPA and music perception research, it suggests that future work ought to focus on disentangling the relative weighting of the various features controlled by performers that contribute to an expressive performance. Since it is frequently alluded to that a performer's manipulation of musical tension is one of the strongest contributors to an expressive performance, further empirical research must attempt to break down this high-level feature into meaningful collections of well-defined features that would be useful for MPA.

The research surveyed in this section highlights the importance of human perception in MPA research, especially

as it pertains to the communication of emotion, musical structure, and creating an aesthetically pleasing performance. In fact, the successful modeling of perceptually-relevant performance attributes, such as those that mark "expressiveness," could have a large impact not only for MPA but for many other areas of MIR research, such as computer-generated composition and performance, automatic accompaniment, virtual instrument design and control, or robotic instruments and HCI (see, e.g., the range of topics discussed in [46]). A major obstacle impeding research in this area is the inability to successfully isolate (and therefore understand) the various performance characteristics that contribute to a so-called "expressive" performance from a listener's perspective. Existing literature reviews on the topic of MPA have not been able to shed much light on this problem, in part because researchers frequently disagree (or conflate) the various definitions of "expressive." Careful experimental design and/or meta-analyses across both MPA and cognition research, as well as cross-collaboration between MIR and music cognition researchers, may therefore prove fruitful areas for future research.

5. PERFORMANCE ASSESSMENT

Assessment-focused MPA deals with modeling how we as humans assess a musical performance. While this is technically a subset of listener-focused MPA, its importance to MPA research and music education warrants a tailored review of research in this area. Performance assessment is a critical and ubiquitous aspect of music pedagogy: students rely on regular feedback from teachers to learn and improve skills, recitals are used to monitor progress, and selection into ensembles is managed through competitive auditions. The performance parameters on which these assessments are based are not only subjective but also ill-defined, leading to large differences in subjective opinion among music educators [100, 108]. Work within Assessment-focused MPA seeks to increase the objectivity of performance assessments [62], and build accessible and reliable tools for automatic assessment.

Over the last decade, several researchers have worked towards developing tools capable of automatic music performance assessment which can be categorized based on: (i) the parameters of the performance that are assessed, and (ii) the technique/method used to design these systems.

Tools for performance assessment typically assess one or more performance parameters which are usually related to the accuracy of the performance in terms of pitch and timing [56, 72, 106, 113], or quality of sound (timbre) [47, 74]. In building an assessment tool, the choice of parameters may depend on the proficiency level of the performer being assessed. For example, beginners will benefit more from feedback in terms of low-level parameters such as pitch or rhythmic accuracy as opposed to feedback on higher-level parameters such as articulation or expression.

Assessment tools can also vary based on the granularity of assessments. Tools may simply classify a performance as 'good' or 'bad' [47, 64], or grade it on a scale, say from 1 to 10 [72]. Systems may provide fine-grained note-by-note assessments [74] or analyze entire performances and report a single assessment score [64, 72].

While different methods have been used to create performance assessment tools, the common approach has been to use descriptive features extracted from the audio recording of a performance based on which a cognitive model predicts the assessment. This approach requires availability of performance data (recordings) along-with human (expert) assessments for the rated parameters.

The level of sophistication of cognitive models was limited especially for early attempts; e.g., simple classifiers such as Support Vector Machines were used to predict human ratings. In this case, descriptive features became an important aspect of the system design. In some approaches, standard spectral and temporal features such as Spectral Centroid, Spectral Flux, and Zero-Crossing Rate were used [47]. In others, features aimed at capturing cognitive aspects of music perception were hand-designed using either musical intuition or expert knowledge [2, 52, 64, 74]. For instance, Nakano et al. used features measuring pitch stability and vibrato as inputs to a simple classifier to rate the quality of vocal performances [64]. Several studies also attempted to combine low-level audio features with hand-designed feature sets [56, 106, 113], as well as incorporating information from the musical score into feature computation [6, 7, 18, 106].

Recent methods, however, have transitioned towards using advanced ML techniques such as Sparse Coding [36, 114, 116] and Deep Learning [72] as a proxy to sophisticated cognitive models. Contrary to earlier methods which focused on extracting cognitively intuitive or important features, these techniques input raw data (usually in the form of pitch contours or spectrograms) and train the models to automatically learn meaningful features so as to accurately predict the assessment ratings.

In some ways, this evolution in methodology has mirrored that of other MIR tasks: there has been a gradual transition from feature design to feature learning. Feature design and feature learning have an inherent trade-off. Learned features extract relevant information from data which might not be represented in the hand-crafted feature set. This is evident from their superior performance at assessment modeling tasks [72, 114]. However, this superior performance comes at the cost of low interpretability. Learned features tend to be abstract and cannot be easily understood. Custom-designed features, on the other hand, typically either measure a simple low-level characteristic of the audio signal or link to high-level semantic concepts such as pitch or rhythm which are intuitively interpretable. Thus, such models allow analysis that can aid in the interpretation of semantic concepts for music performance assessment. For instance, Gururani et al. analyzed the impact of different features on an assessment prediction task and found that features measuring tempo variations were particularly critical, and that score-aligned features performed better than score-independent features [35].

In spite of several attempts across varied performance parameters using different methods, the important features

for assessing music performances remain unclear. This is evident from the average performance of these tools in modeling human judgments. Most of the presented systems either work well only for very select data [47] or have comparably low prediction accuracies [106, 113], rendering them unusable in most practical scenarios. While this may be partially attributed to the subjective nature of the task itself, there are several other factors which have limited the improvement of these tools. First, most of the models are trained on small task-specific or instrument-specific datasets that might not reflect noisy real-world data. This reduces the generalizability of these models. The problem becomes more serious for data-hungry methods such as Deep Learning which require large amounts of data for training. Second, the distribution of ground-truth (expert) ratings given by human judges is in many datasets skewed towards a particular class or value [35]. This makes it challenging to train unbiased models. Finally, the number of parameters required to adequately model a performance results in high dimensional data. While the typical approach is to train different models for different parameters, this approach necessitates availability of performance data along-with expert assessments for all these parameters. In many occasions, such assessments are either not available or are costly to obtain.

Given these data-related challenges, an interesting direction for future research might consider leveraging models which are successful at assessing a few parameters (and/or instruments) to improve the performance of models for other parameters (and/or instruments). This approach, usually referred to as transfer learning, has been found to be successful in other MIR tasks [9]. In addition, the ability to interpret and understand the features learned by end-to-end models will play an important role in improving assessment tools. Interpretability of neural networks is still an active area of research in MIR, and performance assessment is an excellent test-bed for developing such methods.

6. CONCLUSION

The previous sections outlined insights gained by MPA at the intersection of audio content analysis, empirical musicology, and music perception research. These insights are of importance for better understanding the process of making music as well as affective user reactions to music. Furthermore, they enable a considerable range of applications spanning a multitude of different areas including systematic musicology, music education, MIR, and computational creativity, leading to a new generation of music discovery and recommendation systems, and generative music systems. The most obvious application example connecting MPA and MIR is music tutoring software. Such software aims at supplementing teachers by providing students with insights and interactive feedback by analyzing and assessing the audio of practice sessions. The ultimate goals of an interactive music tutor are to highlight problematic parts of the students' performance, provide a concise yet easily understandable analysis, give specific and understandable feedback on how to improve, and individualize the curriculum depending on the students' mistakes and general progress. Various (commercial) solutions are already available, exhibiting a similar set of goals. These systems adopt different approaches, ranging from traditional music classroom settings to games targeting a playful learning experience. Examples for tutoring applications are SmartMusic [59], Yousician [117], Music Prodigy [99], and SingStar [94].

Performance parameters have a long history being either explicitly or implicitly part of MIR systems. For instance, core MIR tasks such as music genre classification and music recommendation systems have been utilizing tempo and dynamics features successfully [28]. Generative models often require performance data to allow for the rendition of a convincing output. This obviously includes performance rendition systems that take a score and attempt to render a human-like output [60, 67], but it is also important for models of improvisation such as jazz solos as pitch information is part of the performance.

Despite such practical applications, there are still many open topics and challenges that need to be addressed. The main challenges of MPA have been summarized at the end of the sections above. The related challenges to the MIR community, however, are multi-faceted as well. First, the fact that the majority of the presented studies use manual annotations instead of automated methods should encourage the MIR community to re-evaluate the measures of success of their proposed systems if, as it appears to be, the outputs lack the robustness or accuracy required for a detailed analysis even for tasks considered to be 'solved.' Second, the missing separation of score and performance parameters when framing research questions or problem definitions can impact not only interpretability and reusability of insights but might also reduce algorithm performance. If, e.g., a music emotion recognition system does not differentiate between the impact of core musical ideas and performance characteristics, it will have a harder time differentiating relevant and irrelevant information. Thus, it is essential for MIR systems to not only differentiate between score and performance parameters in the system design phase but also analyze their respective contributions during evaluation. Third, lack of data continues to be a challenge for both, MIR core tasks and MPA; a focus on approaches for limited data [61], weakly labeled data, and unlabeled data [115] could help address this problem..

In conclusion, the fields of MIR and MPA each depend on the advances in the other field. In addition, music perception and cognition, while not a traditional topic within MIR, can be seen as an important linchpin for the advancement of MIR systems that depend on reliable and diverse perceptual data. Cross-disciplinary approaches to MPA bridging methodologies and data from music cognition and MIR are likely to be most influential for future research. Empirical, descriptive research driven by advanced audio analysis is necessary to extend our understanding of music and its perception, which in turn will allow us to create better systems for music analysis, music understanding, and music creation.

7. REFERENCES

[1] Jakob Abeßer, Klaus Frieler, Estefanía Cano, Martin Pfleiderer, and Wolf-Georg Zaddach. Score-Informed Analysis of Tuning, Intonation, Pitch Modulation, and Dynamics in Jazz Solos. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 25(1):168–177, January 2017.

[2] Jakob Abeßer, Johannes Hasselhorn, Christian Dittmar, Andreas Lehmann, and Sascha Grollmisch. Automatic Quality Assessment of Vocal and Instrumental Performances of Ninth-Grade and Tenth-Grade Pupils. In *International Symposium on Computer Music Multidisciplinary Research (CMMR)*, 2013.

[3] Helena Bantula, Sergio Iván Giraldo, and Rafael Ramirez. Jazz Ensemble Expressive Performance Modeling. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, New York, 2016.

[4] Klaus-Ernst Behne and Burkhard Wetekam. Musikpsychologische Interpretationsforschung: Individualitat und Intention. *Musikpsychologie. Jahrbuch der Deutschen Gesellschaft für Musikpsychologie*, 10:24–37, 1993.

[5] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, Dec 2013.

[6] Jordi Bonada, Alex Loscos, and Oscar Mayor. Performance Analysis and Scoring of the Singing Voice. In *Audio Engineering Society Conference: 35th International Conference: Audio for Games*. Audio Engineering Society, 2009.

[7] Baris Bozkurt, Ozan Baysal, and Deniz Yüret. A Dataset and Baseline System for Singing Voice Assessment. In *International Symposium on Computer Music Multidisciplinary Research (CMMR)*, pages 430–438, 2017.

[8] George John Broze III. *Animacy, anthropomimesis, and musical line*. PhD Thesis, The Ohio State University, 2013.

[9] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Transfer Learning for Music Classification and Regression Tasks. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 141–149, Suzhou, 2017.

[10] Ching-Hua Chuan and Elaine Chew. A Dynamic Programming Approach to the Extraction of Phrase Boundaries from Tempo Variations in Expressive Performances. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Vienna, 2007.

[11] Eric F Clarke. Imitating and Evaluating Real and Transformed Musical Performances. *Music Perception*, 10:317–341, 1993.

[12] Eric F Clarke. Rhythm and Timing in Music. In *The Psychology of Music*, pages 473–500. Academic Press, San Diego, 2nd edition, 1998.

[13] Eric F Clarke. Listening to Performance. In John Rink, editor, *Musical Performance — A Guide to Understanding*. Cambridge University Press, Cambridge, 2002.

[14] Eric F Clarke. Understanding the Psychology of Performance. In John Rink, editor, *Musical Performance – A Guide to Understanding*. Cambridge University Press, Cambridge, 2002.

[15] Martin Clayton. *Time in Indian Music: Rhythm, Metre, and Form in North Indian Rag Performance*. Oxford University Press, August 2008. Google-Books-ID: pGOTwtQzxBYC.

[16] Simone Dalla Bella and Caroline Palmer. Tempo and Dynamics in Piano Performance: The Role of Movement Amplitude. In *Proc. of the 8th International Conference on Music Perception & Cognition (ICMPC)*, Evanston, August 2004.

[17] Johanna Devaney, Michael I. Mandel, Daniel P.W. Ellis, and Ichiro Fujinaga. Automatically Extracting Performance Data from Recordings of Trained Singers. *Psychomusicology: Music, Mind and Brain*, 21(1-2):108–136, 2011.

[18] Johanna Devaney, Michael I Mandel, and Ichiro Fujinaga. A Study of Intonation in Three-Part Singing using the Automatic Music Performance Analysis and Comparison Toolkit (AMPACT). In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Porto, 2012.

[19] Roberto Dillon. Extracting Audio Cues in Real Time to Understand Musical Expressiveness. In *Proc. of the MOSART workshop*, Barcelona, November 2001.

[20] Roberto Dillon. A Statistical Approach to Expressive Intention Recognition in Violin Performances. In *Proc. of the Stockholm Music Acoustics Conference (SMAC)*, Stockholm, August 2003.

[21] Roberto Dillon. *On the Recognition of Expressive Intentions in Music Playing: A Computational Approach with Experiments and Applications*. PhD Thesis, University of Genoa, Faculty of Engineering, Genoa, 2004.

[22] Tomislav Dimov. *Short Historical Overview and Comparison of the Pitch Width and Speed Rates of the Vibrato Used in Sonatas and Partitas for Solo Violin by Johann Sebastian Bach as Found in Recordings of Famous Violinists of the Twentieth and the Twenty-First Centuries*. D.M.A., West Virginia University, West Virginia, 2010.

[23] Simon Dixon, Werner Goebl, and Gerhard Widmer. The Performance Worm: Real Time Visualisation of Expression based on Langner's Tempo Loudness Animation. In *Proc. of the International Computer Music Conference (ICMC)*, Göteborg, September 2002.

[24] Dorottya Fabian and Emery Schubert. Musical Character and the Performance and Perception of Dotting, Articulation and Tempo in 34 Recordings of Variation 7 from J.S. Bach's Goldberg Variations (BWV 988). *Musicae Scientiae*, 12(2):177–206, July 2008.

[25] Dorottya Fabian and Emery Schubert. Baroque Expressiveness and Stylishness in Three Recordings of the D minor Sarabanda for Solo Violin (BWV 1004) by JS Bach. *Music Performance Research*, 3:36–55, 2009.

[26] Dorottya Fabian, Emery Schubert, and Richard Pulley. A Baroque Träumerei: The Performance and Perception of two Violin Renditions. *Musicology Australia*, 32(1):27–44, July 2010.

[27] Anders Friberg and Johan Sundberg. Does Music Performance Allude to Locomotion? A Model of Final Ritardandi Derived from Measurements of Stopping Runners. *Journal of the Acoustical Society of America*, 105(3):1469–1484, 1999.

[28] Zhouyu Fu, Guojun Lu, Kai Ming Ting, and Dengsheng Zhang. A Survey of Audio-Based Music Classification and Annotation. *Trans. Multimedia*, 13(2):303–319, April 2011.

[29] Alf Gabrielsson. Once Again: The Theme from Mozart's Piano Sonata in A Major (K. 331): A Comparison of Five Performances. In Alf Gabrielsson, editor, *Action and Perception in Rhythm and Music*, pages 81–103. Royal Swedish Academy of Music, No. 55., Stockholm, 1987.

[30] Alf Gabrielsson. The Performance of Music. In Diana Deutsch, editor, *Psychology of Music*. Academic Press, San Diego, 2nd edition, 1998.

[31] Bruno Gingras, Marcus T Pearce, Meghan Goodchild, Roger T Dean, Geraint Wiggins, and Stephen McAdams. Linking Melodic Expectation to Expressive Performance Timing and Perceived Musical Tension. *Journal of Experimental Psychology: Human Perception and Performance*, 42(4):594, 2016.

[32] Robert O Gjerdingen. Shape and Motion in the Microstructure of Song. *Music Perception*, 6:35–64, 1988.

[33] Rong Gong. *Automatic Assessment of Singing Voice Pronunciation: A Case Study with Jingju Music*. PhD Thesis, Universitat Pompeu Fabra, Barcelona, 2018.

[34] Chitralekha Gupta and Preeti Rao. Objective Assessment of Ornamentation in Indian Classical Singing. In Sølvi Ystad, Mitsuko Aramaki, Richard Kronland-Martinet, Kristoffer Jensen, and Sanghamitra Mohanty, editors, *Speech, Sound and Music Processing: Embracing Research in India*, Lecture Notes in Computer Science, pages 1–25. Springer Berlin Heidelberg, 2012.

[35] Siddharth Gururani, K Ashis Pati, Chih-Wei Wu, and Alexander Lerch. Analysis of Objective Descriptors for Music Performance Assessment. In *Proc. of the International Conference on Music Perception and Cognition (ICMPC)*, Toronto, Ontario, Canada, 2018.

[36] Yoonchang Han and Kyogu Lee. Hierarchical Approach to Detect Common Mistakes of Beginner Flute Players. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 77–82, Taipei, 2014.

[37] Artur Hartmann. Untersuchungen über das metrische Verhalten in musikalischen Interpretationsvarianten. *Archiv für die gesamte Psychologie*, 84:103–192, 1932.

[38] Peter Hill. From Score to Sound. In John Rink, editor, *Musical Performance -– A Guide to Understanding*. Cambridge University Press, Cambridge, 2002.

[39] Patricia Howes, Jean Callaghan, Pamela Davis, Dianna Kenny, and William Thorpe. The Relationship between Measured Vibrato Characteristics and Perception in Western Operatic Singing. *Journal of Voice*, 18(2):216–230, June 2004.

[40] David Huron. Tone and voice: A derivation of the rules of voice-leading from perceptual principles. *Music Perception: An Interdisciplinary Journal*, 19(1):1–64, 2001.

[41] Luis Jure, Ernesto Lopez, Martin Rocamora, Pablo Cancela, Haldo Sponton, and Ignacio Irigaray. Pitch Content Visualization Tools for Music Performance Analysis. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Porto, 2012.

[42] Patrik N Juslin. Cue Utilization in Communication of Emotion in Music Performance: Relating Performance to Perception. *Journal of Experimental Psychology*, 26(6):1797–1813, 2000.

[43] Patrik N Juslin. Five Facets of Musical Expression: A Psychologist's Perspective on Music Performance. *Psychology of Music*, 31(3):273–302, July 2003.

[44] Patrik N Juslin and Petri Laukka. Communication of Emotions in Vocal Expression and Music Performance: Different Channels, Same Code? *Psychological Bulletin*, 129(5):770–814, September 2003.

[45] Roger A Kendall and Edward C Carterette. The Communication of Musical Expression. *Music Perception*, 8(2):129–164, 1990.

[46] Alexis Kirke and Eduardo R. Miranda, editors. *Guide to Computing for Expressive Music Performance*. Springer Science & Business Media, August 2012. Google-Books-ID: d9petWBuqHEC.

[47] Trevor Knight, Finn Upham, and Ichiro Fujinaga. The Potential for Automatic Assessment of Trumpet Tone Quality. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Miami, 2011.

[48] Jörg Langner and Werner Goebl. Representing Expressive Performance in Tempo-Loudness Space. In *Proc. of the Conference of the European Society for the Cognitive Sciences of Music (ESCOM)*, Liege, April 2002.

[49] Leon van Noorden and Dirk Moelants. Resonance in the Perception of Musical Pulse. *Journal of New Music Research*, 28(1):43–66, 1999.

[50] Alexander Lerch. *Software-Based Extraction of Objective Parameters from Music Performances*. GRIN Verlag, München, 2009.

[51] Alexander Lerch. *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. Wiley-IEEE Press, Hoboken, 2012.

[52] Pei-Ching Li, Li Su, Yi-Hsuan Yang, Alvin WY Su, and others. Analysis of Expressive Musical Terms in Violin Using Score-Informed and Expression-Based Audio Features. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 809–815, Malaga, 2015.

[53] Cynthia CS Liem and Alan Hanjalic. Expressive Timing from Cross-Performance and Audio-based Alignment Patterns: An Extended Case Study. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Miami, 2011.

[54] Cynthia CS Liem and Alan Hanjalic. Comparative Analysis of Orchestral Performance Recordings: An Image-based Approach. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Malaga, 2015.

[55] Paul Luizard, Erik Brauer, and Stefan Weinzierl. Singing in Physical and Virtual Environments: How Padapt to Room Acoustical Conditions. In *Proc. of the AES Conference on Immersive and Interactive Audio*, page 8, York, 2019. AES.

[56] Yin-Jyun Luo, Li Su, Yi-Hsuan Yang, and Tai-Shih Chi. Detection of Common Mistakes in Novice Violin Playing. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Malaga, 2015.

[57] Rebecca Bowman MacLeod. *Influences of Dynamic Level and Pitch Height on the Vibrato Rates and Widths of Violin and Viola Players*. PhD Thesis, Florida State University, Tallahassee, Florida, 2006.

[58] Hans-Joachim Maempel. Musikaufnahmen als Datenquellen der Interpretationsanalyse. In Heinz von Lösch and Stefan Weinzierl, editors, *Gemessene Interpretation — Computergestützte Aufführungsanalyse im Kreuzverhör der Disziplinen*, Klang und Begriff, pages 157–171. Schott, Mainz, 2011.

[59] MakeMusic, Inc. SmartMusic, April 2019. https://www.smartmusic.com, last accessed 04/11/2019.

[60] Iman Malik and Carl Henrik Ek. Neural Translation of Musical Style. In *Proc. of the NeurIPS Workshop on Machine Learning for Creativity and Design*, Long Beach, California, USA, 2017.

[61] Brian McFee, Eric J Humphrey, and Juan Pablo Bello. A Software Framework for Musical Data Augmentation. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Malaga, 2015.

[62] Gary E McPherson and William F Thompson. Assessing Music Performance: Issues and Influences. *Research Studies in Music Education*, 10(1):12–24, June 1998.

[63] Toshie Nakamura. The Communication of Dynamics between Musicians and Listeners through Musical Performance. *Perception & Psychophysics*, 41(6):525–533, 1987.

[64] Tomoyasu Nakano, Masataka Goto, and Yuzuru Hiraga. An Automatic Singing Skill Evaluation Method for Unknown Melodies using Pitch Interval Accuracy and Vibrato Features. In *Proc. of the International Conference on Spoken Language Processing (ICSLP)*, pages 1706–1709, 2006.

[65] Krish Narang and Preeti Rao. Acoustic Features for Determining Goodness of Tabla Strokes. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 257–263, Suzhou, China, 2017.

[66] Mitchell S Ohriner. Grouping Hierarchy and Trajectories of Pacing in Performances of Chopin's Mazurkas. *Music Theory Online*, 18(1), 2012.

[67] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. This Time with Feeling: Learning Expressive Musical Performance. *Neural Computing and Applications*, pages 1–13, 2018.

[68] Eitan Ornoy and Shai Cohen. Analysis of Contemporary Violin Recordings of 19th Century Repertoire: Identifying Trends and Impacts. *Frontiers in Psychology*, 9:2233, 2018.

[69] Caroline Palmer. Mapping Musical Thought to Musical Performance. *Journal of Experimental Psychology: Human Perception and Performance*, 15(2):331–346, 1989.

[70] Caroline Palmer. *Timing in skilled music performance*. PhD Thesis, Cornell University, Ithaca, NY, 1989.

[71] Caroline Palmer. Music Performance. *Annual Review of Psychology*, 48:115–138, 1997.

[72] K Ashis Pati, Siddharth Gururani, and Alexander Lerch. Assessment of Student Music Performances Using Deep Neural Networks. *Applied Sciences*, 8(4):507, March 2018.

[73] Jeroen Peperkamp, Klaus Hildebrandt, and Cynthia CS Liem. A Formalization of Relative Local Tempo Variations in Collections of Performances. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, 2017.

[74] O Romani Picas, H Parra Rodriguez, Dara Dabiri, Hiroshi Tokuda, Wataru Hariya, Koji Oishi, and Xavier Serra. A Real-time System for Measuring Sound Goodness in Instrumental Sounds. In *Proc. of the Audio Engineering Society Convention*, volume 138, Warsaw, 2015.

[75] Dirk-Jan Povel. Temporal Structure of Performed Music: Some Preliminary Observations. *Acta Psychologica*, 41(4):309–320, 1977.

[76] Bruno H Repp. Expressive Microstructure in Music: A Preliminary Perceptual Assessment of Four Composers'" Pulses". *Music Perception*, 6(3):243–273, 1989.

[77] Bruno H Repp. Patterns of Expressive Timing in Performances of a Beethoven Minuet by Nineteen Famous Pianists. *Journal of the Acoustical Society of America (JASA)*, 88(2):622–641, August 1990.

[78] Bruno H Repp. A Constraint on the Expressive Timing of a Melodic Gesture: Evidence from Performance and Aesthetic Judgment. *Music Perception*, 10:221–243, 1992.

[79] Bruno H Repp. Music as Motion: A Synopsis of Alexander Truslit's (1938) Gestaltung und Bewegung in der Musik. *Psychology of Music*, 21:48–72, 1993.

[80] Bruno H Repp. The Art of Inaccuracy: Why Pianists' Errors are Difficult to Hear. *Music Perception*, 14(2):161–184, 1996.

[81] Bruno H Repp. The Dynamics of Expressive Piano Performance: Schumann's 'Träumerei' Revisited. *Journal of the Acoustical Society of America (JASA)*, 100(1):641–650, 1996.

[82] Bruno H Repp. A Microcosm of Musical Expression. I. Quantitative Analysis of Pianists' Timing in the Initial Measures of Chopin's Etude in E major. *Journal of the Acoustical Society of America (JASA)*, 104(2):1085–1100, 1998.

[83] Bruno H Repp. Obligatory "Expectations" of Expressive Timing Induced by Perception of Musical Structure. *Psychological Research*, 61(1):33–43, March 1998.

[84] Craig S Sapp. Comparative Analysis of Multiple Musical Performances. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Vienna, September 2007.

[85] Craig S Sapp. Hybrid Numeric/Rank Similarity Metrics for Musical Performance Analysis. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Philadelphia, September 2008.

[86] Zora Schärer Kalkandjiev and Stefan Weinzierl. The Influence of Room Acoustics on Solo Music Performance: An Empirical Case Study. *Acta Acustica united with Acustica*, 99(3):433–441, May 2013.

[87] Zora Schärer Kalkandjiev and Stefan Weinzierl. The Influence of Room Acoustics on Solo Music Performance. An Experimental Study. *Psychomusicology: Music, Mind, and Brain*, 25(3):195–207, 2015.

[88] Emery Schubert and Dorottya Fabian. The Dimensions of Baroque Music Performance: A Semantic Differential Study. *Psychology of Music*, 34(4):573–587, 2006.

[89] Emery Schubert and Dorottya Fabian. A Taxonomy of Listeners' Judgments of Expressiveness in Music Performance. In Dorottya Fabian, Renee Timmers, and Emery Schubert, editors, *Expressiveness in Music Performance: Empirical approaches across styles and cultures*. Oxford University Press, July 2014.

[90] Carl E Seashore. *Psychology of Music*. McGraw-Hill, New York, 1938.

[91] L Henry Shaffer. Timing in Solo and Duet Piano Performances. *The Quarterly Journal of Experimental Psychology*, 36A:577–595, 1984.

[92] Hervine Siegwart and Klaus R Scherer. Acoustic Concomitants of Emotional Expression in Operatic Singing: The Case of Lucia in Ardi gli incensi. *Journal of Voice*, 9(3):249–260, 1995.

[93] John A Sloboda. The Communication of Musical Metre in Piano Performance. *The Quarterly Journal of Experimental Psychology Section A*, 35(2):377–396, May 1983.

[94] Sony Interactive Entertainment Europe. SingStar, April 2019. http://www.singstar.com, last accessed 04/11/2019.

[95] Johan Sundberg. How can Music be Expressive? *Speech Communication*, 13(1):239–253, October 1993.

[96] Johan Sundberg. The Singing Voice. In Sascha Frühholz and Pascal Belin, editors, *The Oxford Handbook of Voice Perception*. Oxford University Press, December 2018.

[97] Johan Sundberg, Filipa MB Lã, and Evangelos Himonides. Intonation and Expressivity: A Single Case Study of Classical Western Singing. *Journal of Voice*, 27(3):391–e1, 2013.

[98] Haruto Takeda, Takuya Nishimoto, and Shigeki Sagayama. Rhythm and Tempo Recognition of Music Performance from a Probabilistic Approach. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Barcelona, 2004.

[99] The Way of H, Inc. (dba Music Prodigy). Music Prodigy, April 2019. http://www.musicprodigy.com, last accessed 04/11/2019.

[100] Sam Thompson and Aaron Williamon. Evaluating Evaluation: Musical Performance Assessment as a Research Tool. *Music Perception: An Interdisciplinary Journal*, 21(1):21–41, 2003.

[101] Renee Timmers. Predicting the Similarity between Expressive Performances of Music from Measurements of Tempo and Dynamics. *Journal of the Acoustical Society of America (JASA)*, 117(1), 2005.

[102] Neil P M Todd. The Dynamics of Dynamics: A Model of Musical Expression. *Journal of the Acoustical Society of America*, 91:3540–3550, 1992.

[103] Neil P M Todd. The Kinematics of Musical Expression. *Journal of the Acoustical Society of America*, 97:1940–1949, 1995.

[104] Ken'ichi Toyoda, Kenzi Noike, and Haruhiro Katayose. Utility System for Constructing Database of Performance Deviations. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Barcelona, 2004.

[105] Sam Van Herwaarden, Maarten Grachten, and W Bas De Haas. Predicting Expressive Dynamics in Piano Performances Using Neural Networks. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, 2014.

[106] Amruta Vidwans, Siddharth Gururani, Chih-Wei Wu, Vinod Subramanian, Rupak Vignesh Swaminathan, and Alexander Lerch. Objective Descriptors for the Assessment of Student Music Performances. In *Proc. of the AES Conference on Semantic Audio*, Erlangen, 2017. Audio Engineering Society (AES).

[107] Sandrine Vieillard, Mathieu Roy, and Isabelle Peretz. Expressiveness in Musical Emotions. *Psychological Research*, 76(5):641–653, September 2012.

[108] Brian C Wesolowski, Stefanie A Wind, and George Engelhard. Examining Rater Precision in Music Performance Assessment: An Analysis of Rating Scale Structure using the Multifaceted Rasch Partial Credit Model. *Music Perception: An Interdisciplinary Journal*, 33(5):662–678, 2016.

[109] Gerhard Widmer. Applications of Machine Learning to Music Research: Empirical Investigations into the Phenomenon of Musical Expression. In *Machine Learning, Data Mining and Knowledge Discovery: Methods and Applications*. Wiley & Sons, 1997.

[110] Gerhard Widmer. Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artificial Intelligence*, 146(2):129–148, June 2003.

[111] Gerhard Widmer and Werner Goebl. Computational Models of Expressive Music Performance: The State of the Art. *Journal of New Music Research*, 33(3):203–216, September 2004.

[112] Gerhard Widmer and Patrick Zanon. Automatic Recognition of Famous Artists by Machine. In *Proc. of the 16th European Conference on Artificial Intelligence (ECAI)*, Valencia, August 2004.

[113] Chih-Wei Wu, Siddharth Gururani, Christopher Laguna, K Ashis Pati, Amruta Vidwans, and Alexander Lerch. Towards the Objective Assessment of Music Performances. In *Proc. of the International Conference on Music Perception and Cognition (ICMPC)*, pages 99–103, San Francisco, 2016.

[114] Chih-Wei Wu and Alexander Lerch. Assessment of Percussive Music Performances with Feature Learning. *International Journal of Semantic Computing*, 12(3):315–333, 2018.

[115] Chih-Wei Wu and Alexander Lerch. From Labeled to Unlabeled Data – On the Data Challenge in Automatic Drum Transcription. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, 2018.

[116] Chih-Wei Wu and Alexander Lerch. Learned Features for the Assessment of Percussive Music Performances. In *Proc. of the International Conference on Semantic Computing (ICSC)*, Laguna Hills, 2018. IEEE.

[117] Yousician Oy. Yousician, April 2019. https://www.yousician.com, last accessed 04/11/2019.

[118] Shuo Zhang, Rafael Caro Repetto, and Xavier Serra. Understanding the Expressive Functions of Jingju Metrical Patterns through Lyrics Text Mining. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017.

INTELLIGENT USER INTERFACES FOR MUSIC DISCOVERY: THE PAST 20 YEARS AND WHAT'S TO COME

Peter Knees [1] **Markus Schedl** [2] **Masataka Goto** [3]
[1] Faculty of Informatics, TU Wien, Vienna, Austria
[2] Institute of Computational Perception, Johannes Kepler University Linz, Austria
[3] National Institute of Advanced Industrial Science and Technology (AIST), Japan
peter.knees@tuwien.ac.at, markus.schedl@jku.at, m.goto@aist.go.jp

ABSTRACT

Providing means to assist the user in finding music is one of the original motivations underlying the research field known as Music Information Retrieval (MIR). Therefore, already the first edition of ISMIR in the year 2000 called for papers addressing the topic of "User interfaces for music IR". Since then, the way humans interact with technology to access and listen to music has substantially changed, not least driven by the advances of MIR and related research fields such as machine learning and recommender systems.

In this paper, we reflect on the evolution of MIR-driven user interfaces for music browsing and discovery over the past two decades. We argue that three major developments have transformed and shaped user interfaces during this period, each connected to a phase of new listening practices: first, connected to personal music collections, intelligent audio processing and content description algorithms that facilitate the automatic organization of repositories and finding music according to sound qualities; second, connected to collective web platforms, the exploitation of user-generated metadata pertaining to semantic descriptions; and third, connected to streaming services, the collection of online music interaction traces on a large scale and their exploitation in recommender systems.

We review and contextualize work from ISMIR and related venues from all three phases and extrapolate current developments to outline possible scenarios of music recommendation and listening interfaces of the future.

1. INTRODUCTION

With a history of five years of ISMIR conferences, in 2004, Downie [11] attempts to define the research field of Music Information Retrieval (MIR) as "a multidisciplinary research endeavor that strives to develop innovative content-based searching schemes, novel interfaces, and evolving networked delivery mechanisms in an effort to make the world's vast store of music accessible to all". Given the music industry landscape and how people listen to music 15 years later, this definition has not only stood the test of time, but also proven to be visionary.

With its origins in Information Retrieval research (cf. [5]), one of the original motivations underlying MIR was indeed to develop technology and provide means to assist the user in finding music. As the way humans interact with technology to access and listen to music has substantially changed since then, user interfaces for music discovery remain to be a pivotal element in MIR research. [1]

In this paper, we reflect on the evolution of MIR-driven user interfaces for music browsing and discovery over the past two decades—from organizing personal music collections to streaming a personalized selection from "the world's vast store of music". Therefore, we connect major developments that have transformed and shaped MIR research in general and user interfaces in particular to prevalent and emerging listening practices at the time. We identify three main phases that have each laid the foundation for the next and review work that focuses on the specific aspects of these phases.

First, we investigate the phase of growing digital personal music collections and interfaces built upon intelligent audio processing and content description algorithms in section 2. These algorithms facilitate the automatic organization of repositories and finding music in personal collections, as well as commercial repositories according to sound qualities. Second, in section 3, we investigate the emergence of collective web platforms and their exploitation for listening interfaces. The extracted user-generated metadata often pertains to semantic descriptions and complements the content-based methods that facilitated the developments of the preceding phase. This phase also constitutes an intermediate step towards exploitation of collective listening data, which is the driving force behind the third, and ongoing phase, which is connected to streaming services (section 4). Here, the collection of online music interaction traces on a large scale and their exploitation in recommender systems are defining elements.

[1] So do interfaces for "active listening," aiming at increasingly engaging the listener through augmented experiences and allowing also musically untrained listeners to gain deeper insights into aspects of the music they are consuming [20]. However, in this work we will not emphasize these interfaces.

Extrapolating these and other ongoing developments, we outline possible scenarios of music recommendation and listening interfaces of the future in section 5.

2. PHASE 1: CONTENT-BASED MUSIC RETRIEVAL INTERFACES

Based on the technological advancements in encoding and compression of audio signals (most notably mp3) together with the establishment of the Internet as mainstream communication medium and distribution channel, and, in rapid succession, the development of high capacity portable music players in the late 1990s, digital music has not only stirred up the IT industry, but also initiated a profound change in the way people "use" music.

At the time, the most popular and conventional interfaces for such music access display the list of bibliographic information (metadata) such as titles and artist names. When the number of musical pieces in a personal music collection was not large, music interfaces with the title list and mere text searches based on bibliographic information were useful enough to browse the whole collection to choose pieces to listen to. However, as the accessible collection grows, such simple interfaces become insufficient, and new research approaches targeting the retrieval, classification, and organization of music emerge.

"Intelligent" interfaces for music retrieval became a research field of interest with the developments in content-based music retrieval [6]. A landmark in this regard was the development of query by humming systems [31] and search engines indexing sound properties of loudness, pitch, and timbre [77] that initiated the emancipation of music search systems from traditional text- and metadata-based indexing and query interfaces. While interfaces were still very much targeted at presenting results in sequential order according to relevance to a query, in the early 2000s, MIR research proposed several alternatives to facilitate music discovery.

2.1 Map-based music browsing and discovery

Interfaces that allow content-based searches for music retrieval are useful when people can formulate good queries and especially when users are looking for a particular work, but sometimes it is difficult to come up with an appropriate query when faced with a huge music collection and vague search criteria. Interfaces for music browsing and discovery are therefore proposed to let users encounter unexpected but interesting musical pieces or artists. Visualization of a music collection is one way to provide users with various bird's-eye views and comprehensive interactions. The most popular visualization is to project musical pieces or artists onto a 2D or 3D space by using music similarity. 2D visualizations also lend themselves to being applied on tabletop interfaces for intuitive access and interaction, e.g. [30]. The trend of spatially arranging collections for exploration can be seen throughout the past 20 years and is still unbroken.

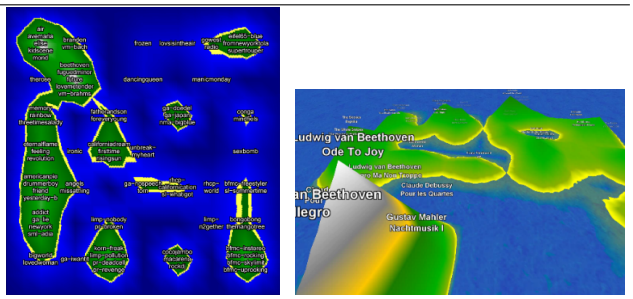


Figure 1. Islands of Music (left) and nepTune (right): music browsing interfaces that let a user explore a music collection by using a metaphor of "islands" visualizing self-organized clusters.

One of the earliest interfaces is *GenreSpace* [69] that visualizes musical pieces with genre-specific colors in a 3D space. Coloring of each piece is determined by automatic genre classification. The layout of pieces is determined by principal component analysis (PCA), which projects high-dimensional audio feature vectors into 3D positions. This idea is frequently used in other more recent interfaces.

Another early interface called *Islands of Music* [48] visualizes musical pieces on a 2D space representing an artificial landscape. It uses a self-organizing map (SOM) to arrange musical pieces so that similar pieces are located near each other. As shown on the left side in Figure 1, it uses a metaphor of "islands" that represent self-organized clusters of similar pieces. The denser the regions (more items in the same cluster), the higher the landscape (up to "mountains" for very dense regions). Sparse regions are represented by the ocean. This interface provides three different views corresponding to similarities based on three aspects: (1) timbre analysis, (2) rhythm analysis, and (3) metadata like artist and genre. A user can smoothly change focus from one view to another while exploring how the organization changes. Several extensions of the Islands of Music idea were proposed in following years. An aligned SOM is used by Pampalk et al. in [48] to enable a seamless shift of focus between clusterings created for different musical aspects, for instance, between a SOM created only on rhythm features and one created only on timbre features. The *nepTune* interface presented by Knees et al. in [36], as shown on the right side in Figure 1, enables exploration of music collections by navigating through a three-dimensional artificial landscape. Variants include a mobile version [27] and a larger-scale version using a growing hierarchical self-organizing map (GHSOM) [10] that automatically structures the map into hierarchically linked individual SOMs [57]. Neumayer et al. propose a method to automatically generate playlists by drawing a curve on the SOM visualization [47]. Lübbers and Jarke [42] present a browser employing multi-dimensional scaling (MDS) and SOM to create 3-dimensional landscapes. In contrast to the *Islands of Music* metaphor, they use an inverse height map, meaning that agglomerations of songs are visualized as valleys, while clusters are separated by mountains. Their interface further enables the user to adapt

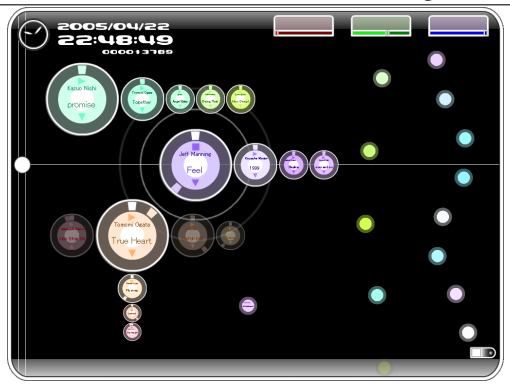


Figure 2. Musicream: A music playback interface that lets a user unexpectedly come across various musical pieces similar to those liked by the user.

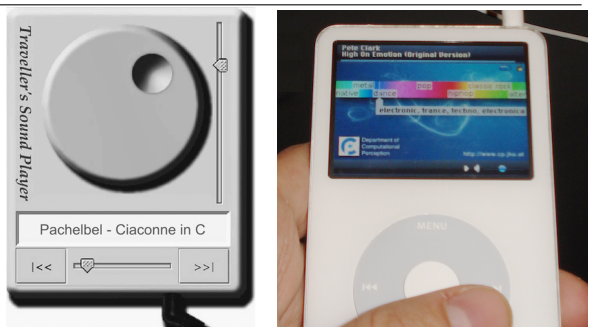


Figure 3. "Reinventing the Wheel" (left) and its application on an iPod using the dial for browsing the collection (right).

the landscape by building or removing mountains, which triggers an adaptation of the underlying similarity measure.

Another SOM-based browsing interface is *Globe of Music* [40], which maps songs to a sphere instead of a plane by means of a GeoSOM [78]. Mörchen et al. [46] employ an emergent SOM and the U-map visualization technique [70] to color-code similarities between neighboring clusters. Vembu and Baumann incorporate a dictionary of musically related terms to describe similar artists [74].

In the *Search Inside the Music* application [38], Lamere and Eck use a three-dimensional MDS projection. Their interface provides different views that arrange images of album covers according to the output of the MDS, either in a cloud, a grid, or a spiral. Vad et al. [71] apply t-SNE [72] to mood- and emotion-related descriptors, which they infer from low-level acoustic features. The result of the data projection is visualized on a two-dimensional map, around which the authors build an interface to support the creation of playlists by drawing a path and by area selection.

While the above interfaces focus on musical pieces, interfaces focusing on artists have also been investigated. For example, *Artist Map* [73] is an interface that enables users to explore and discover artists. This interface projects artists onto a 2D space and visualizes them as small dots with genre-specific, tempo-specific, or year-specific colors. This visualization can also be used to create playlists by drawing paths and specifying regions.

Other examples use, e.g., metaphors of a "galaxy" or "cosmos," or extend visualizations with additional information. *MusicGalaxy* [65], for example, is an exploration interface that uses a similarity-preserving projection of musical pieces onto a 2D galaxy space. It takes timbre, rhythm, dynamics, and lyrics into account in computing the similarity and uses an adaptive non-linear multi-focus zoom lens that can simultaneously zoom multiple regions of interest while most interfaces support only a single region zooming. As a similar metaphor, "planetarium" has been used in *Songrium*, [2] a public web service for interactive visualization and exploration of *web-native music* on video sharing services [23]. It uses similarity-preserving projection of pieces onto both 2D and 3D galaxy spaces and provides various functions: analysis and visualization of derivative works, and interactive chronological visualization and playback of musical pieces. *Instrudive* [66] enables users to browse and listen to musical pieces by focusing on instrumentation detected automatically. It visualizes each musical piece as a multicolored pie chart in which different colors denote different instruments. The ratios of the colors indicate relative duration in which the corresponding instruments appear in the piece.

[2] `https://songrium.jp`

2.2 Content-based filtering and sequential play

When a collection of music becomes huge, it is not feasible to visualize all pieces in the collection. Other types of interfaces that visualize a part of the music collection instead of the whole have also been proposed. An example is *Musicream* [21], a user interface that focuses on inducing active user interactions to discover and manage music in a huge collection. The idea behind Musicream is to see if people can break free from stereotyped thinking that music playback interfaces must be based on lists of song titles and artist names. To satisfy the desire "I want to hear something," it allows a user to unexpectedly come across various pieces similar to ones that the user likes. As shown on the right side in Figure 2, disk icons representing pieces flow one after another from top to bottom, and a user can select a disk and listen to it. By dragging a favorite disk in the flow, which serves as the query, the user can easily pick out other pieces similar to the query disk (attach similar disks) by using content-based similarity. In addition, to satisfy a desire like "I want to hear something my way," Musicream gives a user greater freedom of editing playlists by generating a playlist of playlists. Since all operations are automatically recorded, the user can also visit and retrieve a past state as if using a time machine.

The *FM4 Soundpark Player* makes content-based suggestions by showing up to five similar tracks in a graph-like manner [17] and constructing "mixtapes" from given start and end tracks [15]. *VocalFinder* [16] enables content-based retrieval of songs with vocals that have similar vocal timbre to the query song.

Visualization of a music collection is not always necessary to develop music interfaces. Stewart et al. [64] present

an interface that uses only sound auralization and haptic feedback to explore a large music collection in a two or three-dimensional space.

The article *"Reinventing the Wheel"* [51] revealed that a single-dial browsing device can be a useful interface for musical pieces stored on mobile music players. The whole collection is ordered in a circular locally-consistent playlist by using the Traveling Salesman algorithm so that similar pieces can be arranged adjacently. The user may simply turn the wheel to access different pieces. This interface also has the advantage of combining two different similarity measures, one based on timbre analysis and the other based on community metadata analysis. Figure 3 shows the conceptual prototype as well as an extended implementation on an Apple iPod [60], the most popular mobile listening device at the time.

3. PHASE 2: COLLABORATIVE AND AUTOMATIC SEMANTIC DESCRIPTION

While content-based analysis allowed for unprecedented views on music collections based on sound, interfaces built solely upon the extracted information were not able to "explain" the music contained or give semantically meaningful support for orientation within the collections. That is, while they are able to capture qualities of the sound of the contained music, they largely neglect human concepts of music organization, such as (sub-)genres or listening purposes, e.g., during activities like workouts. This information is however typically found on the web and ranges from user-generated tags to unstructured bits of expressed opinions (e.g., forum posts or comments in social media) to more detailed reviews and encyclopedic articles (containing, e.g., biographies and discography release histories). In MIR, this type of data is often referred to as community metadata or music context data [35]. These online "collaborative efforts" of describing music are resulting in a rich vocabulary of semantic labels and have shaped music retrieval interfaces towards music information systems. In parallel, platforms like *Last.fm* [3] (in this context better: *AudioScrobbler*), take advantage of users being increasingly always connected to the Internet and tracking listening events for the sake of identifying listening patterns and making recommendations, leading to the phase of automatic playlisting and music recommendation (cf. section 4). In this section, we focus on tags as a main driver of MIR research and music interfaces.

3.1 Collaborative platforms and music information systems

With music related information being ubiquitous on the web, dedicated web platforms that provide background knowledge on artists emerge, e.g., the AllMusic Guide, depending on editorial content. Using new technologies, such music information systems can, however, also be built by aggregating information extracted from various sources using text mining methods [59] or by taking advantage of

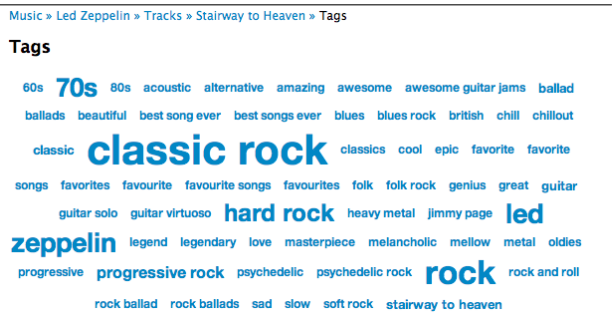


Figure 4. Last.fm tags of Led Zeppelin (source: `https://musicmachinery.com/tag/lastfm/`)

the "wisdom of the crowd" and building collaborative platform like the above mentioned Last.fm.

A central feature of Last.fm is to allow users to tag their music, ideally resulting in a democratic ground truth of what could be considered the semantic dimensions of the corresponding tracks, cf. Figure 4. However, typical problems arising with this type of information are noisy and non-trustworthy information as well as data sparsity and cold start issues mostly due to popularity biases, cf. [37].

MIR research during this phase has therefore dealt extensively with auto-tagging, i.e., automatically inferring semantic labels from the audio signal of a music piece (or related data), to overcome this shortcoming, e.g., [3, 12, 33, 44, 63, 68, 76].

Alternative approaches to generate semantic labels include human workforce. *TagATune* [39] is a game that pairs players across the Internet who try to determine whether they are listening to the same song by typing tags. In return for entertaining users, TagATune has collected interesting tags for a database of songs. Other examples of interfaces that were designed to collect useful information while engaging with music are *MajorMiner* [43], *HerdIt* [2], and *Moodswings* [34] (cf. section 4.1).

A more traditional way to obtain musically informed labels is to have human experts, e.g. trained musicians, manually label music tracks according to predefined musical categories. This approach is followed by the Music Genome Project, [4] and serves as the foundation of *Pandora's* automatic radio stations (cf. section 4).

As a consequence of these efforts, during this phase, the question of how to present and integrate this information into interfaces was secondary to the question of how to obtain it, as will become obvious next.

3.2 Visual interfaces

With the trend towards web-based interfaces, visualization and map based interfaces integrating semantic information have been proposed.

MusicRainbow [49] is a user interface for discovering unknown artists, which follows the above idea of a single-dial browsing device but features informative visualization. As shown in Figure 5, artists are mapped on a circular rainbow where colors represent different styles of music.

[3] `https://last.fm`

[4] `https://www.pandora.com/about/mgp`

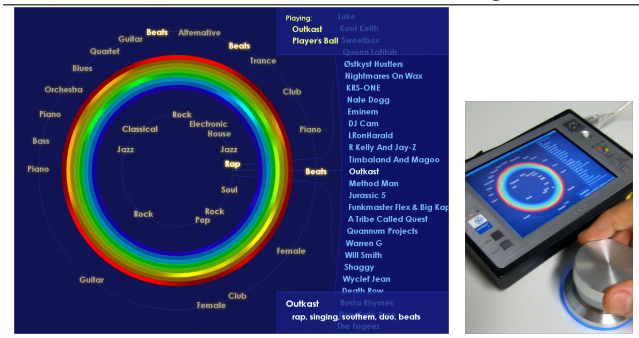


Figure 5. MusicRainbow: An artist discovery interface that enables a user to actively browse a music collection by using audio-based similarity and web-based labeling.

Similar artists are automatically mapped near each other by using the Traveling Salesman algorithm and summarized with word labels extracted from artist-related web pages. A user can rotate the rainbow by turning a knob and find an interesting artist by referring to the word labels.

The *nepTune* interface shown in Figure 1 also provides a mode that integrates text-based information extracted from artist web pages for supporting navigation in the 3D environment. To this end, labels referring to genres, instruments, origins, and eras serve as landmarks.

Other approaches explore music context data to visualize music over real geographical maps, rather than computing a clustering based on audio descriptors. For instance, Govaerts and Duval extract geographical information from biographies and integrate it into a visualization of radio station playlists [22]. Hauger and Schedl extract listening events and location information from microblogs and visualize both on a world map [25].

Lyrics are also important elements of music. By using semantic topics automatically estimated from lyrics, new types of visual interfaces for lyrics retrieval can be achieved. *LyricsRadar* [55] is a lyrics retrieval interface that uses latent Dirichlet allocation (LDA) to analyze topics of lyrics and visualizes the topic ratio for each song by using the topic radar chart. It then enables a user to find her favorite lyrics interactively. *Lyric Jumper* [67] is a lyrics-based music exploratory web service that enables a user to choose an artist based on topics of lyrics and find unfamiliar artists who have similar profile to her favorite artist. It uses an advanced topic model that incorporates an artist's profile of lyrics topics and provides various functions such as topic tendency visualization, artist ranking, artist recommendation, and lyric phrase recommendation.

4. PHASE 3: RECOMMENDER INTERFACES AND CONTINUOUS STREAMING

With ubiquitous Internet connection and a development of computer and entertainment systems to be always online, personal music collections have lost relevance to many people, as virtually all music content is available at all times. In essence, such subscription streaming services like Spotify, Pandora, Deezer, Amazon Music, or Apple Music have transformed the music business and music listening alike.

Central element to these services is the aspect of personalization, i.e., providing foremost a user-tailored view onto the available collections of allegedly tens of millions of songs. Discovery of music is therefore performed by the system, based on the user profile of past interactions, rather than by the user herself.

Interfaces for music recommendation can support music listening in more personalized ways. Music recommendation typically models personal preferences of users by using their listening histories or explicit user feedback, e.g. [7, 62]. It then generates a set of recommended musical pieces or artists for each user. This recommendation can be implemented by using collaborative filtering based on users' past behaviors and exhibits patterns of music similarity not captured by content-based approaches [61]. When the playback order of recommended pieces is important, automatic playlist generation is also used, e.g. [4, 24, 45].

The main challenges of this type of algorithms are, as in all other domains of recommender systems, cold start problems. The approach taken to remedy these are again to integrate additional information on the music items to recommend, i.e. facets of content and metadata as applied in the earlier phases, by building hybrid recommenders on top of pure collaborative filtering. Additionally, context-awareness plays an important role, for instance to recommend music for daily activities [75].

An overview over aspects, techniques, and challenges of music recommender systems can be found in [58]. Therefore, in this section, we do not elaborate on the basics of music recommender systems but highlight again interfaces that focus on personalization and user-centric aspects, as we consider these to be the bridge to future intelligent music listening interfaces.

4.1 Recommender interfaces

Although most related studies have focused on methods and algorithms of music recommendation and playlist generation, some studies focus on interfaces.

MusicSun [50] is a user interface for artist recommendation. A user first puts favorite artist names into a "sun" metaphor, a circle in the center of the screen, and then obtains a ranked list of recommended artists. The sun is visualized with some surrounding "rays" that are labeled with words to summarize the query artists in the sun. By interactively selecting a ray, the user can look at and listen to the corresponding recommended artists.

Moodplay [1] is an interactive music recommender system that uses a hybrid recommendation algorithm based on mood metadata and audio content. A user first constructs a profile by entering favorite artist names and then obtains a ranked list of recommended artists. It highlights those artist positions in a latent mood space visualization, showing various mood labels. The centroid of profile artist positions is used to recommend nearby artists. The change of a user's preference is interactively modeled by moving

in this space and its trail is used to recommend artists.

In *MoodSwings* [34], users try to match each other while tracing the trajectory of music through a 2D emotion space. The users' input provides metadata on the emotional impression of songs as it changes over time.

Recent studies deal with the design of recommender user interfaces regarding complexity and user control [28] and the implications of recommender techniques on the discovery of music in playlist building [32].

4.2 Psychologically-inspired music recommendation

Recently, music recommender research is experiencing a boost on topics related to psychology-informed recommendation. In particular the psychological concepts of personality and affect (mood and emotion) are increasingly integrated into prototypes. The motivation for this is that while listening to music both personality traits and affective states have been shown to strongly influence music preferences [14, 52, 56].

Lu and Tintarev [41] propose a system that re-ranks results of a collaborative filtering approach according to the degree of diversity each song contributes to the recommendation list. Since previous studies showed that personality is most strongly correlated with music key, genre, and number of artists, the authors implement diversity through these features and adjust results depending on the listener's personality. Fernández-Tobías et al. [13] propose a personality-aware matrix factorization approach that integrates a latent user factor describing users' personality in terms of the Big Five/OCEAN model (openness, conscientiousness, extraversion, agreeableness, and neuroticism) [29]. Deng et al. [9] propose an emotion-aware recommender for which they extract music listening information and emotions from posts in Sina Weibo, [5] a popular Chinese microblogging service, adopting a lexicon-based approach (Chinese dictionaries and emoticons). *FocusMusicRecommender* [79] recommends and plays back musical pieces suitable to the user's current concentration level estimated from the user's behavior history.

5. THE NEXT PHASE: THE FUTURE OF INTELLIGENT MUSIC USER INTERFACES

In terms of interfaces, we observe strong trends towards context-awareness and personalization, also on the level of individual user and personality traits that should guide the recommendation process when other sufficient interaction data is unavailable. The central challenge behind these facets is to accurately infer the user's intent in an action (listening, skipping, etc.), i.e., to uncover the reasons why humans indulge in music, from the comparatively limited signal that is received.

On the other hand, we see the development in the realm of music generation and variation algorithms, which permit to create content based on large repositories of examples (cf. recent work by *Google Magenta* [6] [26, 53, 54]) and/or with the help of informed rules and templates, e.g., for automatic video soundtrack creation or adaptive music generation in video games marketed by companies such as *Jukedeck* [7] or *Melodrive*, [8] respectively.

In the long run, we expect the border of these domains to blur, i.e., there will be no difference in accessing existing, recorded music and music automatically created by the system tailored to the listener's needs. More concretely, as discussed as one of grand challenges in MIR in [19], we envision music streaming systems that deliver preferred content based on the user's current state and situational context, automatically change existing music content to fit the context of the user, e.g., by varying instruments, arrangements, or tempo of the track, [9] and even create new music based on the given setting.

With the current knowledge of streaming platforms about a user's preferences, context sensing devices running the music apps, and first algorithms to variate and generate content, the necessary ingredients for such a development seem to be available already.

6. CONCLUSIONS

We identified three phases of listening culture and discussed corresponding intelligent interfaces. Interfaces pertaining to the first phase focus on structuring and visualizing smaller scale music collections, such as personal collections or early digital sales repositories. In terms of research prototypes, this phase is most driven by content-based MIR algorithms. The second phase deals with web-based interfaces and information systems, with a strong focus on textual descriptions in the form of collaborative tags. MIR research during this phase therefore deals with automatic tagging of music and utilization of tag information in interfaces. Finally, the third and current phase is shaped by lean back experiences driven by automatic playlist algorithms and personalized recommendation systems. MIR research is therefore shifting towards exploitation of user interaction data, however always with a focus on integration of content-based methods, community metadata, user information, and contextual information of the user. While the former three strategies are typically applied to remedy cold start problems, integrating context-awareness is often an additional source thereof.

The trend of personalizing listening experiences leads us to belief that, in the not too distant future, music listening will not only be a matter of delivering the right music at the right time, but also of generating and "shaping" the right music for the situation the user is in. We will therefore see a confluence of music retrieval and (interactive) music generation – with ample challenges for MIR research ahead.

[5] `http://weibo.com`

[6] `https://magenta.tensorflow.org`

[7] `https://www.jukedeck.com`

[8] `https://melodrive.com`

[9] One of the earliest approaches to customize or personalize existing music is *"music touch-up"* [18], where several examples such as *Drumix* [80] and *AutoMashUpper* [8] were developed. Lamere's *Infinite Jukebox* (`http://infinitejukebox.playlistmachinery.com`) can also be seen as an example toward this direction.

7. REFERENCES

[1] Ivana Andjelkovic, Denis Parra, and John O'Donovan. Moodplay: Interactive mood-based music discovery and recommendation. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization (UMAP 2016)*, pages 275–279, 2016.

[2] Luke Barrington, Damien O'Malley, Douglas Turnbull, and Gert Lanckriet. User-centered design of a social game to tag music. In *Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP 2009)*, pages 7–10, 2009.

[3] Thierry Bertin-Mahieux, Douglas Eck, François Maillet, and Paul Lamere. Autotagger: A model for predicting social tags from acoustic features on large music databases. *Journal of New Music Research*, 37(2):115–135, 2008.

[4] Geoffray Bonnin and Dietmar Jannach. Automated generation of music playlists: Survey and experiments. *ACM Computing Surveys*, 47(2):26:1–26:35, 2014.

[5] Donald Byrd and Michael Fingerhut. The history of ISMIR – a short happy tale. *D-Lib Magazine*, 8(11), 2002.

[6] Michael Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.

[7] Òscar Celma. *Music Recommendation and Discovery – The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer, 2010.

[8] Matthew E. P. Davies, Philippe Hamel, Kazuyoshi Yoshii, and Masataka Goto. AutoMashUpper: Automatic creation of multi-song music mashups. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(12):1726–1737, 2014.

[9] Shuiguang Deng, Dongjing Wang, Xitong Li, and Guandong Xu. Exploring user emotion in microblogs for music recommendation. *Expert Systems with Applications*, 42(23):9284–9293, 2015.

[10] Michael Dittenbach, Dieter Merkl, and Andreas Rauber. Hierarchical clustering of document archives with the growing hierarchical self-organizing map. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN 2001)*, 2001.

[11] J. Stephen Downie. The scientific evaluation of music information retrieval systems: Foundations and future. *Computer Music Journal*, 28:12–23, 2004.

[12] Douglas Eck, Paul Lamere, Thierry Bertin-Mahieux, and Stephen Green. Automatic generation of social tags for music recommendation. In *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, 2008.

[13] Ignacio Fernández-Tobías, Matthias Braunhofer, Mehdi Elahi, Francesco Ricci, and Iván Cantador. Alleviating the new user problem in collaborative filtering by exploiting personality information. *User Modeling and User-Adapted Interaction*, 26(2-3):221–255, 2016.

[14] Bruce Ferwerda, Marko Tkalcic, and Markus Schedl. Personality traits and music genres: What do people prefer to listen to? In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization (UMAP 2017)*, pages 285–288, 2017.

[15] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Gerhard Widmer. Playlist generation using start and end songs. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, pages 173–178, 2008.

[16] Hiromasa Fujihara, Masataka Goto, Tetsuro Kitahara, and Hiroshi G. Okuno. A modeling of singing voice robust to accompaniment sounds and its application to singer identification and vocal-timbre-similarity-based music information retrieval. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):638–648, 2010.

[17] Martin Gasser and Arthur Flexer. FM4 Soundpark: Audio-based music recommendation in everyday use. In *Proceedings of the 6th Sound and Music Computing Conference (SMC 2009)*, 2009.

[18] Masataka Goto. Active music listening interfaces based on signal processing. In *Proceedings of the 2007 IEEE International Conference on Acoustics, Speech, and Signal Processing (IEEE ICASSP 2007)*, pages 1441–1444, 2007.

[19] Masataka Goto. Grand challenges in music information research. In Meinard Muller, Masataka Goto, and Markus Schedl, editors, *Dagstuhl Follow-Ups: Multimodal Music Processing*, pages 217–225. Dagstuhl Publishing, 2012.

[20] Masataka Goto and Roger B. Dannenberg. Music interfaces based on automatic music signal analysis: New ways to create and listen to music. *IEEE Signal Processing Magazine*, 36(1):74–81, 2019.

[21] Masataka Goto and Takayuki Goto. Musicream: Integrated music-listening interface for active, flexible, and unexpected encounters with musical pieces. *IPSJ (Information Processing Society of Japan) Journal*, 50(12):2923–2936, 2009.

[22] Sten Govaerts and Erik Duval. A web-based approach to determine the origin of an artist. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, 2009.

[23] Masahiro Hamasaki, Masataka Goto, and Tomoyasu Nakano. Songrium: Browsing and listening environment for music content creation community. In *Proceedings of the 12th Sound and Music Computing Conference (SMC 2015)*, pages 23–30, 2015.

[24] Negar Hariri, Bamshad Mobasher, and Robin Burke. Context-aware music recommendation based on latent-topic sequential patterns. In *Proceedings of the 6th ACM Conference on Recommender Systems (RecSys 2012)*, pages 131–138, 2012.

[25] David Hauger and Markus Schedl. Exploring geospatial music listening patterns in microblog data. In *Proceedings of the 10th International Workshop on Adaptive Multimedia Retrieval (AMR 2012)*, 2012.

[26] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew Dai, Matt Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer: Generating music with long-term structure. In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, 2019.

[27] Sebastian Huber, Markus Schedl, and Peter Knees. nepDroid: An intelligent mobile music player. In *Proceedings of the ACM International Conference on Multimedia Retrieval (ACM ICMR 2012)*, 2012.

[28] Yucheng Jin, Nava Tintarev, and Katrien Verbert. Effects of personal characteristics on music recommender systems with different levels of controllability. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys 2018)*, pages 13–21, 2018.

[29] Oliver P. John, Eileen M. Donahue, and Robert L. Kentle. The big five inventory—versions 4a and 54, 1991.

[30] Carles F. Julià and Sergi Jordà. SongExplorer: A tabletop application for exploring large collections of songs. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, 2009.

[31] Tetsuya Kageyama, Kazuhiro Mochizuki, and Yosuke Takashima. Melody retrieval with humming. In *Proceedings of the 1993 International Computer Music Conference (ICMC 1993)*, pages 349–351, 1993.

[32] Iman Kamehkhosh, Geoffray Bonnin, and Dietmar Jannach. Effects of recommendations on the playlist creation behavior of users. *User Modeling and User-Adapted Interaction*, 2019.

[33] Joon Hee Kim, Brian Tomasik, and Douglas Turnbull. Using artist similarity to propagate semantic information. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, 2009.

[34] Youngmoo E. Kim, Erik M. Schmidt, and Lloyd Emelle. MoodSwings: A collaborative game for music mood label collection. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, pages 231–236, 2008.

[35] Peter Knees and Markus Schedl. A survey of music similarity and recommendation from music context data. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 10(1), 2013.

[36] Peter Knees, Markus Schedl, Tim Pohle, and Gerhard Widmer. An innovative three-dimensional user interface for exploring music collections enriched with meta-information from the web. In *Proceedings of the 14th ACM International Conference on Multimedia (ACM Multimedia 2006)*, 2006.

[37] Paul Lamere. Social tagging and music information retrieval. *Journal of New Music Research*, 37(2):101–114, 2008.

[38] Paul Lamere and Douglas Eck. Using 3D visualizations to explore and discover music. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[39] Edith L. M. Law, Luis von Ahn, Roger B. Dannenberg, and Mike Crawford. TagATune: A game for music and sound annotation. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, pages 361–364, 2007.

[40] Stefan Leitich and Martin Topf. Globe of Music - music library visualization using GeoSOM. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[41] Feng Lu and Nava Tintarev. A diversity adjusting strategy with personality for music recommendation. In *Proceedings of the 5th Joint Workshop on Interfaces and Human Decision Making for Recommender Systems, co-located with ACM Conference on Recommender Systems (RecSys 2018)*, 2018.

[42] Dominik Lübbers and Matthias Jarke. Adaptive multimodal exploration of music collections. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, 2009.

[43] Michael I. Mandel and Daniel P.W. Ellis. A web-based game for collecting music metadata. *Journal of New Music Research*, 37(2):151–165, 2008.

[44] Michael I. Mandel, Razvan Pascanu, Douglas Eck, Yoshua Bengio, Luca M. Aiello, Rossano Schifanella, and Filippo Menczer. Contextual tag inference. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 7S(1):32:1–32:18, 2011.

[45] Brian McFee and Gert Lanckriet. The natural language of playlists. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, 2011.

[46] Fabian Mörchen, Alfred Ultsch, Mario Nöcker, and Christian Stamm. Databionic visualization of music collections according to perceptual distance. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, 2005.

[47] Robert Neumayer, Michael Dittenbach, and Andreas Rauber. PlaySOM and PocketSOMPlayer, alternative interfaces to large music collections. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, 2005.

[48] Elias Pampalk, Simon Dixon, and Gerhard Widmer. Exploring music collections by browsing different views. *Computer Music Journal*, 28(2):49–62, 2004.

[49] Elias Pampalk and Masataka Goto. MusicRainbow: A new user interface to discover artists using audio-based similarity and web-based labeling. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, 2006.

[50] Elias Pampalk and Masataka Goto. MusicSun: A new approach to artist recommendation. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[51] Tim Pohle, Peter Knees, Markus Schedl, Elias Pampalk, and Gerhard Widmer. "Reinventing the Wheel": A novel approach to music player interfaces. *IEEE Transactions on Multimedia*, 9(3):567–575, 2007.

[52] Peter J. Rentfrow and Samuel D. Gosling. The do re mi's of everyday life: The structure and personality correlates of music preferences. *Journal of Personality and Social Psychology*, 84(6):1236–1256, 2003.

[53] Adam Roberts, Jesse Engel, Sageev Oore, and Douglas Eck. Learning latent representations of music to generate interactive musical palettes. In *Proceedings of the 2018 ACM Workshop on Intelligent Music Interfaces for Listening and Creation (MILC 2018)*, 2018.

[54] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, pages 4364–4373, 2018.

[55] Shoto Sasaki, Kazuyoshi Yoshii, Tomoyasu Nakano, Masataka Goto, and Shigeo Morisihima. LyricsRadar: A lyrics retrieval system based on latent topics of lyrics. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, pages 585–590, 2014.

[56] Markus Schedl, Emilia Gómez, Erika Trent, Marko Tkalčič, Hamid Eghbal-Zadeh, and Agustín Martorell. On the interrelation between listener characteristics and the perception of emotions in classical orchestra music. *IEEE Transactions on Affective Computing*, 9:507–525, 2018.

[57] Markus Schedl, Christian Höglinger, and Peter Knees. Large-scale music exploration in hierarchically organized landscapes using prototypicality information. In *Proceedings of the ACM International Conference on Multimedia Retrieval (ACM ICMR 2011)*, 2011.

[58] Markus Schedl, Peter Knees, Brian McFee, Dmitry Bogdanov, and Marius Kaminskas. Music Recommender Systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 453–492. Springer, 2nd edition, 2015.

[59] Markus Schedl, Gerhard Widmer, Peter Knees, and Tim Pohle. A music information system automatically generated via web content mining techniques. *Information Processing & Management*, 47:426–439, 2011.

[60] Dominik Schnitzer, Tim Pohle, Peter Knees, and Gerhard Widmer. One-touch access to music on mobile devices. In *Proceedings of the 6th International Conference on Mobile and Ubiquitous Multimedia (MUM 2007)*, pages 103–109, 2007.

[61] Malcolm Slaney. Web-scale multimedia analysis: Does content matter? *IEEE MultiMedia*, 18(2):12–15, 2011.

[62] Malcolm Slaney and William White. Similarity based on rating data. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, pages 479–484, 2007.

[63] Mohamed Sordo. *Semantic Annotation of Music Collections: A Computational Approach*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2012.

[64] Rebecca Stewart, Mark Levy, and Mark Sandler. 3D interactive environment for music collection navigation. In *Proceedings of the 11th International Conference on Digital Audio Effects (DAFx-08)*, 2008.

[65] Sebastian Stober and Andreas Nürnberger. MusicGalaxy — an adaptive user-interface for exploratory music retrieval. In *Proceedings of the 7th Sound and Music Computing Conference (SMC 2010)*, pages 23–30, 2010.

[66] Takumi Takahashi, Satoru Fukayama, and Masataka Goto. Instrudive: A music visualization system based on automatically recognized instrumentation. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR 2018)*, pages 561–568, 2018.

[67] Kosetsu Tsukuda, Keisuke Ishida, and Masataka Goto. Lyric Jumper: A lyrics-based music exploratory web service by modeling lyrics generative process. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, pages 544–551, 2017.

[68] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Towards musical query-by-semantic-description using the CAL500 data set. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (ACM SIGIR 2007)*, 2007.

[69] George Tzanetakis, Georg Essl, and Perry Cook. Automatic musical genre classification of audio signals. In *Proceedings of the 2nd International Symposium on Music Information Retrieval (ISMIR 2001)*, pages 205–210, 2001.

[70] Alfred Ultsch and Hans Peter Siemon. Kohonen's self-organizing feature maps for exploratory data analysis. In *Proceedings of the International Neural Network Conference (INNC 1990)*, pages 305–308, 1990.

[71] Beatrix Vad, Daniel Boland, John Williamson, Roderick Murray-Smith, and Peter Berg Steffensen. Design and evaluation of a probabilistic music projection interface. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, pages 134–140, 2015.

[72] Laurens van der Maaten and Geoffrey Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[73] Rob van Gulik and Fabio Vignoli. Visual playlist generation on the artist map. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, pages 520–523, 2005.

[74] Shankar Vembu and Stephan Baumann. A self-organizing map based knowledge discovery for music recommendation systems. In *Proceedings of the 2nd International Symposium on Computer Music Modeling and Retrieval (CMMR 2004)*, 2004.

[75] Xinxi Wang, David Rosenblum, and Ye Wang. Context-aware mobile music recommendation for daily activities. In *Proceedings of the 20th ACM International Conference on Multimedia (ACM Multimedia 2012)*, pages 99–108, 2012.

[76] Brian Whitman and Daniel P. W. Ellis. Automatic record reviews. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, pages 470–477, 2004.

[77] Erling Wold, Thom Blum, Douglas Keislar, and James Wheaton. Content-based classification, search, and retrieval of audio. *IEEE MultiMedia*, 3(3):27–36, 1996.

[78] Yingxin Wu and Masahiro Takatsuka. Spherical self-organizing map using efficient indexed geodesic data structure. *Neural Networks*, 19(6–7):900–910, 2006.

[79] Hiromu Yakura, Tomoyasu Nakano, and Masataka Goto. FocusMusicRecommender: A system for recommending music to listen to while working. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces (ACM IUI 2018)*, pages 7–17, 2018.

[80] Kazuyoshi Yoshii, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. Drumix: An audio player with functions of realtime drum-part rearrangement for active music listening. *IPSJ (Information Processing Society of Japan) Journal*, 48(3):1229–1239, 2007.

20 YEARS OF AUTOMATIC CHORD RECOGNITION FROM AUDIO

Johan Pauwels[1] **Ken O'Hanlon**[1] **Emilia Gómez**[2] **Mark B. Sandler**[1]
[1] Centre for Digital Music, Queen Mary University of London
[2] Music Technology Group, Universitat Pompeu Fabra
{j.pauwels, k.o.ohanlon, mark.sandler}@qmul.ac.uk, emilia.gomez@upf.edu

ABSTRACT

In 1999, Fujishima published *Realtime Chord Recognition of Musical Sound: a System using Common Lisp Music*. This paper kickstarted an active research topic that has been popular in and around the ISMIR community. The field of Automatic Chord Recognition (ACR) has evolved considerably from early knowledge-based systems towards data-driven methods, with neural network approaches arguably being central to current ACR research. Nonetheless, many of its core issues were already addressed or referred to in the Fujishima paper. In this paper, we review those twenty years of ACR according to these issues. We furthermore attempt to frame current directions in the field in order to establish some perspective for future research.

1. INTRODUCTION

This year marks the twentieth anniversary of Fujishima's [17] seminal ACR system. In this work the author proposed the calculation of a 12-D chroma feature which gets compared to a dictionary of binary chord templates. The label of the most similar chord template is then considered to be the chord output. Fujishima also proposed exploiting the temporal continuity of chords by smoothing the chromas across time to produce less noisy labels, and suggested that musical information could be exploited. This system outline created the framework within which much of the early research on ACR would happen.

Perhaps the most striking evolution in ACR has been the move from knowledge-driven to data-driven systems. Initially, data-driven elements were used as one-for-one replacements, or additions, of elements to the framework set by Fujishima. Examples are more sophisticated chroma features, learnt Gaussian chord models or HMM-based temporal models. More recently ACR research has been dominated by Deep Learning (DL). In DL-ACR the relationship of tasks and elements becomes blurred as systems have become more integrated. This is perhaps most evident in the system of McFee and Bello [40], which appears superficially to be a singular unit, although closer inspection reveals convolutional filters providing short-term context, recurrent elements modelling musical language, chroma, and auxiliary targets for extra musical context. We therefore consider that the system outline provided by Fujishima is still valid in a discussion of ACR.

ACR systems have developed considerably in these twenty years and commercial products providing such functionality have recently been developed. However, there is still room for improvement. For instance, while ACR results have continually progressed, complex chords are still recognised less well than major and minor triads. Nonetheless, ACR results have improved to the point where the ambiguity in chord labels is now becoming a topic of research. Such ambiguity can be introduced by the chord labels themselves, or may result from different interpretations of chord and melody. An interesting aspect of modern ACR research derived from user research in commercial products is the prediction of user interpretations in the presence of ambiguity.

In the rest of this paper we review previous and current research in ACR. Based on such review, we propose several areas that have potential for improvement in ACR. We first consider the related features and chord models, before discussing temporal and musical context. We furthermore consider ambiguity and subjectivity in ACR, and problems associated with chord vocabularies before concluding.

2. PROBLEM 1: FINDING AN APPROPRIATE FEATURE REPRESENTATION

The chroma feature has been perhaps the most influential idea in ACR, and much early ACR research focused on producing chroma variants. Typically the chroma feature is calculated by summing the energy of elements of the same pitch class in a preliminary pitch spectrum feature. In Fujishima's paper, and subsequent others, the pitch feature was calculated by gathering the energy under windows of a spectrogram that are positioned logarithmically in frequency. Later approaches used a constant-Q transform (CQT) [4] which places windows on a multi-scale spectrogram, affording higher resolution of low frequencies, at the cost of lower temporal resolution. Such pitch-based spectra are more compact than linear spectrograms and afford simple summation in the chroma calculation, a process referred to as pitch folding. Indeed, pitch folding of logarithmic spectra into chroma features is perhaps the most important technical aspect of Fujishima's paper, as it

affords a low-dimensional semantically meaningful feature that is easily derived. However, the log-frequency spectrum may result in corruption of harmonic structure as e.g. the second overtone of a root note is located in a pitch bin labelled with the class of its perfect fifth. Such windowing also results in bins of higher pitch being of large frequency width, and possibly containing energy from several pitch class sources. However, even in the presence of such potential sources of error the pitch folded chroma feature has endured as a staple of ACR.

Many chroma variants have been designed to attempt to counter the negative effects of pitch folding. In the simplest case, a spectral weighting is employed that lowers the effect given to higher pitches, which are more likely to contain misassigned harmonics. Other researchers have applied extra weight based upon the harmonic structure in the spectrogram, for instance the harmonic product spectrum is windowed and folded in the Enhanced Pitch Class Profile [34] and in [41]. Alternatively note spectral templates are employed through convolution [18] and for spectral decomposition [38] although the log-frequency windowed spectrogram has generally prevailed. Other chroma refinements through pitch feature manipulation have been proposed. Placing less effect on the relative coefficients in the pitch feature, thereby inducing a level of timbre-invariance, is encouraged by regularisation using e.g. log compression [42] which is seen to be useful for ACR [10].

Researchers have attempted to create chroma with sharper definition. High-resolution spectral methods such as parabolic interpretation [18] and spectral reassignment [27, 46] have been used in order to sharpen a spectrogram before windowing. Likewise, sharpness of features can be diminished when a piece is not tuned to the standard 440Hz, and tuning estimation is often performed to counter this problem [18, 21]. Alternatively, higher dimensional chroma have been employed to avoid these pitfalls [18,59].

In DL-ACR systems, the network is expected to learn any necessary weightings or transforms from training on data. DNNs are often employed as direct classifiers, and for ACR can be trained with one-hot chord class vectors [3,14,15,61,68]. Using chroma feature targets has become a popular design choice in ACR [30,31]. As in more traditional ACR methods, CQT and other log-frequency spectra are seen to predominate as input features [24, 30, 40, 61]. Different inputs have occasionally been employed, e.g. the Harmonic CQT is employed in [64]. Dimensionality reduction of input data has been explored with principal component analysis (PCA) applied to a spectrogram [3] and to a CQT [68]. It is unclear whether PCA was useful other than for dimensionality reduction, and its related decrease in computational expense. Tuning is often ignored, possibly with the assumption that the data-driven systems have the capability to learn to deal with tuning errors. Some exceptions to this are the inputs used in [14,15], while training data of a CNN-based ACR system is augmented using detuning in [31].

So far there is a lack of comparative evaluation of the effects of the input feature in DL-ACR, with most papers instead focussing on the effects of later steps in the processing chain. In the particular case where chroma targets are employed in DL-ACR, the DNN can be seen simply as a replacement for e.g. tuning, compression, weighting and pitch folding. It is no surprise that fast convergence is observed in such DNNs, although rarely reported. Few alternatives to logarithmic spectra have been considered, unlike other music processing tasks where raw audio input observed has been examined [56]. We propose that the use of linear spectra for DL-ACR should be explored. Linear spectrograms do come with the caveat that invariance to pitch-shift is lost, a property that is attractive for use with CNNs. However, overtones of a given root frequency can be expected to be found approximately equidistant from each other in a linear spectrogram, a structure that should also be extractable. In the log-frequency spectrogram, such overtone structure is not so simply presented because harmonics of orders that do not possess an integer base-2 logarithm are misassigned in pitch class. One can consider this a form of information loss that may be detrimental to ACR, particularly when a large vocabulary of chords is employed and some exploration of spectrogram-based training should be undertaken. Alternatively, a multiple input DNN system could be employed using both CQT and Fourier spectrograms.

3. PROBLEM 2: DEFINING WHAT A CHORD LOOKS LIKE IN FEATURE SPACE

Chord classification requires models of chords that can be compared to a given feature such as chroma. Originally, the use of binary chroma templates for classification was proposed in tandem with the pitch-folded chroma [17]. In such a binary chroma template each pitch expected to be active in a chord is set to one. Comparison of a chroma vector with a dictionary of binary templates affords a simple classification approach to chord recognition, with each chroma vector labelled according to the most similar template. Most ACR research has similarly focussed on the comparison of chroma-based features and chord models. The most notable early alternative chord feature is the tonal centroid, or tonnetz, feature [20] which is actually a partial Fourier Transform of a chroma feature. Binary templates formed the basis of much early ACR research [21] and many chroma estimation methods [33, 38] implicitly targeted outputting chroma vectors similar to binary vectors, a goal that can still be seen in DNNs when chroma targets are employed [30, 64]. Binary templates possess the ability to model a chord when there is no representative data available, and despite their simplicity are often effective [10]. However, binary templates may be unrealistic as the effects of misasssignment of harmonics in chroma features are ignored.

An alternative perspective is to place less emphasis on manipulating chroma vectors, and create templates that are more similar to the expected data. A template can be synthesised as in [48, 50], where chord templates are formed by summing spectra of several note templates, parameterised by a number of harmonics and a roll-off factor

determining the energy in each harmonic [19]. This approach has largely been overlooked as data-driven methods that naturally learn models akin to the data became popular. Early data-driven models learnt chord models from labelled chroma features, with several different model types explored. Perhaps the most common of these have been multivariate Gaussian (mixture) models, a generative approach learnt from data, which have been applied in the context of chroma features [10, 59] and tonal centroid features [35], and also in the context of DNN features [24] Other chroma based classifiers have included support vector machines [41, 57], which have also been applied to DNN chroma outputs [68], and random forests [25].

Other research has looked beyond chroma, considering chord classification on the full pitch spectrum. A linear full spectrum classifier is applied in [8] while logistic regression is used in [30] where it is seen to improve on a similar classification applied to chroma features. Such results indicate some limitations for ACR of the chroma feature, which may lose useful information in the pitch folding process. The chroma representation is compact and octave-invariant, qualities that have allowed templates and simple Gaussians to be effective in ACR. Such approaches may be less viable in the context of extra spectral information, with data-driven methods more likely to be able to deal with the extra information coming from the increased dimensionality. The octave-invariance also prevents the chroma feature from distinguishing between different inversions of a given chord. A bass chroma calculated from a window on the lower octaves of the pitch feature was introduced in [39] in order to capture inversions. Extending this, a three windowed chroma with Gaussian models was proposed in [9] approximating full-spectrum analysis.

In DL-ACR there are different options for training and classification. Aside from chroma target outputs, DNNs can also employ more standard black-box classification vectors where each element denotes the likelihood of a different chord [14, 15, 68] or can even be trained using both chroma and label targets [3]. Furthermore, chord classification with DNNs may be performed using activations of the penultimate layer of the network as a feature, which may be passed to a subsequent network [3, 31, 61].

A new perspective is seen in the most recent DL-ACR methods, which are being trained to learn extra information encoded through the use of auxiliary targets. This can be clearly seen in the works [40] and [64] where alongside the chroma feature, one-hot feature targets representing the bass note [40, 64], root note [40], highest pitch note [64] alongside a distinctive no-chord target [40] have been introduced to network training. Such approaches, which we call *target label engineering* should be of benefit in ACR. More possibilities for auxilary targets may exist. This target label engineering displays a turnaround in ACR feature research; where traditionally the focus has been on manipulating features close to the input, it seems that more attention may now be given to the target labels, and the sorts of information that ACR researchers might like to extract.

4. PROBLEM 3: THE MISMATCH BETWEEN PROCESSING RATE AND CHORD RATE

To locate chords in time, feature representations consist of multiple time-localised frames. The simplest way is to create features at a constant, but arbitrary rate, which determines the processing rate of the entire system. This approach was taken by Fujishima, whose features had a rate of 3.906 Hz. In this case, it is important to make the rate high enough, because it effectively imposes a time grid onto which all chord labels are projected, and making this grid too coarse leads to misalignment at the chord boundaries. Therefore later approaches generally increased this feature rate to the order of tens of Hz.

The rate of chord changes is typically an order of magnitude higher than the frame rate. Although the exact number depends on the music piece, the Isophonics dataset used for MIREX has an average chord change rate of 0.46 Hz for example [52]. Especially in the traditional methods where the features are processed frame-wise, the high frame rate compared to the chord rate leads to a high risk of fragmented chord output. A number of techniques have therefore been tried to enforce temporal continuity between frames, starting with a simple smoothing filter. Fujishima himself used a mean filter, but median [47] filters have also been used. These smoothing filters can be applied either to chord probabilities [48] or to the features themselves [2] to remove noise such as percussive sounds and non-harmonic melody notes. In the latter case, a chord model then works on a smoother feature representation, which indirectly also leads to a less fragmented chord output.

The drawback of blindly applying a smoothing filter is that chord transitions may be smeared, leading to inaccuracy in boundary estimation and will cause short chords to be smoothed out even if they are very apparent in the signal. A better method is therefore to consider the chord matching outputs as the observations of an HMM and smooth them with the Viterbi algorithm [2, 5, 59]. The strength of each of the chord candidates is then taken into account and the diagonal elements of the transition matrix control the probability of a chord change [2]. The Viterbi smoothing has been shown to outperform filtering of the chroma output, and filtering of the feature representation does not bring any additional benefits [10].

A consequence of using a HMM for smoothing is that the imposed duration distribution takes the shape of a geometric distribution, meaning that the shorter a chord, the more likely it becomes. Since this obviously is not a realistic distribution, the usage of duration-explicit HMMs has been explored, which allow arbitrary duration distributions. The shape of the distribution was not found to have a major influence on the results, however [8].

An advantage of modern, recurrent neural network (RNN) based approaches is that the chord duration distribution is learnt automatically as part of the chord modelling, without further intervention. RNNs [3] or, more recently, long short-term memory (LSTM) units [16, 61, 64] are fed the feature frames one by one, but remember previ-

ous input which can be used in the prediction of the current frame. With the advent of bidirectional LSTMs [16, 64], future frames can now also be taken into account.

Feedforward networks do not learn a chord duration distribution, as they have no memory elements, but achieve temporal stability by processing multiple time frames at once instead of isolated frames [22]. The entire local environment can then be used to learn the label of the middle frame, which includes learning to integrate over time and to avoid short disturbances. A context window of 1.5 s was found to be optimal in [30]. Of course, convolutional-recurrent models [40] combine the benefits of both.

Following sporadic usage of conditional random fields (CRFs) in traditional systems [5], the combination of CRFs and deep learning is gaining popularity [31, 64]. In this case, a CRF is stacked onto a neural network as the last layer to smooth the output, where one advantage is that they can be trained together for maximum discriminability.

Another way of handling the difference between feature and chord rates is to reduce the discrepancy from the start. This can be accomplished by segmenting the time axis of the feature representation in a musically meaningful way. A first option is to use the output of a beat-tracker to resample the feature representation onto a beat-synchronous grid [2, 36, 60]. The underlying assumption is that chords only change at beat times. The features can then be smoothed over the inter-beat interval without risk of blurring the chord boundaries. It has been shown, however, that beat-synchronous processing may not be advantageous compared to smoothing the chord output with an HMM [10]. A potential reason is the reliance of this method on a correct beat estimation, which was not as common at the time these experiments were performed as it is today. A certain benefit of beat-synchronous features is that the processing rate of systems built around them is lower (0.33–1 Hz for 60–180 BPM), so they run faster.

A final possibility is to explicitly determine chord boundaries before attempting to identify the chords they delineate [13, 20]. This way, the features or chord output can be maximally smoothed without drawbacks, but determining a good chord segmentation function is hardly an easier problem. A recent deep learning approach consists of two stages [64], the first determines the chord segmentation and triad using a small vocabulary, while the second stage picks the final chord type from a larger vocabulary.

Going forwards, we note that blind feature segmentation has been amply used by neural network approaches, but musically meaningful segmentation has not. Although one of the advantages of deep learning is that a network can come up with the most optimal feature representation itself, making the input to the network more musically explicit would be worth investigating. Feeding beat-synchronous features into a network would allow it to learn chord duration distributions expressed in terms of beats instead of frames, which could be more expressive. Such an approach would accumulate the errors of the beat tracking though, so multi-task learning [65] where a single network jointly learns to predict multiple outputs might be better. Beat-tracking could be learnt together with chord recognition. In case intermediate features are desired, chroma could be calculated together with chord segmentation.

5. PROBLEM 4: ACHIEVING LONG-TERM CONSISTENCY IN CHORD SEQUENCES

Multiple chords in a sequence do not follow each other randomly, but exhibit strong temporal links. Typical chord patterns have emerged throughout history, which have been studied by scholars such that expert knowledge about them is available [58]. ACR systems have been trying to incorporate this knowledge, initially with expert-based approaches, nowadays driven by data.

Implementation-wise, chord sequence consistency is mostly dealt with together with duration modelling of a single chord, as discussed in Section 4. In HMM-based approaches, the off-diagonal elements of the transition matrix determine where a chord change will lead to, whereas the diagonal elements influence when it takes place. The sources of knowledge for chord change information are different from the ones used for duration though. The doubly nested circle of fifths has been used frequently [2, 49] (or abused, as it is a model for key similarity) as well as other expert theories [39, 54].

The chord change probabilities of those probabilistic models have also been determined through corpus analysis [43, 54], before deep learning approaches became popular [31, 40, 61, 64, 68]. Typical for deep learning is that chord changes can be handled together with chord duration and models (problems 2, 3 and 4) by a single network [40, 64] to maximally exploit their mutual information. Nonetheless, some of the proposed approaches find it beneficial to handle the problems separately [32], in a way reminiscent of earlier knowledge-based systems.

Taking into account chords beyond the directly adjacent ones remains a challenge. A standard HMM can only model preceding chords indirectly, through chaining bigrams, so increasing the Markov order to trigrams and 4-grams has been tried [26], but their overall improvement remained small. As for deep learning techniques, the typical receptive field of feedforward neural networks is too small (1.5 s for [30]) to contain multiple chord changes and learn long-term dependencies. Recurrent neural networks have the theoretical advantage that all previous and/or future frames can be remembered, but vanishing gradient problems restrict their long-term memory in practice [1]. Attention mechanisms, which are used in machine translating to remember the context of long phrases, are potential solutions to this problem. The Transformer architecture [63] is one candidate for future exploration.

Since long-term chord dependencies are so hard to take into account with the aforementioned techniques, a couple of alternatives have been proposed that rely on musical structure. Feature representations of a repeated section have been averaged in order to make them more stable [11, 37] and that average has then been used for all instances of that section. A more probabilistic version of this idea has been explored in a statistical-relational frame-

work [51]. While these techniques do not exactly use long term dependencies to improve chord recognition, at least they ensure that repeated sections are consistent.

6. PROBLEM 5: EXPLOITING RELATIONSHIPS WITH RELATED MUSICAL CONCEPTS

Chords are just one way of describing musical content. Other musical concepts such as key, genre, bass or melody describe different aspects of the same piece of music. Several research efforts have incorporated ACR in systems that recognise multiple musical concepts, jointly or in sequence, to exploit the mutual information between them. We've already discussed the use of beat information to address Problem 3, so this section focusses on other concepts.

Practically, we see that probabilistic graphical models such as HMMs [35, 54] or dynamic Bayesian networks (DBNs) [39] have been the dominant approach so far to integrate related concepts into ACR. Their inherent modularity is exploited to decompose the probabilistic relations between concepts into more comprehensive factors.

Arguably the musical concept most related to chords is the musical key, as the latter also describes the harmony in a music piece, albeit on a longer temporal scale, i.e. a key spans multiple chords in sequence. Assigning different probabilities to all combinations of keys and chords is therefore a way to exploit their relationship. These probabilities can be derived from musical knowledge [39, 54] or data-driven [26, 35, 54]. One advantage of extracting keys with chords is that the chord sequence can be expressed relative to the key, in a representation close to functional harmony analysis, which has been shown to have a higher information density [58]. The required keys to make this possible can be derived prior to [26] or jointly with chord recognition [35, 54]. However, it has not been proven that using key-independent relative chord representations lead to improvements in actual chord recognition.

Downbeat is also related to chords in the sense that chords are more likely to change on downbeats than on other beats. This link has been exploited by making chord transition probabilities depend on the metric position of a beat in a measure. One example is a joint downbeat-chord system [50] that can deal with different time signatures and added or deleted beats. Another approach included beat-dependent chord transitions as part of a larger system that involves chord inversion and key as well [39], but which is limited to the 4/4 time signature. In both cases, the probabilities were determined by expert knowledge.

Other relationships between musical concepts can be exploited in a similar way. Bass notes, for instance, can be strongly indicative of the chord being played. They have the advantage of being comparatively easy to identify in a spectrum and are used as such to inform ACR [62, 67]. A joint key-chord-structure system has been proposed [53] based on the hypothesis that certain chord sequences are more likely at the start or end of high-level structures such as chorus or verse. Finally, different genres have different idiomatic chord sequences, so genre-dependent context modelling has also been examined [34, 44].

None of the recent deep-learning approaches have involved other musical concepts so far. In theory, it would be possible to identify the key segments of a piece beforehand and then transpose all segments of all pieces into one single key, instead of augmenting the feature representation as in [22, 40]. However, just like feeding beat-synchronous representations into a network, this would suffer from errors in the preceding key recognition step. Most likely a better solution would be to create specific networks that recognise chords together with any of the discussed concepts in a multi-task learning process such as [65]. Unfortunately, data annotated with multiple concepts is even less available than data that is annotated with just chords.

7. PROBLEM 6: HANDLING AMBIGUITY AND SUBJECTIVITY

Fujishima's system was tested using a strongly controlled setup. His dataset consisted mostly of clean, well-defined chords produced by an electronic keyboard using three different presets. He furthermore tailored his settings to these specific sonorities, but noticed that these settings didn't translate well to real music. Indeed, when we want to apply chord recognition to real music, the situation is very different from a controlled environment.

An enormous variety of music exists that may not even contain chords. Music can be monophonic, from a musical tradition that doesn't know the concept, or polyphonic without containing chords (e.g. a fugue). Even if chords are present in a music signal, it can also contain many sources – such as untuned percussion – that disturb the perception of a chord in a listener, human or mechanical. Furthermore, what exactly contributes to a chord is also ill-defined. The distinction between chord sequence and melodic line is not clear cut and can be a matter of opinion. Also the granularity of a chord sequence, for instance whether fast approach chords or anacrusis (pickup) chords are transcribed, depends on the user or use-case. A final cause of ambivalence is when chords are only implied, not audible as such, as with arpeggiated chords for instance.

Because all of these reasons human annotators aren't in total agreement when it comes to transcribing chords. The reported agreement on the root of a chord lies between 76% [29] (4 annotators) and 94% [12] (2 annotators) on average, but large outliers towards the bottom can appear in individual files [23, 45]. When comparing algorithmic output to a single human reference output, it is therefore hard to tell if any disagreements are valid alternative interpretations or outright errors. Although this phenomenon has been diagnosed and quantified [23, 45] multiple times already, final solutions are yet to emerge. The availability of chord datasets with multiple annotations, such as [45] (20 songs annotated by 5 persons) and [29] (50 songs annotated by 4 persons) is certainly a first step on the way. That former dataset has been used to learn the idiosyncrasies of different annotators and create personalised algorithmic output tailored to their preferences [28]. While it does provide a means of personalisation, it requires example transcriptions for each user, which might not be avail-

able. Furthermore multiple viewpoint annotations do not answer the question of whether an algorithmic output differing from all N viewpoints is plain wrong or the $N+1$th valid viewpoint. A different way of evaluating will be necessary for this, but it will probably be hard to scale. Asking human experts whether an automatic transcription is a good chord analysis (even if not exactly their own) would work, but would also mean that every different setting of an algorithm needs to be evaluated manually. Even by crowdsourcing the process, it would be impossible to test multiple parameterisations of a system.

8. PROBLEM 7: CHORD VOCABULARY AND ASSOCIATED BALANCE PROBLEMS

Fujishima's system was able to distinguish no less than 27 different chord types, but subsequent systems quickly reduced this number to two [2] or four [21] types of triad. Apart from [39], small vocabulary sizes remained the defacto standard until the last few years, with an upward trend that seems more definitive this time [14, 40].

The necessity of imposing an algorithmic chord vocabulary stems directly from the choice of treating chord recognition as a classification problem. This design choice is extremely wide-spread, but is not inherently part of the task. Chord class-free approaches can be envisaged, such as labelling sets of active chromas, where the complexity is then shifted towards determining when a chroma is active (itself potentially a classification problem). A vocabulary-free model of chords such as proposed in [66] can be used to label the chroma sets, but only the model in isolation has been tested so far, not as part of a full ACR system.

A particularity to increasing the classes in ACR is that the distinction between them becomes smaller the more classes are added. Especially when triads and tetrads (4-chroma chords) are mixed, because the set of chromas in tetrads are a superset of the chromas in their associated triad. Commonly used flat classification approaches assume disjoint categories, and therefore aren't an optimal match for the problem [23]. A form of hierarchical or alternatively multi-stage classification [64] is one way to address this problem, but these need to be explored further in the future. Different ways of representing target labels to make classes more orthogonal [7, 40, 64] constitute another type of solution, one in which standard flat classification algorithms can continue being used. Multiple auxiliary labels are introduced in this approach, which we called *target label engineering* in Section 3. Alternatively, multiple distances between chords have been used as targets [7]. These chord label representations constitute a modern take on introducing musical knowledge into deep learning. Where feature engineering tried to shape the audio input into features that were maximally discriminative, target label engineering aims to do the same working back from the label output.

A further complication when considering chord labels as separate classes is that their frequency of occurrence is strongly unbalanced. For instance, the five most common chord types account for over 80% of popular music datasets [6, 64]. This imbalance in chord distribution affects both training data-driven ACR systems and evaluation. For training, new strategies need to be developed to avoid overemphasising the most frequent classes [16]. In evaluation, improvements on rare chords are barely visible when using a metric that reports the percentage of time the correct chord is found. Discussing performance on multiple levels, with rarer chords separated, is necessary to get more insight into algorithm performance [55].

Finally, the chord imbalance itself differs from dataset to dataset. Genre, instrumentation, cultural origin, key and chord distributions are all linked, and are manifested as a skew towards certain roots and chord types, as well as variance in timbre. Since the majority of available annotated data consists of anglophonic pop music, its representativeness is questionable. Until current algorithms are cross-checked with new datasets containing e.g. Latin, jazz and metal, their general applicability remains unproven.

9. CONCLUSIONS

In this paper, we discussed 20 years of research on automatic chord recognition, starting with Fujishima's paper [17]. Even though modern ACR systems differ strongly in their proposed technical solutions, we find that his initial system was very effective at identifying the problems that arise during the creation of ACR systems. We therefore compared past and recent solutions thematically according to these problems, with the intention of inspiring future work on this topic.

We note a tendency towards tackling the different problems in ACR as a single integrated approach, in contrast to the compartmentalised strategies of the early years. This evolution follows the move from knowledge-driven to data-driven approaches. The lack of available training data in the early years called for knowledge-based systems, which in turn required systems to be broken down into smaller components for them to remain comprehensible. Each component usually dealt with one sub-problem in isolation. Early data-driven approaches replaced these knowledge-based components with learnt ones, while retaining the modular structure. The arrival of deep learning permitted to replace all these components by a single system that is better at exploiting the interactions between the ACR problems. Nonetheless, some researchers choose to keep the modularised approach, to increase interpretability or to employ specific training data or procedures.

A consequence of moving towards integrated systems, is that the comparison between them becomes harder. No general blueprint of a DL-ACR system can be given, as many approaches compete, and therefore it is difficult to translate findings of one system to the other or to combine parts of two systems into one. Without a doubt, more innovations in deep learning will make their way to ACR, which will keep the field moving fast for the foreseeable future. It is encouraging to see that the specific characteristics of music are starting to show up in the deep learning approaches, with musical knowledge guiding the training procedure and architecture.

10. ACKNOWLEDGEMENTS

This work has been funded by the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/L019981/1.

11. REFERENCES

[1] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint*, 2018.

[2] Juan Pablo Bello and Jeremy Pickens. A robust mid-level representation for harmonic content in music signals. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, pages 304–311, 2005.

[3] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio chord recognition with Recurrent Neural Networks. In *Proceedings of the 14th Conference of the International Society for Music Information Retrieval (ISMIR)*, 2013.

[4] Judith C. Brown. Calculation of a constant Q spectral transform. *Journal of the Acoustical Society of America*, 89(1):425–434, January 1991.

[5] John Ashley Burgoyne, Laurent Pugin, Corey Kereliuk, and Ichiro Fujinaga. A cross-validated study of modelling strategies for automatic chord recognition in audio. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, pages 251–254, 2007.

[6] John Ashley Burgoyne, Jonathan Wild, and Ichiro Fujinaga. An expert ground-truth set for audio chord recognition and music analysis. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, pages 633–638, 2011.

[7] Tristan Carsault, Jérôme Nika, and Philippe Esling. Using musical relationships between chord labels in automatic chord extraction tasks. In *Proceedings of the 19th Conference of the International Society for Music Information Retrieval (ISMIR)*, 2018.

[8] Ruofeng Chen, Weibin Shen, Ajay Srinivasamurthy, and Parag Chordia. Chord recognition using duration-explicit hidden Markov models. In *Proceedings of the 13th International Conference on Music Information Retrieval (ISMIR)*, pages 445–450, 2012.

[9] Taemin Cho. *Improved techniques for automatic chord recognition from music audio signals*. PhD thesis, New York University, 2013.

[10] Taemin Cho and Juan P. Bello. On the relative importance of individual components of chord recognition systems. *IEEE Transactions on Audio, Speech and Language Processing*, 22(2):477–492, February 2014.

[11] Taemin Cho and Juan Pablo Bello. A feature smoothing method for chord recognition using recurrence plots. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, pages 651–656, 2011.

[12] Trevor de Clercq and David Temperley. A corpus analysis of rock harmony. *Popular Music*, 30(1):47–70, January 2011.

[13] Alessio Degani, Marco Dalai, Riccardo Leonardi, and Pierangelo Migliorati. Harmonic change detection for musical chords segmentation. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2015. conference-date: June 29 2015-July 3 2015 conference-venue: Turin, Italy.

[14] Junqi Deng and Yu-Kwong Kwok. A hybrid Gaussian-HMM-deep learning approach for automatic chord estimation with very large vocabulary. In *Proceedings of the 17th Conference of the International Society for Music Information Retrieval (ISMIR)*, pages 812–818, 2016.

[15] Junqi Deng and Yu-Kwong Kwok. Large vocabulary automatic chord estimation with an even chance training scheme. In *Proceedings of the 18th Conference of the International Society for Music Information Retrieval (ISMIR)*, 2017.

[16] Junqi Deng and Yu-Kwong Kwok. Large vocabulary automatic chord estimation using bidirectional long short-term memory recurrent neural network with even chance training. *Journal of New Music Research*, 47(1):53–67, 2018.

[17] Takuya Fujishima. Realtime chord recognition of musical sound: a system using Common Lisp Music. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 464–467, Ann Arbor, MI, USA, 1999. MPublishing, University of Michigan Library.

[18] Emilia Gómez. *Tonal description of music audio signals*. PhD thesis, Universitat Pompeu Fabra, 2006.

[19] Emilia Gómez. Tonal description of polyphonic audio for music content processing. *INFORMS Journal on Computing, Special Cluster on Computation in Music*, 18(3):294–304, Summer 2006.

[20] Christopher Harte, Mark Sandler, and Martin Gasser. Detecting harmonic change in musical audio. In *Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia (AMCMM)*, pages 21–26, New York, NY, USA, 2006. ACM.

[21] Christopher A. Harte and Mark B. Sandler. Automatic chord identification using a quantised chromagram. In *Proceedings of the 118th Convention of the Audio Engineering Society*, Barcelona, Spain, May 28–31 2005.

[22] Eric J. Humphrey and Juan P. Bello. Rethinking automatic chord recognition with Convolutional Neural Networks. In *Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 357–362, Boca Raton, FL, 12–15 December 2012.

[23] Eric J. Humphrey and Juan P. Bello. Four timely insights on automatic chord estimation. In *Proceedings of the 16th Conference of the International Society for Music Information Retrieval (ISMIR)*, pages 673–679, 2015.

[24] Eric J. Humphrey, Taemin Cho, and Juan P. Bello. Learning a robust Tonnetz-space transform for automatic chord recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 453–456. IEEE, 2012.

[25] Junyan Jiang, Wei Li, and Yiming Wu. Extended abstract for MIREX 2017 submission: Chord recognition using random forest model. In *Proceedings of the Music Information Retrieval Evaluation Exchange (MIREX)*, 2017.

[26] Maksim Khadkevich and Maurizio Omologo. Use of hidden Markov models and factored language models for automatic chord recognition. In *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR)*, pages 561–566, 2009.

[27] Maksim Khadkevich and Maurizio Omologo. Reassigned spectrum-based feature extraction for GMM-based automatic chord recognition. *EURASIP Journal on Audio, Speech, and Music Processing*, 2013(15):1–12, June 2013.

[28] Hendrik Vincent Koops, W. Bas de Haas, Jeroen Bransen, and Anja Volk. Chord label personalization through deep learning of integrated harmonic interval-based representations. In *Proceedings of the First International Workshop on Deep Learning for Music*, pages 19–25, 2017.

[29] Hendrik Vincent Koops, W. Bas de Haas, John Ashley Burgoyne, Jeroen Bransen, Anna Kent-Muller, and Anja Volk. Annotator subjectivity in harmony annotations of popular music. *Journal of New Music Research*, pages 232–252, 2019.

[30] Filip Korzeniowski and Gerhard Widmer. Feature learning for chord recognition: The deep chroma extractor. In *Proceedings of the 17th Conference of the International Society for Music Information Retrieval (ISMIR)*, 2016.

[31] Filip Korzeniowski and Gerhard Widmer. A fully convolutional deep auditory model for musical chord recognition. In *Proceedings of the IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2016.

[32] Filip Korzeniowski and Gerhard Widmer. Improved chord recognition by combining duration and harmonic language models. In Emilia Gómez, Xiao Hu, Eric Humphrey, and Emmanouil Benetos, editors, *Proceedings of the 19th Conference of the International Society for Music Information Retrieval (ISMIR)*, pages 10–17, 2018.

[33] Kyogu Lee. Automatic chord recognition using enhanced pitch class profile. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 306–313, Ann Arbor, MI, USA, 2006. MPublishing, University of Michigan Library.

[34] Kyogu Lee. A system for automatic chord transcription using genre-specific hidden Markov models. In Nozha Boujemaa, Marcin Detyniecki, and Andreas Nürnberger, editors, *Adaptive Multimedia Retrieval: Retrieval, User, and Semantics*, volume 4918 of *Lecture Notes in Computer Science*, pages 134–146. Springer Berlin Heidelberg, 2008.

[35] Kyogu Lee and Malcolm Slaney. Acoustic chord transcription and key extraction from audio using key-dependent HMMs trained on synthesized audio. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):291–301, February 2008.

[36] Namunu C. Maddage, Changsheng Xu, Mohan S. Kankanhalli, and Xi Shao. Content-based music structure analysis with applications to music semantics understanding. In *Proceedings of the 12th ACM International Conference on Multimedia (ACM MM)*, pages 112–119. ACM, 10 October 2004.

[37] Matthias Mauch and Simon Dixon. Using musical structure to enhance automatic chord transcription. In *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR)*, pages 231–236, 2009.

[38] Matthias Mauch and Simon Dixon. Approximate note transcription for the improved identification of difficult chords. In *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, pages 135–140, 2010.

[39] Matthias Mauch and Simon Dixon. Simultaneous estimation of chords and musical context from audio. *IEEE Transactions on Audio, Speech and Language Processing*, 18(6):1280–1289, August 2010.

[40] Brian McFee and Juan Pablo Bello. Structured training for large-vocabulary chord recognition. In *Proceedings of the 18th Conference of the International Society for Music Information Retrieval (ISMIR)*, 2017.

[41] Joshua Morman and Lawrence Rabiner. A system for the automatic segmentation and classification of chord sequences. In *Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia (AMCMM)*, pages 1–10. ACM, 27 October 2006.

[42] Meinard Müller and Sebastian Ewert. Towards timbre-invariant audio features for harmony-based music. *IEEE Transactions on Audio, Speech and Language Processing*, 18(3):649–662, March 2010. ISSN: 1558-7916.

[43] Yizhao Ni, Matt McVicar, Raúl Santos-Rodríguez, and Tijl De Bie. An end-to-end machine learning system for harmonic analysis of music. *IEEE Transactions on Audio, Speech and Language Processing*, 20(6):1771–1783, August 2012.

[44] Yizhao Ni, Matt McVicar, Raúl Santos-Rodríguez, and Tijl De Bie. Using hyper-genre training to explore genre information for automatic chord estimation. In *Proceedings of the 13th International Conference on Music Information Retrieval (ISMIR)*, pages 109–114, 2012.

[45] Yizhao Ni, Matt McVicar, Raúl Santos-Rodríguez, and Tijl De Bie. Understanding effects of subjectivity in measuring chord estimation accuracy. *IEEE Transactions on Audio, Speech and Language Processing*, 21(12):2607–2615, December 2013.

[46] Ken O'Hanlon and Mark B. Sandler. Comparing CQT and reassignment based chroma features for template-based automatic chord recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 860–864, 2019.

[47] Laurent Oudre, Cédric Févotte, and Yves Grenier. Probabilistic template-based chord recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 19(8):2249 – 2259, November 2011.

[48] Laurent Oudre, Yves Grenier, and Cédric Févotte. Chord recognition by fitting rescaled chroma vectors to chord templates. *IEEE Transactions on Audio, Speech and Language Processing*, 19(7):2222 – 2233, September 2011.

[49] Hélène Papadopoulos and Geoffroy Peeters. Large-scale study of chord estimation algorithms based on chroma representation and HMM. In *Proceedings of the International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 53–60, Bordeaux, France, June 25–27 2007.

[50] Hélène Papadopoulos and Geoffroy Peeters. Joint estimation of chords and downbeats from an audio signal. *IEEE Transactions on Audio, Speech and Language Processing*, 19(1):138–152, January 2011.

[51] Hélène Papadopoulos and George Tzanetakis. Models for music analysis from a Markov logic networks perspective. *IEEE Transactions on Audio, Speech and Language Processing*, 25:19–34, 2017.

[52] Johan Pauwels. *Exploiting prior knowledge during automatic key and chord estimation from musical audio*. PhD thesis, Faculty of Engineering and Architecture, 2016.

[53] Johan Pauwels, Florian Kaiser, and Geoffroy Peeters. Combining harmony-based and novelty-based approaches for structural segmentation. In *Proceedings of the 14th Conference of the International Society for Music Information Retrieval (ISMIR)*, pages 138–143, Curitiba, Brazil, 2013.

[54] Johan Pauwels and Jean-Pierre Martens. Combining musicological knowledge about chords and keys in a simultaneous chord and local key estimation system. *Journal of New Music Research*, 43(3):318–330, 2014.

[55] Johan Pauwels and Geoffroy Peeters. Evaluating automatically estimated chord sequences. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 749–753, 2013.

[56] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik M. Schmidt, Andreas F. Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. In *Proceedings of the 19th Conference of the International Society for Music Information Retrieval (ISMIR)*, pages 636–644, 2018.

[57] Zhongyang Rao, Xin Guan, and Jianfu Teng. Chord recognition based on temporal correlation support vector machine. *Applied Sciences*, 6(5):157, 2016.

[58] Ricardo Scholz, Emmanuel Vincent, and Frédéric Bimbot. Robust modeling of musical chord sequences using probabilistic n-grams. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 53–56, 2009.

[59] Alexander Sheh and Daniel P.W. Ellis. Chord segmentation and recognition using EM-trained hidden Markov models. In *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR)*, pages 183–189, 2003.

[60] Arun Shenoy and Ye Wang. Key, chord, and rhythm tracking of popular music recordings. *Computer Music Journal*, 29(3):75–86, 2005.

[61] Siddharth Sigtia, Nicolas Boulanger-Lewandowski, and Simon Dixon. Audio chord recognition with a hybrid recurrent neural network. In *Proceedings of the 16th Conference of the International Society for Music Information Retrieval (ISMIR)*, pages 127–133, 2015.

[62] Kouhei Sumi, Katsutoshi Itoyama, Kazuyoshi Yoshii, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. Automatic chord recognition based on probabilistic integration of chord transition and bass pitch estimation. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, pages 39–44, 2008.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint*, 2017.

[64] Yiming Wu and Wei Li. Automatic audio chord recognition with MIDI-trained deep feature and BLSTM-CRF sequence decoding model. *IEEE Transactions on Audio, Speech and Language Processing*, 27(2):355–366, February 2019.

[65] Mu-Heng Yang, Li Su, and Yi-Hsuan Yang. Highlighting root notes in chord recognition using cepstral features and multi-task learning. In *Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1–8. IEEE, 2016.

[66] Kazuyoshi Yoshii and Masataka Goto. A vocabulary-free infinity-gram model for nonparametric Bayesian chord progression analysis. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, pages 645–650, 2011.

[67] Takuya Yoshioka, Tetsuro Kitahara, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. Automatic chord transcription with concurrent recognition of chord symbols and boundaries. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, pages 100–105, 2004.

[68] Xinquan Zhou and Alexander Lerch. Chord detection using deep learning. In *Proceedings of the 16th Conference of the International Society for Music Information Retrieval (ISMIR)*, pages 52–58, 2015.

Session A

ZERO-SHOT LEARNING FOR AUDIO-BASED MUSIC CLASSIFICATION AND TAGGING

Jeong Choi[1*] **Jongpil Lee**[1*] **Jiyoung Park**[2] **Juhan Nam**[1]
[1] Graduate School of Culture Technology, KAIST
[2] NAVER Corp.
{jeong.choi, richter, juhannam}@kaist.ac.kr, j.y.park@navercorp.com
* Equally contributing authors.

ABSTRACT

Audio-based music classification and tagging is typically based on categorical supervised learning with a fixed set of labels. This intrinsically cannot handle unseen labels such as newly added music genres or semantic words that users arbitrarily choose for music retrieval. Zero-shot learning can address this problem by leveraging an additional semantic space of labels where side information about the labels is used to unveil the relationship between each other. In this work, we investigate the zero-shot learning in the music domain and organize two different setups of side information. One is using human-labeled attribute information based on Free Music Archive and OpenMIC-2018 datasets. The other is using general word semantic information based on Million Song Dataset and Last.fm tag annotations. Considering a music track is usually multi-labeled in music classification and tagging datasets, we also propose a data split scheme and associated evaluation settings for the multi-label zero-shot learning. Finally, we report experimental results and discuss the effectiveness and new possibilities of zero-shot learning in the music domain.

1. INTRODUCTION

Audio-based music classification and tagging is a task that predicts musical categories or attributes such as genre, mood, instruments and other song quality from music tracks. Current state-of-the-arts algorithms are based on supervised learning of deep convolutional neural networks that directly predict the labels in the output layer [18]. That is, the neural networks are trained to minimize the prediction errors with regards to the labels. The prediction results can be used to automatically annotate music tracks or retrieve music tracks using the labels as a query word [32]. By the nature of the setting in the supervised learning, however, the approach allows only a fixed set of word labels in the annotation and retrieval.

Zero-shot learning is a learning paradigm that can overcome this limitation and enables the trained model to predict unseen labels [12, 20]. For example, it allows the model to predict newly added music genres after the training or retrieve songs using a query word that users arbitrarily choose. This is possible by utilizing side information that derives a separate semantic space from labels. For example, the side information can be musical instrument annotation vectors of music genres or word embedding learned from sentences. The zero-shot learning approach conducts supervised learning between the semantic space and audio feature space. Once the mapping between two embedding spaces is learned, the model can predict unseen labels. Figure 1 illustrates the concept of zero-shot learning applied to music classification and tagging.

The zero-shot learning approach was previously applied to music data [28]. However, they focused on evaluating a specific semantic embedding method that works for general multimedia data rather than delving into zero-shot learning in the music domain. In this work, we carefully investigate how the concept of zero-shot learning can be properly applied to audio-based music classification and tagging. Specifically, we designed two settings of side information. One is using human-labeled attribute information and the other is using general word semantic information. Also, considering a music track is usually multi-labeled in music classification and tagging datasets, we propose a data split scheme that yields a comprehensive list of combinations for seen or unseen audio and labels, and evaluate them in the various settings. Through the experiments, we show the effectiveness and new possibilities of zero-shot learning in the music domain.

2. BACKGROUND: ZERO-SHOT LEARNING

Zero-shot learning has been studied mainly for object recognition in the field of computer vision [34]. They have attempted to build a model that can recognize novel visual categories without any associated training samples by employing a joint embedding space of both images and their class labels [34]. It was originally inspired by human's ability to recognize objects without seeing training examples or even create new categories dynamically based on semantic analysis [9]. What enables this semantic exploitation of unseen images is "information transfer" that

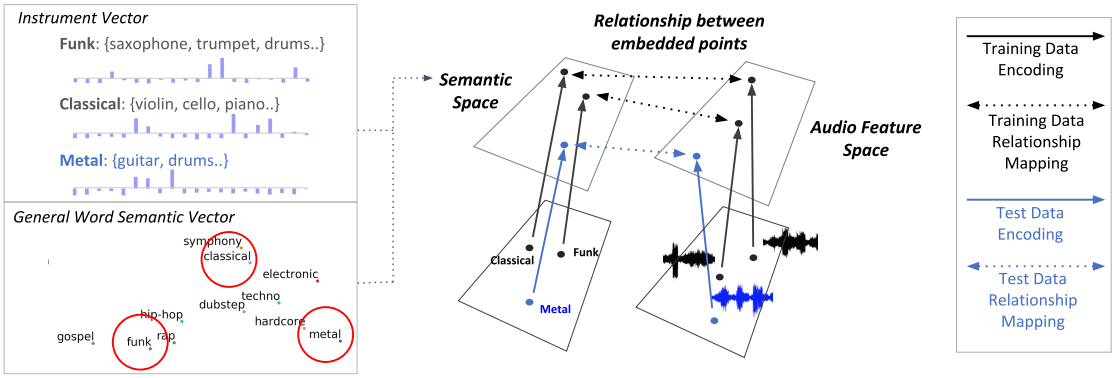


Figure 1. Overview of zero-shot learning concept applied to music classification and tagging.

applies the knowledge learned in an auxiliary domain to the main targeted task. Thus, it aims to explore how seen and unseen images are semantically related based on the side information.

The types of side information can be largely divided into two categories [9, 34, 37]. One is human-annotated attributes of the labels. In the computer vision community, there are publicly available human-annotated attribute datasets such as AWA (Animals With Attributes) [12] and CUB (Caltech-UCSD Birds-200) [35]. For example, AWA offers images of 50 animals annotated on 85 attributes associated with animals characteristics such as "furry" or "has tail". These examples may be equivalent to music genre classification where corresponding attributes are musical instruments. These attributes can be used as binary output to train a classifier and infer unseen class based on similarity measure [2, 12] or to learn their relative strength [21, 30]. When such explicit attributes data are not available, the semantic attributes can be learned with hierarchy of classes or other relationships [14,25,27]. In general, this attribute-based approach has advantages in interpretability but requires well-defined attributes and annotation data.

The other type of side information is a general word semantic space learned from different resources. A commonly used choice is neural language models such as Word2Vec [1, 8, 15, 16, 19, 36] and GloVe [1, 23, 36]. Another choice is learning relational semantic embeddings using pre-defined lexical hierarchy such as WordNet [17,27]. These general semantic spaces have an advantage in that they have a large set of words in the vocabulary to predict unseen labels. However, if the target labels have a specific context, the general semantic space may fail to capture it [28].

3. DATA SPLIT SCHEME FOR MULTI-LABEL ZERO-SHOT LEARNING

Many of music classification and tagging datasets have multiple labels to annotate music tracks. However, zero-shot learning in the image domain has been treated primarily as a single-label problem, although a few studies have attempted to address it in multi-label classification [10,14]. An important difference between single-label and multi-label zero-shot learning is data split scheme for training and test. In general, in zero-shot learning, both data and labels are split into seen and unseen sets. In the single-label setup, the label split can automatically divide the dataset into training and test sets (Figure 2 (a)). In the multi-label setup, however, specifying reasonable instance or label splits is not straightforward.

3.1 Previous Approaches

Most of previous works on multi-label zero-shot learning are conducted the instance-first split [26]. They first split instances into train and test, and only used seen labels for training as shown in the left of Figure 2 (b). In this case, some of the instances in the train set can have positive annotations for unseen labels. As an alternative, the label-first split was proposed in [33]. They first split labels into seen and unseen groups, and select training instances to have no positive annotation for unseen labels and select test instances to have at least positive annotation on unseen labels as shown in the right of Figure 2 (b). However, in this case, due to the nature of multi-label data, too many instances can be assigned to the test set.

Meanwhile, it is an important issue to determine which split in instance (train and test) or label (seen and unseen) should be evaluated in measuring zero-shot learning performance. A generalized zero-shot learning evaluation setting is proposed in [37]. It includes both seen and unseen labels at test time to examine more natural annotation performance compared to use only unseen labels at test time. In multi-label zero-shot learning, however, the data split and evaluation settings are still not clear and there is no agreed consensus yet.

3.2 Proposed Data Split Scheme

We propose a data split scheme and evaluation settings for multi-label zero-shot learning to measure the performance in more refined and various settings. The proposed data split is shown in Figure 2 (c). We first divide labels into seen (X) and unseen (Y) groups and then split instances into three groups. The first subset (A) of instances are labeled with at least one from seen labels and not labeled with any of unseen labels. The second subset (B) of in-

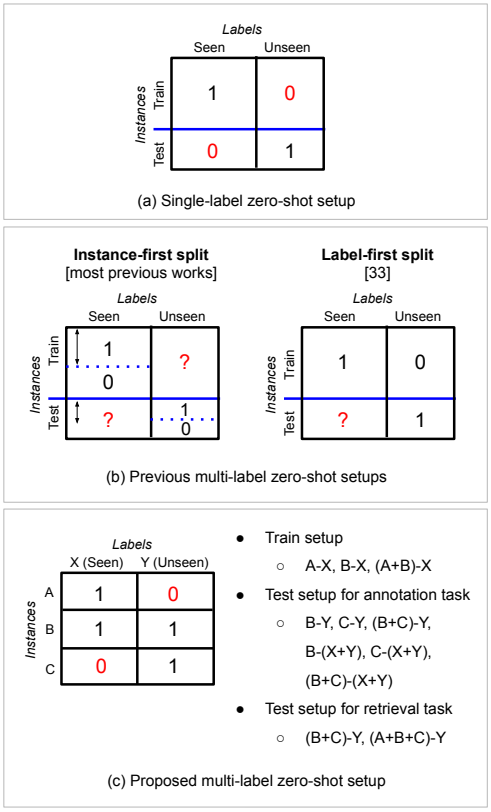


Figure 2. Data split for zero-shot learning

stances are labeled with at least one from each of seen and unseen labels. Lastly, the third subset (C) of instances are only labeled with at least one from unseen labels. Then we create the three setups (including train setup, test setup for annotation task, and test setup for retrieval task) with a combination of instance subsets (A, B, and C) and label subsets (X and Y) as described in Figure 2 (c).

In this configuration, the label-first split can be obtained when the training set is A-X (a subset where instances are from A and labels are from X) and the test set is (B+C)-Y (a subset where instances are from (B+C) and labels are from Y). Also, the generalized zero-shot learning evaluation setting can be expressed to include not only the Y area but also the (X+Y) area at test time. We therefore extend the split and evaluation settings more comprehensively, allowing for many aspects of multi-label zero-shot learning to be considered.

We take the train setup from A-X, B-X, and (A+B)-X. The instances in A-X only contain annotations on seen labels and the instances in B-X contain annotations on both seen and unseen labels. By distinguishing these two areas, we expect to see the difference in model learning by using instances with and without annotations in the unseen labels in multi-label zero-shot learning.

The test setup for the annotation task is made up of combinations of B, C, or (B+C) with Y. In addition, combinations of B, C, or (B+C) with (X+Y) can be also considered to measure generalized zero-shot learning performance on the annotation task as explained in the Section 3. The test setup for retrieval task is composed of (B+C)-Y and (A+B+C)-Y. We can regard (A+B+C)-Y as a case of generalized zero-shot learning evaluation setting because the retrieval is performed not only on the instances of unseen labels ((B+C) split) but also on instances of seen labels (A split). The C-Y area cannot be formed in the retrieval evaluation. The reason for this is that once we split labels into seen and unseen and then assign overlapping instances to the B area, we may not guarantee that all the unseen labels have at least one positive activation on the instances in the C area.

4. MODEL

4.1 Deep Embedding Model

Zero-shot learning is primarily performed by a compatibility function that maps multimedia embedding and semantic embedding. The joint embedding methods can be categorized into learning linear compatibility, nonlinear compatibility, intermediate attribute classifier, and their hybrid [37]. In this work, we focus on learning nonlinear compatibility. The model takes audio mel-spectrogram as input rather than audio embedding extracted from pre-trained model. A convolutional neural network (CNN) module for audio is learned directly with semantic embedding from ground truth annotations. Figure 3 illustrates the model architecture. The model takes audio from one module and randomly selected a positive word and a negative word from the ground truth annotations of the audio via the semantic vector lookup table. Following the previous works [8, 22], the loss function is chosen as a max-margin hinge loss as below:

$$\mathrm{L}(A, W) = \max\left[0, \Delta - Rel\left(A, W^{+}\right) + Rel\left(A, W^{-}\right)\right]$$

where Δ is the margin, W^{+} denotes the label with positive annotation for the audio input, and W^{-} denotes the label with negative annotation. The cosine similarity of the last hidden layer of the audio module and semantic module is used as a relevance score [22]:

$$Rel(A, W) = Similarity_{cosine}\left(y_A, y_W\right) = \frac{y_A^T y_W}{|y_A||y_W|}$$

where y_A and y_W denote the output of the last hidden layer for the audio module and the semantic module, respectively.

The mel-spectrogram based audio CNN module is constructed with four 1D convolutional layers with a 2D filter [5, 7, 13, 24]. Each layer is followed by a rectified linear unit (ReLU) activation and a max pooling layer. We added a convolutional layer and an average pooling layer on top of them to construct a fixed-size audio embedding vector compatible to the semantic module. The semantic module is constructed by adding a fully connected layer over a semantic embedding (the output of semantic vector lookup table). In this case, the semantic embedding (or semantic vector lookup table) can be composed with both human-annotated attributes data or general word semantic space.

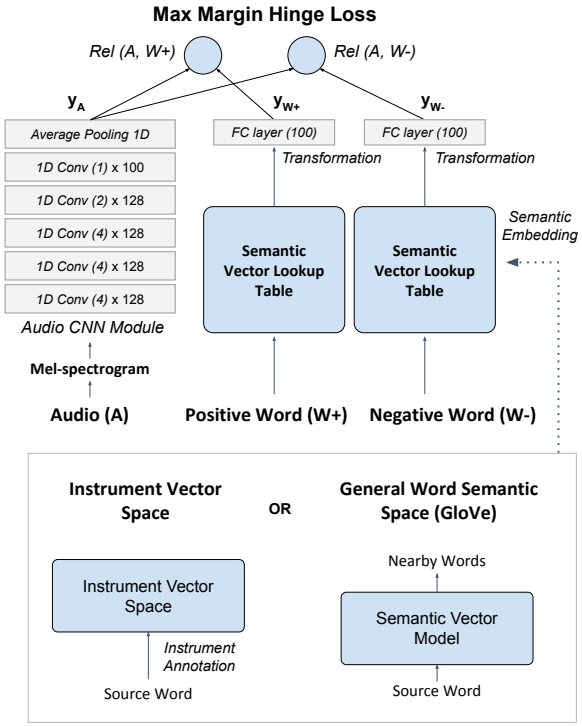


Figure 3. Deep embedding model for zero-shot learning using instrument vector space or general word semantic space.

4.2 Classification Model

The deep embedding model can be used for not only the zero-shot learning setting but also the conventional multi-label classification and tagging problem where we train and evaluate the model with only seen labels. Therefore, we construct a classification model and compare it with the deep embedding model to evaluate the learning capability of the deep embedding model. The classification model is basically the same as the audio module of the deep embedding model but the output is the binary representation of the labels. To this end, we added a fully-connected output layer with the size of seen labels on top of the average pooling layer of the audio module. We used the sigmoid activation and binary cross-entropy loss for the multi-label classification.

4.3 Training Details

We extract mel-spectrogram from audio with 128 mel bins, 1024 size FFT (Hanning window), and 512 size hop in 22,050 Hz sampling rate. We standardized the mel-spectrogram across all training data to have zero mean and unit variance. We randomly selected three seconds of audio chunk (130 frames) as an input size to the CNN module. We optimized the model using stochastic gradient descent with 0.9 Nesterov momentum, 0.001 learning rate, and $1e^{-6}$ learning rate decay for all models and datasets. Our system is implemented in Python 3.5.2, Keras 2.2.2, and Tensorflow-gpu 1.6.0 for the back-end of Keras. [1]

[1] The source code is available at `https://github.com/kunimi00/ZSL_music_tagging`

	Label Split			Instance Split			
Dataset	X (seen)	Y (unseen)	Total	A	B	C	Total
FMA	125	32	157	11606	6935	925	19466
MSD	900	226	1126	199385	188967	18057	406409

Table 1. Data split statistics.

In the test phase, we took the average of the fixed-size audio embedding vectors over a single music track to obtain a track-level embedding, and made the predictions by the distance between track-level audio embedding and tag embedding.

5. EXPERIMENTAL SETTINGS

We apply the proposed data split scheme and the deep embedding model to publicly available music classification and tagging datasets. We experiment with this in two settings of side information that we discussed in Section 2. For all the splits, we reserve 10% instances of train set as a validation set randomly. [2]

5.1 Experiment 1: Genre with Instrument Attributes

Musical instrument is one of the most important elements that determine music genre. Thus, recognizing instruments in a music track can be a strong cue to predict an unseen (or unheard) genre (assuming that the predictor learned the genre from some literature). For this experiment, we use Free Music Archive (FMA) [6] and OpenMIC-2018 datasets [11]. FMA contains audio files and genre annotations. OpenMIC-2018, which was originally designed for multiple instrument recognition, has 20 different instrument annotations to the audio files in FMA. We filtered the audio files to have both genre and instrument annotations. As a result, 19,466 audio with 157 genre labels and 20 instrument annotations are left. Following the proposed data split scheme, we randomly split 157 labels into 125 seen and 32 unseen ones. Then, three groups of audio instances (A, B, C) are created naturally. The statistics of audio and labels is described in Table 1.

The annotations on 20 different instruments in the OpenMIC-2018 dataset are labeled as a likelihood measure summarized from crowd-sourced annotations. They are regarded as positive if the likelihood value of an instrument annotation is larger than 0.5 and, otherwise, negative. The exact value of 0.5 means that it's unannotated. So, we treated positive and negative information independently and created 40 dimensional instrument vectors that can represent both positive and negative information. The genre to instrument attribute relationship is then constructed by accumulating 40 dimensional instrument vectors of the songs according to the genre labels. Finally, we standardized the instrument vectors to have zero mean and unit variance. This was used as the semantic vector lookup table in the learning model.

[2] All the data splits are available along with the source code.

Annotation Task		FMA					MSD				
Train	Test	AUC-i	MAP-i	P@1	P@5	P@10	AUC-i	MAP-i	P@1	P@5	P@10
A-X	B-Y	0.8736	0.5118	0.3465	0.4810	0.5008	0.8559	0.3391	0.3895	0.2944	0.3109
	C-Y	0.7917	0.3887	0.1563	0.0813	0.0643	0.8956	0.4536	0.3867	0.4278	0.4403
	(B+C)-Y	0.8640	0.4973	0.3333	0.4648	0.4855	0.8593	0.3491	0.3893	0.3060	0.3221
	B-(X+Y)	0.8834	0.3524	0.3725	0.2604	0.3056	0.9107	0.2664	0.4653	0.2735	0.2265
	C-(X+Y)	0.7958	0.1316	0.0162	0.0841	0.1089	0.9019	0.1594	0.0522	0.1292	0.1449
	(B+C)-(X+Y)	0.9207	0.4264	0.4018	0.3540	0.3932	0.9315	0.2842	0.3762	0.2645	0.2504
B-X	B-Y	0.8907	0.5633	0.4099	0.5371	0.5545	0.8679	0.3904	0.4803	0.3481	0.3623
	C-Y	0.8144	0.4008	0.2281	0.3556	0.3834	0.8926	0.4814	0.4379	0.4573	0.4674
	(B+C)-Y	0.8817	0.5442	0.3885	0.5158	0.5344	0.8700	0.3983	0.4766	0.3576	0.3715
	B-(X+Y)	0.9373	0.5142	0.5740	0.4206	0.4778	0.9217	0.2733	0.4462	0.2651	0.2283
	C-(X+Y)	0.8414	0.1886	0.0389	0.1494	0.1694	0.8941	0.1389	0.0666	0.1045	0.1231
	(B+C)-(X+Y)	0.8872	0.3606	0.3713	0.2881	0.3263	0.9276	0.2509	0.3272	0.2232	0.2147
(A+B)-X	B-Y	0.8864	0.5036	0.3090	0.4748	0.4951	0.8632	0.3563	0.4349	0.3124	0.3278
	C-Y	0.8286	0.4224	0.2530	0.3803	0.4064	0.8971	0.4836	0.4405	0.4591	0.4699
	(B+C)-Y	0.8796	0.4940	0.3024	0.4636	0.4846	0.8662	0.3674	0.4354	0.3252	0.3402
	B-(X+Y)	0.9118	0.4275	0.4784	0.3350	0.3813	0.9231	0.2977	0.5111	0.3094	0.2605
	C-(X+Y)	0.8276	0.1923	0.0530	0.1507	0.1738	0.9044	0.1920	0.1016	0.1629	0.1780
	(B+C)-(X+Y)	0.9073	0.3812	0.3770	0.3064	0.3445	0.9370	0.2984	0.3987	0.2808	0.2660

Table 2. Zero-Shot learning results for annotation task.

5.2 Experiment 2: General Word Semantic Space

The other type of side information is word embedding learned from a large-scale text dataset separately from music datasets. It can represent words as vectors in a semantic space. We adopted GloVe as a word embedding model [23]. Instead training it from scratch, we used a publicly available pre-trained GloVe model [3]. It consists of 300-dimensional vectors of 19 million vocabularies trained from documents in Common Crawl data. Since this can cover a large vocabulary of words, we used a music dataset with rich annotations, which is Million Song Dataset (MSD) with the Last.fm tag annotations [3]. From the full set of 498,035 tags in the Last.fm annotations, we filtered the tags that correspond to 2,000 genre/sub-genre classes contained in Tagtraum genre ontology [29]. We filtered the result (1800 tags) again into 1,126 tags after eliminating missing words in the dictionary of the pre-trained GloVe model. We used 406,409 audio instances annotated with the refined 1,126 tags. Following the proposed data split scheme, we randomly split 1,126 tags into 900 seen and 226 unseen ones, and organized three groups of audio instances (A, B and C). They are summarized in Table 1.

5.3 Evaluation Metrics

We used the area under the ROC curve averaged over instance (AUC-i), mean average precision over instance (MAP-i), and precision at K (P@K) as evaluation metrics for the annotation task. The retrieval task is evaluated using the area under the ROC curve averaged over label (AUC-l) and mean average precision over label (MAP-l).

6. RESULTS

6.1 Multi-label Zero-Shot Annotation

We compare the results of the combination of the proposed multi-label zero-shot learning split in the annotation task.

[3] The Common Crawl model was trained with 42B tokens containing 1.9M vocabulary. https://nlp.stanford.edu/projects/glove/

Data Split		FMA		MSD	
Train	Test	AUC-l	MAP-l	AUC-l	MAP-l
A-X	(B+C)-Y	0.6793	0.0904	0.6740	0.0279
	(A+B+C)-Y	0.6771	0.0392	0.6673	0.0149
B-X	(B+C)-Y	0.7194	0.1280	0.6907	0.0295
	(A+B+C)-Y	0.7236	0.0662	0.6843	0.0158
(A+B)-X	(B+C)-Y	0.7314	0.1170	0.6864	0.0310
	(A+B+C)-Y	0.7377	0.0518	0.6789	0.0172

Table 3. Zero-Shot learning results for retrieval task.

From Table 2, we can find that training with (A+B) or B instance set shows better performance than that with A in general. This indicates that the instances in B give better supervision over the entire tag set. In the case of test on C-(X+Y), the MAP-i and P@K scores are very low. This is because the case is generalized zero-shot learning evaluation setting [37] which makes predictions of seen label even when the ground truth of seen labels have only negatives.

We also see that some results have different trends between the two datasets. For example, test on B gives better results than C on FMA, but the results are opposite to those on MSD. We suspect that this may be due to difference in label cardinality, the average number of labels per instance [31], which can significantly affect performance in multi-label classification [4]. Specifically, FMA tracks have cardinality of 1.18 for B-Y and 1.15 for C-Y, whereas MSD tracks have cardinality of 2.04 in B-Y and only 1.11 in C-Y. This lower cardinality may cause better performance in C-Y than B-Y for MSD. However, we need further investigation considering differences in datasets and side information.

6.2 Multi-label Zero-Shot Retrieval

The results of the retrieval task are reported in Table 3. As in the annotation task, the overall performance is high when training with (A+B)-X. Also, the test results on

Data Split			FMA	MSD
Train	Test	Model	AUC-l	AUC-l
A-X	B-X	Deep Embedding Model	0.7381	0.7161
		Classification Model	0.7250	0.6980

Table 4. Results on retrieval task that compares deep embedding model to classification model. The detail of classification model is described in Section 4.2.

Query	Top 5 Retrieved Tracks (Title / Artist)	Original Last.fm Annotation
guitar	Iron Acton / Beak	psychedelic, experimental, krautrock, english, bass
	Drink Whiskey And Shut up / Brian Setzer	rock
	Thar She Blows / The Halibuts	party, surf, surfrock
	All Quiet On 34th Street / Eric Burdon And The New Animals	rock, rocknroll, hardrock, screamo, 00s
	Gimme Some Lovin' / Traffic	60s, british, classicrock, rock
lovely	Eddie My Love / The Chordettes	50s, doowop, oldies, pop, vocal
	Vaya Con Dios / Les Paul & Mary Ford	50s, jazz, oldies
	(I Can't Help You) I'm Falling Too / Skeeter Davis	country, oldies
	Mr. Blue / The Fleetwoods	oldies, 50s, pop, doowop, ballad
	I'm Blue Again / Patsy Cline	blue, country

Table 5. Top 5 retrieved tracks for a query word from unseen tag subset ('guitar') and an arbitrary word ('lovely').

(A+B+C)-Y are lower than that on (B+C)-Y. We can regard the test on (A+B+C)-Y as a generalized zero-shot learning evaluation setting for the retrieval task. This means that even for instances that do not have a positive annotation on unseen label according to the split (instances that were denoted as A), the evaluation is performed including this instances so to consider whether the model actually make a negative prediction on them. Thus, this is a more strict evaluation setting.

6.3 Deep Embedding Model vs. Classification Model

We also conducted an additional experiment to verify the performance of the deep embedding model in the conventional multi-label classification and tagging task [4]. Table 4 shows results in the retrieval scenario following previous work [5, 7, 13, 24]. We can see that the deep embedding model outperforms the classification model on both datasets. This indicates that associating audio with labels via the side information is a powerful approach even in the conventional multi-label classification and tagging task.

[4] In this evaluation, some labels were excluded because there are some labels that have negative annotations for all instances according to the split.

Smells like teen spirit Nirvana	Superstition Stevie Wonder	Theme to Grace / Lament George Winston
classicrock (unseen)	funk (unseen)	folk
punk	soul	instrumental
rock (unseen)	pop (unseen)	jazz
80s (unseen)	jazz	piano (unseen)
alternative	80s (unseen)	singersongwriter
punkrock	blues (unseen)	chillout
90s	classicrock	acoustic
metal	90s (unseen)	blues
vintage	disco	mellow
alternativerock	dance	chill

Table 6. Top 10 auto-tagging results for examples of well-known songs including unseen tags during training.

Query	General Semantic Space	Zero-shot Embedding Space
guitar	bass, acoustic, piano, vocals, violin, percussion, strings, vocal, music, jazz	instrumental, minimal, rock, acidrock, progressiverock, alternative, psychedelic, folkrock, classicrock, band
lovely	awesome, love, cool, romantic, relaxing, summer, christmas, holiday, vintage, soft	relaxing, relax, lovesongs, easylistening, baby, country, romantic, easy, americana, ballad

Table 7. Comparison of top 10 nearest word vectors (out of 1,126 tags) on general semantic space and the trained zero-shot embedding space.

6.4 Case Study

We conducted case studies to better understand the performance of the zero-shot learning model. Table 6 shows the results of annotation on several famous music tracks. They show that the predictions are reasonable for both seen or unseen tags. Table 5 lists retrieved tracks given a query word and their original labels. We can see that the results are reasonable even if we did not use seen tags. Furthermore, we compared the general semantic space of GloVe model to our trained deep embedding space in Table 7. The examples show the deep embedding space learns the relationship between words in a more musically meaningful way.

7. CONCLUSIONS

In this paper, we showed that zero-shot learning is capable of associating music audio with unseen labels using side information. This allow to use a rich vocabulary of words to describe music, thereby enhancing the experience of music retrieval or recommendation in a more human-friendly way. There is a large room to explore for future work. For example, lyrics can be used as side information which can be obtained without manual human annotation. The neural language models can be also trained to contain more musical context, for example, using text descriptions of playlists or music articles.

8. ACKNOWLEDGMENT

This work was supported by NAVER Corp.

9. REFERENCES

[1] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *Proc. of The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2927–2936, 2015.

[2] Ziad Al-Halah, Makarand Tapaswi, and Rainer Stiefelhagen. Recovering the missing link: Predicting class-attribute associations for unsupervised zero-shot learning. In *Proc. of The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5975–5984, 2016.

[3] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, 2011.

[4] Francisco Charte, Antonio Rivera, María José del Jesus, and Francisco Herrera. Improving multi-label classifiers via label reduction with association rules. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 188–199. Springer, 2012.

[5] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2392–2396, 2017.

[6] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. Fma: A dataset for music analysis. In *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[7] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968, 2014.

[8] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2121–2129. Curran Associates, Inc., 2013.

[9] Yanwei Fu, Tao Xiang, Yu-Gang Jiang, Xiangyang Xue, Leonid Sigal, and Shaogang Gong. Recent advances in zero-shot recognition: Toward data-efficient understanding of visual content. *IEEE Signal Processing Magazine*, 35(1):112–125, 2018.

[10] Yanwei Fu, Yongxin Yang, Tim Hospedales, Tao Xiang, and Shaogang Gong. Transductive multi-label zero-shot learning. *British Machine Vision Association*, 2014.

[11] Eric Humphrey, Simon Durand, and Brian McFee. Openmic-2018: an open dataset for multiple instrument recognition. In *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[12] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2014.

[13] Jongpil Lee and Juhan Nam. Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging. *IEEE Signal Processing Letters*, 24(8):1208–1212, 2017.

[14] Thomas Mensink, Efstratios Gavves, and Cees GM Snoek. Costa: Co-occurrence statistics for zero-shot classification. In *Proc. of The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2441–2448, 2014.

[15] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proc. of International conference on Learning Representations (ICLR)*, 2013.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[17] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[18] Juhan Nam, Keunwoo Choi, Jongpil Lee, Szu-Yu Chou, and Yi-Hsuan Yang. Deep learning for audio-based music classification and tagging: Teaching computers to distinguish rock from bach. *IEEE Signal Processing Magazine*, pages 41–51, 2019.

[19] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. In *Proc. of International conference on Learning Representations (ICLR)*, 2013.

[20] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. In *Advances in neural information processing systems*, pages 1410–1418, 2009.

[21] Devi Parikh and Kristen Grauman. Relative attributes. In *International Conference on Computer Vision (ICCV)*, pages 503–510. IEEE, 2011.

[22] Jiyoung Park, Jongpil Lee, Jangyeon Park, Jung-Woo Ha, and Juhan Nam. Representation learning of music using artist labels. In *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, pages 717–724, 2018.

[23] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proc. of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[24] Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In *Proc. of the International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6, 2016.

[25] Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. Learning deep representations of fine-grained visual descriptions. In *Proc. of The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 49–58, 2016.

[26] Zhou Ren, Hailin Jin, Zhe Lin, Chen Fang, and Alan Yuille. Multiple instance visual-semantic embedding. In *Proc. of the British Machine Vision Conference (BMVC)*, volume 1, page 3, 2017.

[27] Marcus Rohrbach, Michael Stark, György Szarvas, Iryna Gurevych, and Bernt Schiele. What helps where– and why? semantic relatedness for knowledge transfer. In *Proc. of The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 910–917. IEEE, 2010.

[28] Ubai Sandouk and Ke Chen. Multi-label zero-shot learning via concept embedding. *arXiv preprint arXiv:1606.00282*, 2016.

[29] Hendrik Schreiber. Genre ontology learning: Comparing curated with crowd-sourced ontologies. In *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, pages 400–406, 2016.

[30] Krishna Kumar Singh and Yong Jae Lee. End-to-end localization and ranking for relative attributes. In *European Conference on Computer Vision*, pages 753–769. Springer, 2016.

[31] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13, 2007.

[32] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 16(2):467–476, 2008.

[33] Qian Wang and Ke Chen. Multi-label zero-shot human action recognition via joint latent embedding. *arXiv preprint arXiv:1709.05107*, 2017.

[34] Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):13, 2019.

[35] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-UCSD birds 200. 2010.

[36] Yongqin Xian, Zeynep Akata, Gaurav Sharma, Quynh Nguyen, Matthias Hein, and Bernt Schiele. Latent embeddings for zero-shot classification. In *Proc. of The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 69–77, 2016.

[37] Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning - the good, the bad and the ugly. In *Proc. of The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

LEARNING NOTATION GRAPH CONSTRUCTION FOR FULL-PIPELINE OPTICAL MUSIC RECOGNITION

Alexander Pacha
Institute of Information Systems Engineering, TU Wien, Austria
alexander.pacha@tuwien.ac.at

Jorge Calvo-Zaragoza
Pattern Recognition and Artificial Intelligence Group
University of Alicante, Spain
jcalvo@dlsi.ua.es

Jan Hajič jr.
Institute of Formal and Applied Linguistics,
Charles University, Prague
hajicj@ufal.mff.cuni.cz

ABSTRACT

Optical Music Recognition (OMR) promises great benefits to Music Information Retrieval by reducing the costs of making sheet music available in a symbolic format. Recent advances in deep learning have turned typical OMR obstacles into clearly solvable problems, especially the stages that visually process the input image, such as staff line removal or detection of music-notation objects. However, merely detecting objects is not enough for retrieving the actual content, as music notation is a configurational writing system where the semantic of a primitive is defined by its relationship to other primitives. Thus, OMR systems must employ a notation assembly stage to infer such relationships among the detected objects. So far, this stage has been addressed by devising a set of predefined rules or grammars, which hardly generalize well. In this work, we formulate the notation assembly stage from a set of detected primitives as a machine learning problem. Our notation assembly is modeled as a graph that stores syntactic relationships among primitives, which allows us to capture the configuration of symbols in a music-notation document. Our results over the handwritten sheet music corpus MUSCIMA++ show 95.2% precision, 96.0% recall, and an F-score of 95.6% in establishing the correct syntactic relationships. When inferring relationships on data from a music object detector, the model achieves 93.2% precision, 91.5% recall and an F-score of 92.3%.

1. INTRODUCTION

Optical Music Recognition is the field of research that investigates how to read music notation in documents computationally. This technology enables many computational tasks that, otherwise, could not be performed directly on the music sources themselves [17]. One interesting application of OMR is concerned with reconstructing the notes encoded in the music-notation document, also referred to as *replayability* [22]. In particular, the objective of the replayability application is to recover the pitches, onsets, durations, and velocities of notes from a document and export them into a symbolic representation. This symbolic representation—e.g., a MIDI file—is already a very useful abstraction of the music itself and allows for plugging in a wide range of music information retrieval tools. However, despite prolonged efforts, the replayability application is still under research [4, 7, 16, 36].

Given the wealth of information that is contained in a music score, the task of decoding its content is usually addressed by dividing the process into smaller stages that represent limited challenges. The general pipeline, proposed first by Bainbridge and Bell [3] and later refined by Rebelo et al. [29], is considered a de-facto standard, which organizes the process into four main blocks: i) preprocessing, which works over the input image to ease further steps and make the system more robust; ii) music object detection, which is in charge of retrieving and classifying all objects and glyphs of the image; iii) notation assembly, which must infer the relationships among the detected objects to reconstruct the music notation itself; and iv) encoding, which exports the symbolic reconstruction into the desired format, typically MIDI for replayability or an XML-based encoding such as MusicXML [15] or MEI [19] for further computational processing.

As our starting point towards completing the OMR pipeline, we assume that the music object detection stage can be solved reliably, which allows us to investigate how to deal with the later stages. In this paper, we want to focus in particular on the third stage, which is responsible for the notation assembly. Although previous work exists, most approaches are based on predefined rules that hardly generalize, and that only work for a limited set of scenarios. In contrast, we propose a well-principled machine learning approach, which addresses the problem in a generalizable way, provided there is convenient training data.

2. RELATED WORK

Most literature on OMR focuses on the first stages of the pipeline. This comes as no surprise because if one struggles with detecting music objects in an image reliably, it is understandable that subsequent stages that build on top of that are often neglected. With the appearance of deep

learning in OMR, however, many steps that traditionally produced suboptimal results, such as the staff-line removal or symbol classification, have seen drastic improvements [14, 26] and are no longer considered obstacles for OMR development.

Deep learning also caused some steps to become obsolete or collapse into a single (bigger) stage. For instance, the music object detection stage, which was traditionally separated into segmentation plus classification stages, is currently addressed in a single step. Convolutional neural networks have been shown to be able to deal with the music object detection stage holistically, without having to remove staff lines at all [25]. A compelling advantage is the capability of these models to be trained in a single step by merely providing pairs of images and positions of the music objects to be found, eliminating the preprocessing step altogether [24, 35]. This issue has been the subject of intense recent research. A comparison of existing approaches to holistic music object detection is presented in the work of Pacha et al. [27].

Since the beginning of the OMR research, there have been attempts to complete the full pipeline, including the notation assembly stage. Below, we introduce some particular proposals to perform this stage that can be found in the existing literature. They can be broadly divided into grammar-based approaches, and approaches that rely on heuristics and pre-defined rules.

2.1 Grammar-based approaches

Formal grammars represent the most widely used description of music notation. This feels natural, given that music notation has syntactic rules and hierarchical structures that invite such a formalization. These grammars are manually built to describe the expected relationships among music-notation objects and then used to reconstruct the music notation from the detected primitives [1–3, 5, 6, 30, 33]. The 2D nature of music notation also inspired graph grammars, as in the work of Fahmy and Blostein [12]. A prominent example of this approach is the DMOS system, proposed by Coüasnon et al. [8,9], which uses a definite clause grammar for describing the relations between graphical objects on two levels: a graphical one that assists the recognition of symbols and a syntactic one, which introduces the musical semantics into the process.

2.2 Heuristical approaches

The other set of approaches relies on *ad hoc* rules for the music notation at hand. This includes assumptions about the configuration and position of the occurring primitives to reconstruct composite symbols and the notation graph [10, 23, 28, 34]. Rossant et al. [31] additionally considered fuzzy modeling, which allows for self-correction during the recognition [32]. Their system evaluated different hypotheses of recognized symbols to verify the compatibility between them.

3. NOTATION ASSEMBLY

The related works clearly show a lack of machine learning approaches. This work aims to fill that gap, by proposing a formulation of the notation assembly stage based on machine learning models. The advantage of such models is that they provide greater flexibility since they can vary their behavior by just changing the provided training set. This is especially interesting for OMR, where there is a great diversity of scenarios depending on the epoch or type of composition of the music scores.

The conventional OMR pipeline foresees that the notation assembly stage infers the relationships among previously detected music objects to retrieve the necessary information to infer the sequence of notes and rests.

Our approach understands that music notation can be modeled as a directed graph $G = (V, T)$, hereafter referred to as Music Notation Graph (MuNG). V represents the set of vertices, where $\zeta(v), v \in V$ is the label associated with a vertex. T represents the set of directed edges, such that $t_i = (v_1, v_2), t_i \in T, v_1, v_2 \in V$ denotes an edge from vertex v_1 to vertex v_2. The primitives that make up the music notation, such as noteheads or stems, are modeled as vertices of this graph, while the relationships between these symbols are modeled by the edges. In our MuNG, the edges are not labeled, but there are two types of relationships:

- Syntactic edges that relate elements syntactically. This includes relationships between primitives that make up a composite symbol, such as an eighth note, which consist of a notehead, a stem, and a flag or beam as well as general relationships, e.g., between an accidental and the notehead that is affected by it.

- Precedence edges that specify the temporal order between notes. In most cases, the position on the horizontal axis is sufficient to infer this kind of relationship; however, for polyphonic music, a more sophisticated mechanism is needed to handle ambiguous situations.

We can, therefore, define the set of edges as $T = S \cup P$, where S is the set of edges that define the syntactic relationships and P is the set of edges that define the precedence relationships. A graphical representation of MuNG is shown in Fig. 1. The primary goal of our work is to train a machine learning model to construct such a MuNG G from a music score image.

4. LEARNING MUSIC NOTATION GRAPH ASSEMBLY

There are existing algorithms that are capable of dealing with the input image and retrieving a set of detected music-notation primitives. In other words, these algorithms process the input and provide the set of vertices V, along with its associated labels and bounding-boxes. In order to complete the OMR pipeline for replayability, we also need to recover the set of edges T.

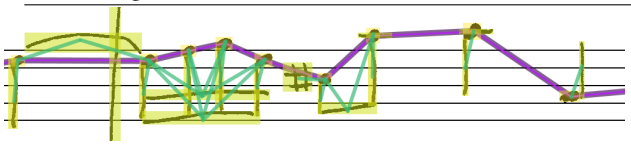


Figure 1: Graphical representation of a Music Notation Graph in a selected excerpt of music notation: vertices are highlighted with transparent yellow bounding boxes around the music-notation primitives, syntactic edges are shown as transparent cyan lines, and precedence edges are shown as transparent purple lines connecting the noteheads.

We propose a principled way of inferring T without resorting to a set of fixed rules but using machine learning. Our system learns to establish these relationships from a conveniently annotated training set so that the rules are implicitly modeled by the machine learning model.

The edges that relate vertices of the set T have an unlabeled binary nature; i.e. for each pair of vertices, a relationship either exists or not. Formally speaking, the inference of these relationships can be formulated as a function $f : V \times V \rightarrow \{0, 1\}$. However, given their different nature, the set of edges S and P are inferred by independent models. To learn the functions f_S and f_P, for the edges of S and P, respectively, we propose to train binary classifiers that receive two vertices and predict whether such relationship must be established or not. To do so, one would have to estimate the potential relationship between each pair of symbols, which entails high computational costs. However, it is obvious that most of these relationships are unfeasible. Since the music object detection stage also retrieves some associated information, such as the label $\zeta(v)$ associated to each vertex and the bounding box of that object in the input score image, we can use this information to filter edges by two criteria:

1. An edge is only feasible if the distance between the bounding boxes of their vertices falls below a certain threshold t. In other words, two vertices that are too far apart cannot be related.

2. An edge is only feasible if the labels of its associated vertices are "compatible", e.g., a notehead with a stem. This eliminates a large number of incompatible combinations, such as an edge between an accidental and a rest. The compatibility map is a fixed list of vertex pairs that, according to the syntax of modern music notation, can hold a relationship to each other.

Then, given two vertices v_1 and v_2, for which their edge is declared feasible, we train a deep convolutional neural network to predict whether there must be an edge from v_1 to v_2 or not. We generate a multi-channel image with a fixed size that serves as input features for the neural network, which consists of:

- Channel 1: the patch of the input score image that is centered at the objects represented by v_1 and v_2.
- Channel 2: the binary mask of the object v_1
- Channel 3: the binary mask of the object v_2

The required information to generate these multi-channel images can be obtained from the bounding boxes of v_1 and v_2, which are expected to be generated during the preceding music object detection stage. Note, that the masks for channel 2 and 3 are obtained from the bounding boxes and the underlying image, which means that the masks can (partially) include other objects as well unless the exact masks are provided via pixelwise segmentation [16, 35].

The network is then fed with this 3-channel image and trained to predict 1 if there should be a relationship between the vertices, and 0 otherwise. Visualizations of the input images are given in Fig. 2.

4.1 Dataset

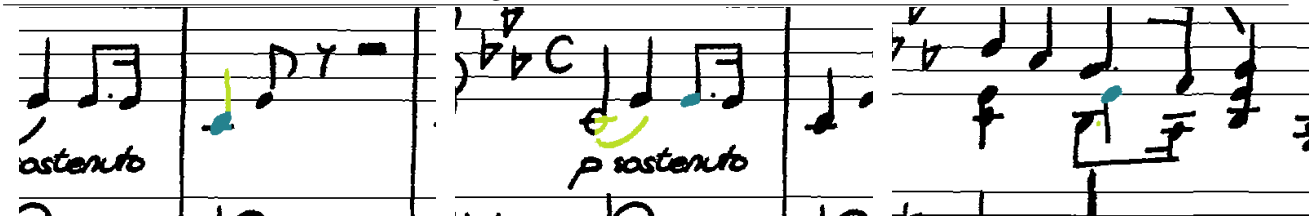
To carry out our experiments we need a corpus, which has annotations for both the individual symbols as well as their relationships. Currently, the only publicly available dataset which fulfills this requirement is the MUSCIMA++ dataset [18] of handwritten music notation. It provides symbol-level annotations as well as relationship annotations for 140 out of 1 000 images from the CVC-MUSCIMA dataset [13]. The MUSCIMA++ dataset contains 91 254 annotated symbols, consisting of both notation primitives and higher-level notation objects, such as key signatures or time signatures as well as 82 247 explicitly marked relationships between symbol pairs.

Unfortunately, the precedence relationships between notes are not included in the MUSCIMA++ dataset, so our experiments consider only the syntactic edges. However, the formulation and the proposed approach are very similar and should work for both kinds of edges.

4.2 Relationship Reconstruction

For learning the relationships, we train a convolutional neural network in PyTorch with five consecutive blocks, each consisting of a convolution, batch normalization, a non-linearity (ReLU), and max-pooling, before going into a fully connected layer with a single output neuron followed by a sigmoid activation function that produces the final estimation. The network has 28 865 parameters in total. We use the Binary Cross-Entropy loss and train with the Adam optimizer [20] until the validation performance has not improved for ten epochs, upon which we stop.

The data-loading routine presents the biggest challenge because it has to construct the multi-channel images as described in Sect. 4. To efficiently generate the set of vertex-pairs, we compute the pairwise distance between all objects in an image but filter them considerably afterward by the distance and compatibility criteria (see Sec. 4). The distance threshold was set to $t = 200$ pixels for including most valid edges from the MUSCIMA++ dataset. Valid relationships between objects that are further apart than 200 pixels are extremely rare and were neglected in favor of



(a) A positive example of two objects that are related.

(b) A negative example of two objects that are unrelated.

(c) A hard negative example of a dot that could be related to the notehead, but is not.

Figure 2: Three samples of images that are used during training. The mask given in channel 2 is shown as bright green overlay and the mask from channel 3 as cyan overlay.

computational efficiency. Our compatibility map contains 225 valid combinations of primitives. To improve the performance even further and simplify the classification task, the input image for the neural network is cropped to a sub-image of 512×256 pixels (width $\times$ height), containing the two objects of interest at its center. Both the distance threshold and the sub-image dimensions are hyperparameters that are dataset-dependent but can be obtained by running a statistical analysis on the used dataset.

We split the 140 images of the dataset into 60 % training data, 20 % validation data, and 20% test data. In each epoch, the network is trained on approximately 250 000 images of candidate pairs. Approximately 25 percent of the candidates contain positive examples. The best results were obtained after just 12 epochs before the network started to overfit and the validation performance declined. The source-code is publicly available on Github. [1]

4.3 Music Object Detection

Since the notation assembly stage begins after the music objects have been detected in the score image, we also wanted to evaluate, how well our approach works on actual detection results. For obtaining such results, we resort to a state-of-the-art music object detector as proposed by Pacha et al. [25] with a minor modification: While we do divide the full page into sub-images containing one stave each, we do not see the need for cutting the images any further. The model selection and training procedure remains unchanged. We split the dataset into 100 images for training, 20 images for validation and 20 images for testing, as proposed by the authors of the MUSCIMA++ dataset. The improved implementation is publicly available. [2]

We evaluate the trained model on the test set for the stave-wise individual images and report the Mean Average Precision (mAP) as defined for the COCO challenge [21] which is a unified metric, commonly used for object detection tasks. The trained model achieves 69.5 % mAP. For comparison, we also report a mAP of 93.3 % when using the mAP as defined for the PASCAL VOC challenge [11], which was used in the original paper. Finally, the images are merged into the full-page results upon we achieve: 34.5 % mAP / 45.2 % w-mAP [3] (COCO) and 53.8 % mAP / 80.9 % w-mAP (PASCAL). As our main focus is on learning relationships and not music object detection, we do not go into further details on these numbers. However, we want to point out that the COCO metric is very strict and probably underestimating the performance of the music object detector (see Fig. 3 for an example output).

4.4 Evaluation Protocol

Once the music objects have been detected, and their relationships established, the system can produce a complete MuNG that can be compared with the reference MuNG, provided as ground truth. However, it is necessary to first establish the correspondences between vertices from the prediction and the ground-truth. To do so, we assume that a detected object v_1 corresponds to a ground-truth object v_2 if they depict the same class $\zeta(v_1) = \zeta(v_2)$ and their Intersection over Union exceeds 50 %.

Once the vertices of the ground-truth are matched with the detected objects, it is possible to compute the statistics. If an established relationship is correct, it is considered a true positive (TP); if an established relationship is incorrect, it is considered a false positive (FP); and, if an expected relationship is not predicted, it is considered a false negative (FN). Then, we can compute precision (P), recall (R), and F-score (F_1) metrics:

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad R = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad F_1 = 2\frac{P \times R}{P + R}$$

P measures how reliable the established relationships are, whereas R measures the ability of the model to retrieve as many relationships as possible. F_1 summarizes both metrics with a single value.

Note that, although our evaluation is primarily focused on the relationships between objects, the used metrics are affected by the performance of the music object detector. Errors from earlier stages of the OMR process propagate to later stages. So if musical objects were missed, their relationships are counted as false negatives. To account for this, we evaluate our model in two ways:

[1] https://github.com/OMR-Research/MungLinker

[2] https://github.com/apacha/MusicObjectDetector-TF

[3] Weighted Mean Average Precision is the Mean Average Precision, weighted by the frequency of the occurring classes, which is higher because frequent classes yielded better results than rare ones.

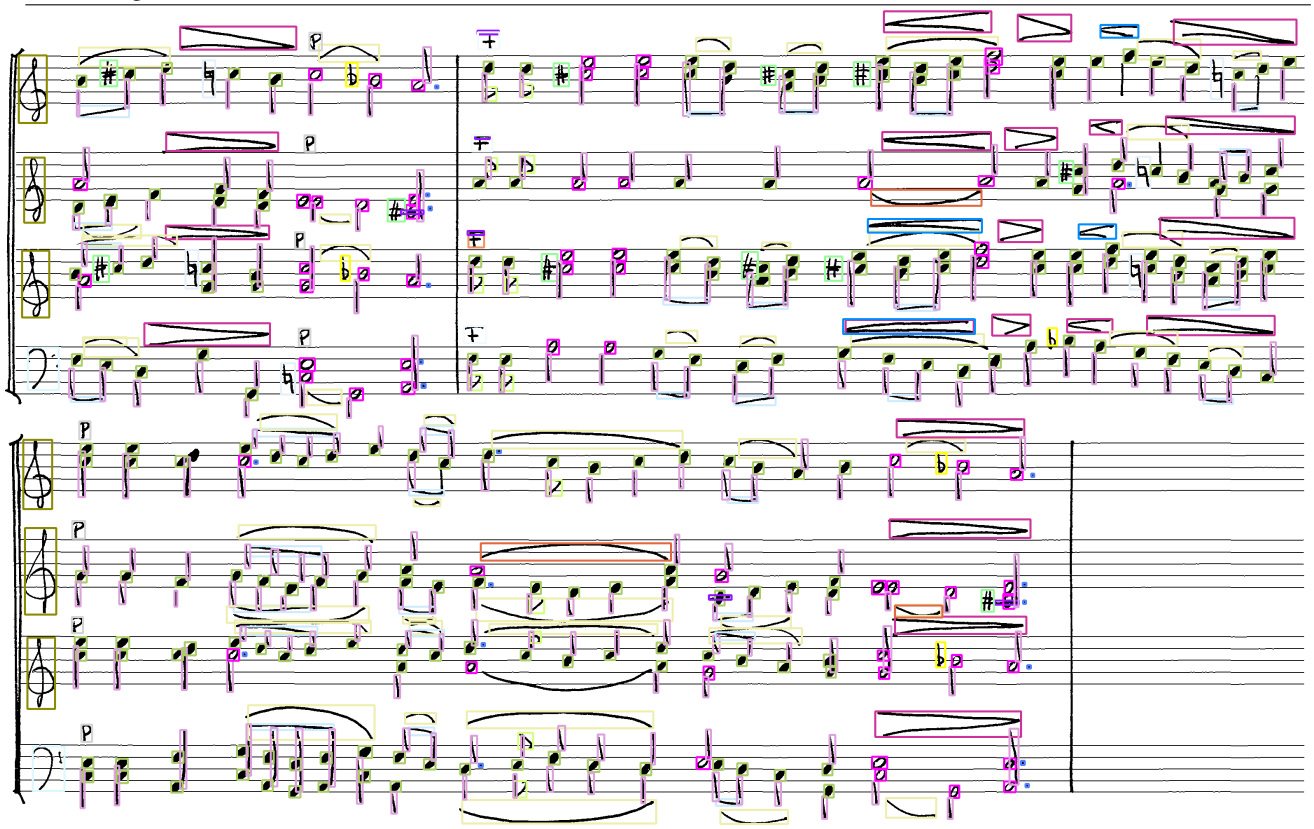


Figure 3: Sample output of the improved music object detector. Each detected object v has a box around it, with the color representing the class $\zeta(v)$ of the particular object, e.g., light green for full-noteheads.

- over a hypothetical set of perfect detections, which we can extract from the ground-truth of the corpus, and
- over the result of an actual music object detector, specifically using the state-of-the-art model, described in Sect. 4.3.

These settings allow us to answer the two questions: Does the proposed approach for reconstructing the MuNG with a machine learning model work at all? If yes, how well does the system perform in a real-world scenario, when confronted with (imperfect) object detector results instead of the perfect ground-truth bounding boxes?

4.5 Results

The main objective of our work is to demonstrate that the notation assembly stage can be formulated as a machine learning task. The main results of our experiments are given in Table 1. It can be observed that the proposed approach is highly effective: in all cases, values above 90 % are reported.

When starting from ground-truth music object detection, our model yields $P = 95.2\,\%$, $R = 96.0\,\%$, and $F_1 = 95.2\,\%$, which indicates a successful approach to completing the OMR pipeline. In case of starting from actual results of a state-of-the-art detector, performance decreases slightly to $P = 93.2\,\%$, $R = 91.5\,\%$, and $F_1 = 92.3\,\%$. We think this is because the location of the objects is not always exact (leading to a lower P) and missing symbols cause relationships to be irrecoverable (leading to a lower R).

	Graph Edges / Relationships		
	Precision	**Recall**	**F-Score**
Perfect Detection	95.2%	96.0%	95.6%
Real Detector	93.2%	91.5%	92.3%

Table 1: Overall performance of the proposed machine learning model to reconstruct syntactic edges of the Music Notation Graph (MuNG), given hypothetically perfect detection results (top row), and given results from a state-of-the-art detector (bottom row).

In order to provide more experimental insights, Table 2 reports 10 out of the 225 compatible combinations of relationships that are most common in the MUSCIMA++ dataset. As might be expected, the *notehead* primitives are involved in all of these frequent combinations. In this regard, our model obtains nearly optimal results for these over-represented cases. Note that these relationships are of particular relevance to be able to decode the notes that appear in the score. When comparing the individual results to the overall results in Table 1, the discrepancy can be explained by looking at the remaining 215 combinations that are not shown. Many of these have a much lower F_1, probably because they are under-represented in the dataset.

From	To	Number of candidate pairs in the dataset	F-Score on the test set
notehead-full	stem	158064	99.5%
notehead-full	beam	61253	98.7%
notehead-full	leger_line	47503	98.1%
notehead-full	slur	24738	96.4%
notehead-full	8th_flag	12877	97.7%
notehead-full	sharp	12563	97.5%
notehead-full	duration-dot	12305	96.7%
notehead-empty	stem	9488	100.0%
notehead-full	staccato-dot	8628	96.8%
notehead-full	natural	7160	98.7%

Table 2: Overview of the ten most common combinations of object-pairs, along with the number of generated candidate pairs in the dataset, as seen by the network. The last column contains the F-scores that were reported for the individual combinations when evaluating the trained model on the test set, containing the ground truth of music primitives v.

5. CONCLUSION AND OUTLOOK

In this work, we study how to complete the OMR pipeline from the previous efforts to detect the music objects within the input image. Our approach seeks the construction of a music notation graph that stores the information of the notation primitives as well as their syntactic and precedence relationships. We propose a machine learning model that can predict whether two primitives are related to each other or not.

Results over the set of syntactic relationships from the handwritten sheet music dataset MUSCIMA++ show that our approach is very effective. We obtain success rates close to the optimum when establishing the correct relationships from the ground-truth primitives ($F_1 = 95.6\%$). When re-evaluating the results starting from the primitives detected by a state-of-the-art music object detector, a slightly lower performance can be observed ($F_1 = 92.3\%$). These figures indicate that the notation assembly stage of the OMR pipeline can be solved reliably with a machine learning model, given a curated set of annotated scores. Comparing our approach to existing methods is extremely difficult, if not impossible, because:

- most existing solutions are black boxes with closed source-code, or there is no available implementation at all,
- only a few systems are capable of handling handwritten modern notation, and
- it is unclear how to compare the music notation assembly stage between two different systems, especially given that the notation graph is only an intermediate representation.

So, although the results are promising, we still see many interesting avenues for further research. For instance, by adding data augmentation during training to make the notation assembly model more robust against variations in the bounding box retrieval of the first stage. Also, we plan to look into providing other meaningful features to the network, such as the class labels $\zeta(v)$ of the involved primitives $v \in V$. Furthermore, we observed that the fixed-sized input patch given to the network is often covering a much larger area than required to contain the objects of interest, especially when they are very close (see Fig. 2c). This could be handled by using size-independent neural network layers such as *Global Pooling*, instead of flattening the features and feeding them into a fully-connected layer, allowing us to adjust the input patch for each sample.

We also believe that the notation assembly stage could benefit from having a broader set of hypotheses about the objects detected in the previous stage, instead of a fixed set of proposals. State-of-the-art music object detectors are based on statistical neural models that are able to provide a probability distribution over the whole set of possible detection hypotheses. When it comes to recognizing, we are typically interested in the most likely hypothesis—the one that is proposed as an answer—forgetting the other ones. However, it is certainly interesting to exploit this statistical modeling: the notation assembly algorithm could establish relationships that are more logical a priori, although the objects involved have a lower probability according to the object detector. These types of approaches have yet to be explored in the field of OMR.

And finally, for completing the OMR pipeline, the encoding stage is still missing. However, we see two benefits of the notation graph representation: the encoding can be implemented by experts in music encoding that are proficient in a particular format and given a complete graph representation, there is no restriction on the actual output format because the graph contains all the information that is present in the image.

6. REFERENCES

[1] Alfio Andronico and Alberto Ciampa. On automatic pattern recognition and acquisition of printed music. In *International Computer Music Conference*, Venice, Italy, 1982.

[2] David Bainbridge. *Extensible optical music recognition*. PhD thesis, University of Canterbury, 1997.

[3] David Bainbridge and Tim Bell. The challenge of optical music recognition. *Computers and the Humanities*, 35(2):95–121, 2001.

[4] Arnau Baró, Pau Riba, Jorge Calvo-Zaragoza, and Alicia Fornés. From optical music recognition to handwritten music recognition: A baseline. *Pattern Recognition Letters*, 123:1–8, 2019.

[5] Pierfrancesco Bellini, Ivan Bruno, and Paolo Nesi. Optical music recognition: Architecture and algorithms. In *Interactive Multimedia Music Technologies*, pages 80–110. IGI Global, Hershey, PA, USA, 2008.

[6] Dorothea Blostein and Henry S. Baird. A critical survey of music image analysis. In *Structured Document Image Analysis*, pages 405–434. Springer Berlin Heidelberg, 1992.

[7] Jorge Calvo-Zaragoza and David Rizo. End-to-end neural optical music recognition of monophonic scores. *Applied Sciences*, 8(4), 2018.

[8] Bertrand Coüasnon, Pascal Brisset, and Igor Stéphan. Using logic programming languages for optical music recognition. In *3rd International Conference on the Practical Application of Prolog*, 1995.

[9] Bertrand Coüasnon and Jean Camillerapp. A way to separate knowledge from program in structured document analysis: Application to optical music recognition. In *3rd International Conference on Document Analysis and Recognition*, pages 1092–1097, 1995.

[10] Michael Droettboom, Ichiro Fujinaga, and Karl MacMillan. Optical music interpretation. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 378–387, Berlin, Heidelberg, 2002.

[11] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015.

[12] Hoda M. Fahmy and Dorothea Blostein. A graph grammar programming style for recognition of music notation. *Machine Vision and Applications*, 6(2):83–99, 1993.

[13] Alicia Fornés, Anjan Dutta, Albert Gordo, and Josep Lladós. CVC-MUSCIMA: A ground-truth of handwritten music score images for writer identification and staff removal. *International Journal on Document Analysis and Recognition*, 15(3):243–251, 2012.

[14] Antonio-Javier Gallego and Jorge Calvo-Zaragoza. Staff-line removal with selectional auto-encoders. *Expert Systems with Applications*, 89:138–148, 2017.

[15] Michael Good. MusicXML: An internet-friendly format for sheet music. Technical report, Recordare LLC, 2001.

[16] Jan Hajič jr., Matthias Dorfer, Gerhard Widmer, and Pavel Pecina. Towards full-pipeline handwritten OMR with musical symbol detection by u-nets. In *19th International Society for Music Information Retrieval Conference*, pages 225–232, Paris, France, 2018.

[17] Jan Hajič jr., Marta Kolárová, Alexander Pacha, and Jorge Calvo-Zaragoza. How current optical music recognition systems are becoming useful for digital libraries. In *5th International Conference on Digital Libraries for Musicology*, pages 57–61, Paris, France, 2018.

[18] Jan Hajič jr. and Pavel Pecina. The MUSCIMA++ dataset for handwritten optical music recognition. In *14th International Conference on Document Analysis and Recognition*, pages 39–46, Kyoto, Japan, 2017.

[19] Andrew Hankinson, Perry Roland, and Ichiro Fujinaga. The music encoding initiative as a document-encoding framework. In *12th International Society for Music Information Retrieval Conference*, pages 293–298, 2011.

[20] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *Computing Research Repository*, abs/1412.6980, 2014.

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014.

[22] Hidetoshi Miyao and Robert Martin Haralick. Format of ground truth data used in the evaluation of the results of an optical music recognition system. In *4th International Workshop on Document Analysis Systems*, pages 497–506, Brasil, 2000.

[23] Kia Ng. Music manuscript tracing. *Lecture Notes in Computer Science*, 2390:322–334, 2002.

[24] Alexander Pacha and Jorge Calvo-Zaragoza. Optical music recognition in mensural notation with region-based convolutional neural networks. In *19th International Society for Music Information Retrieval Conference*, pages 240–247, Paris, France, 2018.

[25] Alexander Pacha, Kwon-Young Choi, Bertrand Coüasnon, Yann Ricquebourg, Richard Zanibbi, and Horst Eidenberger. Handwritten music object detection: Open issues and baseline results. In *13th International Workshop on Document Analysis Systems*, pages 163–168, 2018.

[26] Alexander Pacha and Horst Eidenberger. Towards a universal music symbol classifier. In *14th International Conference on Document Analysis and Recognition*, pages 35–36, Kyoto, Japan, 2017.

[27] Alexander Pacha, Jan Hajič jr., and Jorge Calvo-Zaragoza. A baseline for general music object detection with deep learning. *Applied Sciences*, 8(9):1488–1508, 2018.

[28] Christopher Raphael and Jingya Wang. New approaches to optical music recognition. In *12th International Society for Music Information Retrieval Conference*, pages 305–310, Miami, Florida, 2011.

[29] Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre R.S. Marcal, Carlos Guedes, and Jamie dos Santos Cardoso. Optical music recognition: State-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, 1(3):173–190, 2012.

[30] K. Todd Reed and J. R. Parker. Automatic computer recognition of printed music. In *13th International Conference on Pattern Recognition*, pages 803–807, 1996.

[31] Florence Rossant and Isabelle Bloch. A fuzzy model for optical recognition of musical scores. *Fuzzy Sets and Systems*, 141(2):165–201, 2004.

[32] Florence Rossant and Isabelle Bloch. Robust and adaptive OMR system including fuzzy modeling, fusion of musical rules, and possible error detection. *EURASIP Journal on Advances in Signal Processing*, 2007(1):081541, 2006.

[33] Mariusz Szwoch. Guido: A musical score recognition system. In *9th International Conference on Document Analysis and Recognition*, pages 809–813, 2007.

[34] Lorenzo J. Tardón, Simone Sammartino, Isabel Barbancho, Verónica Gómez, and Antonio Oliver. Optical music recognition for scores written in white mensural notation. *EURASIP Journal on Image and Video Processing*, 2009(1):843401, 2009.

[35] Lukas Tuggener, Ismail Elezi, Jürgen Schmidhuber, and Thilo Stadelmann. Deep watershed detector for music object recognition. In *19th International Society for Music Information Retrieval Conference*, pages 271–278, Paris, France, 2018.

[36] Eelco van der Wel and Karen Ullrich. Optical music recognition with convolutional sequence-to-sequence models. In *18th International Society for Music Information Retrieval Conference*, Suzhou, China, 2017.

AN ATTENTION MECHANISM FOR MUSICAL INSTRUMENT RECOGNITION

Siddharth Gururani[1] **Mohit Sharma**[2] **Alexander Lerch**[1]

[1] Center for Music Technology, Georgia Institute of Technology, USA
[2] School of Interactive Computing, Georgia Institute of Technology, USA
{siddgururani, mohit.sharma, alexander.lerch}@gatech.edu

ABSTRACT

While the automatic recognition of musical instruments has seen significant progress, the task is still considered hard for music featuring multiple instruments as opposed to single instrument recordings. Datasets for polyphonic instrument recognition can be categorized into roughly two categories. Some, such as MedleyDB, have strong per-frame instrument activity annotations but are usually small in size. Other, larger datasets such as OpenMIC only have weak labels, i.e., instrument presence or absence is annotated only for long snippets of a song. We explore an attention mechanism for handling weakly labeled data for multi-label instrument recognition. Attention has been found to perform well for other tasks with weakly labeled data. We compare the proposed attention model to multiple models which include a baseline binary relevance random forest, recurrent neural network, and fully connected neural networks. Our results show that incorporating attention leads to an overall improvement in classification accuracy metrics across all 20 instruments in the OpenMIC dataset. We find that attention enables models to focus on (or 'attend to') specific time segments in the audio relevant to each instrument label leading to interpretable results.

1. INTRODUCTION

Musical instruments, both acoustic and electronic, are necessary tools to create music. Most musical pieces comprise of a combination of multiple musical instruments resulting in a mixture with unique timbre characteristics. Humans are fairly adept at recognizing musical instruments in the music they hear. Recognizing instruments automatically, however, is still an active area of research in the field of Music Information Retrieval (MIR). Instrument recognition in isolated note or single instrument recordings has achieved a fair amount of success [14, 26]. Recognizing instruments in music with multiple simultaneously playing instruments, however, is still a hard problem. The task is difficult because of (i) the superposition (in both time and frequency) of multiple sources/instruments, (ii) the large variation of timbre within one instrument, and (iii) the lack of annotated data for supervised learning algorithms.

Identifying music in audio recordings is helpful for general retrieval systems by allowing users to search for music with specific instrumentation [32]. Instrument recognition can also be helpful for other MIR tasks. For example, instrument tags may be vital for music recommendation systems to model users' affinity towards certain instruments, genre recognition systems could also improve with genre-dependent instrument information. Building models conditioned on a reliable detection of instrumentation could also lead to improvements for tasks such as automatic music transcription, source separation, and playing technique detection.

As mentioned above, one of the challenges in MIR in general, and in instrument recognition in particular, is the lack of large-scale annotated or labeled data for supervised machine learning algorithms [17, 36]. Datasets for instrument recognition in polyphonic music can broadly be divided into strongly and weakly labeled. A weakly labeled dataset (WLD) contains clips that may be several seconds long and have labels for one or more instruments for their entirety without annotating the exact onset and offset times of the instruments. A strongly labeled dataset (SLD), however, contains audio with fine-grained labels of instrument activity. WLDs are easier to annotate compared to SLDs and therefore scale better. Even though SLDs enable strong supervision of learning algorithms, the smaller size may lead to poor performance of deep learning methods. WLDs, however, have the disadvantage that an instrument may be marked positive even if the instrument is active for a very short duration of the entire clip. This makes it challenging to train models with WLDs.

Models for recognition in weakly labeled data may benefit from inferring the specific location in time of the instrument to be recognized. We formulate the polyphonic instrument recognition task as a multi-instance multi-label (MIML) problem, where each weakly labeled example is a collection of short-time instances, each with a contribution towards the labels assigned to the example. Toward that end, we apply an attention mechanism to aggregate the predictions for each short-time instance and compare this approach to other models which include binary-relevance

random forests, fully connected networks, and recurrent neural networks. We hypothesize that the ability of the attention model to weigh relevant and suppress irrelevant predictions for each instrument leads to better classification accuracy. We visualize the attention weights and find that the model is able to mostly localize the instruments, thereby enhancing the interpretability of the classifier.

The next section reviews literature in instrument recognition and audio tagging or classification. Sect. 3 discusses various datasets for instrument recognition and the challenges associated. Next, Sect. 4 formulates the problem and describes the model. Sect. 5 specifies the various experiments and the evaluation metrics to measure performance. We report the results of the experiments and discuss them in Sect. 6. Finally, in Sect. 7 we conclude the paper suggesting future directions for research.

2. RELATED WORK

2.1 Musical Instrument Recognition

Instrument recognition in audio containing a single instrument can refer to both recognition from isolated notes or recognition from solo recordings of pieces. We refer to [15, 26] for a review of literature in single instrument and monophonic instrument recognition.

Current research has focused on instrument recognition in polyphonic and multi-instrument recordings. While traditional approaches extract features followed by classification algorithms were previously prevalent [9, 21], deep neural networks have dominated recent work in this field. Han et al. [13] applied Convolutional Neural Networks (CNNs) to the task of predominant instrument recognition on the IRMAS dataset [5] and outperformed various feature-based techniques. Li et al. [25] proposed to learn features from raw audio using CNNs for instrument recognition using the MedleyDB dataset [4]. Gururani et al. [12] compared various neural network architectures for instrument activity detection using two multi-track datasets containing fine-grained instrument activity annotations: MedleyDB and Mixing Secrets [11] . They found significant improvement of CNNs and Convolutional Recurrent Neural Networks (CRNNs) over fully connected networks and proposed a method for visualizing model confusion in a multi-label setting. Hung et al. [19] utilized the fine-grained instrument activity as well as pitch annotations in the MusicNet dataset [33] and showed the benefits of pitch-conditioning on instrument recognition performance. In follow-up research, Hung et al. [18] proposed a multi-task learning approach for instrument recognition involving the prediction of pitch in addition to instrumentation. They released a synthetic, large-scale, and strongly-labeled dataset generated from MIDI files for evaluation and found that multi-task learning outperforms their previous approach of using pitch features as additional inputs.

2.2 Audio event detection, tagging and classification

The task of audio or sound event classification shares many commonalities with instrument recognition. Both tasks aim to identify a time-variant sound source in a mixture of multiple sound sources. A few key differences are that research in sound event classification typically focuses on uncorrelated sounds such as motor noise, car horns, baby cries, or dog barks, while musical audio is highly correlated. Additionally, music has a rich harmonic and temporal structure usually absent in audio captured from real world acoustic scenes.

For a historic review of work in sound event and audio classification, we refer readers to the survey article by Stowell et al. [31]. We focus on more recent literature involving deep neural network architectures —which are now the standard approach— as well as on methods that focus on addressing weak labels.

Hershey et al. [16] adapted deep CNN architectures from computer vision and found that they are effective for large-scale audio classification. Cakir et al. [6] researched the benefits of CRNNs for sound event detection over models comprising of only CNNs. They found that the ability of RNNs to capture long-term temporal context helps improve performance against models only comprising CNNs. Adavanne et al. [1] proposed to use spatial features extracted from multi-channel audio as inputs for CRNN architectures. They found that presenting these features as separate layers to the model outperforms concatenation of these features at the input stage.

Learning from weakly labeled data has also been a focus in audio classification. Most works utilize the Multiple-Instance Learning (MIL) framework for the task, where each example is a labeled bag containing multiple instances whose labels are unknown. Kumar and Raj [24] utilized support vector machines and neural networks for solving the MIL problem. They train bag-level classifiers capable of predicting instances and are hence also useful for localization of sound events. Similarly, Kong et al. [22] proposed decision-level attention to solve the MIL problem for Audio Set [10] classification. Attention is applied to instance predictions to enable weighted aggregation for bag-level prediction. Kong et al. [23] extended this and propose feature-level attention where instead of applying attention to the instance predictions, it is applied to the hidden layers of a neural network to construct a fixed-size embedding for the bag. Finally a fully connected network predicts the labels for the bag using the embedding vector. McFee et al. [27] compared various methods for aggregating or pooling instance-level predictions. They developed an adaptive pooling operation capable of interpolating between common pooling operations such as mean-, max- or min-pooling.

3. DATA CHALLENGE

In Sec. 2.1, we introduced research on instrument recognition in polyphonic, multi-timbral music. One theme that emerges is that with almost every new publication, a new dataset is released by the authors in an effort to address issues with previous ones. While releasing new datasets is highly encouraged and vital for research in MIR in general, an uncoordinated effort leads to lack of uniformity in the

datasets used. In this section we briefly describe the common datasets for instrument recognition and identify the challenges associated with them.

The IRMAS dataset [5] is a frequently used dataset for predominant instrument recognition. It consists of a separate training and testing set, each containing annotations for 11 predominant instruments. The dataset consists of short excerpts —3 s for training and variable length for testing— of weakly labeled data. One fundamental problem of the IRMAS annotations is that the training set lacks multi-label annotation; this can be problematic for a general use case as instrument co-occurrence is ignored.

The MedleyDB [4] and Mixing Secrets [11] datasets are both multi-track datasets. Due to the availability of instrument-specific stems, strong annotations of instrument activity are available. Thus, these two multi-track datasets provide all the necessary detailed annotations for instrument activity detection and have been used in [12, 25]. These datasets have two disadvantages when training models. First, with a few hundred distinct songs models trained with the data are hardly generalizable. Second, the datasets are not well balanced in terms of either musical genre or instrumentation. However, this may not be a problem if the datasets were larger and the distribution represented the real-world.

Most of these problems were addressed with the release of the OpenMIC dataset [17]. This dataset contains 20,000 10 s clips of audio from different songs across various genres. Each clip is annotated with the presence or absence of one or more of 20 instrument labels. OpenMIC presents a larger sample size as well as a uniform distribution across instruments. It is, however, weakly labeled, i.e., each 10 s clip has instrument presence or absence tags without specific onset and offset times. Due to the nature of weak labels, models cannot be trained using fine-grained instrument activity annotation as done, e.g., in [12, 19]. Additionally, not all clips are labeled with all 20 instruments, i.e., there are missing labels. This complicates the training procedure if models are to predict the presence/absence for all 20 instruments for an input audio clip. Despite their drawbacks, creation of WLDs scales better since weak labels are cheaper to obtain; models capable of exploiting WLDs may thus be vital for the future development of instrument recognition.

4. METHOD

Before describing the model details, we provide a formalization of our approach to the instrument recognition problem in weakly labeled data.

4.1 Pre-Processing

As mentioned in Sect. 3, the OpenMIC dataset consists of 10 s audio clips, each labeled with the presence or absence of one or more of 20 instrument labels. For each audio file in the dataset, the dataset creators also release features extracted from a pre-trained CNN, known as "VGGish" [16]. The VGGish model, based on the VGG architectures for object recognition [30], is trained for audio classification. The model produces a 128-dimensional feature vector for 0.96 s windows of audio with no overlap. The features are ZCA-whitened and quantized to 8-bits. For a 10 s audio file, we obtain a 10×128-dimensional matrix. We also normalize the 8-bit integers to a quantized range of $[0, 1]$.

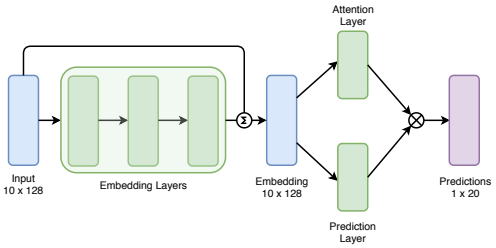


Figure 1: Model Architecture

4.2 Formulation

4.2.1 Multi-Instance Multi-Label Problem

In the most general setting, instrument recognition can be framed as Multi-Instance Multi-Label (MIML) classification [38, 42, 43]. Under this setting, we are given a training dataset $\{(\mathbf{X}_1, \mathbf{Y}_1), \dots (\mathbf{X}_m, \mathbf{Y}_m)\}$ where $\mathbf{X}_i$ is a bag containing r instances $X_i = \{\boldsymbol{x}_{i,1}, \dots \boldsymbol{x}_{i,r}\}$ and $\mathbf{Y}_i = [\boldsymbol{y}_{i,1}, \dots, \boldsymbol{y}_{i,L}] \in \{0, 1\}^L$ is a label vector with L labels with $\boldsymbol{y}_{i,j} = 1$ if any of the instances in $\mathbf{X}_i$ contains label j. In the remainder of this section, we will drop the indices used to reference a specific data point and simply represent a sample from the dataset as $(\mathbf{X}, \mathbf{Y})$. In our case, a bag $\mathbf{X}$ refers to the 10×128-dimensional feature matrix representing one audio clip and each bag contains 10 instances. Our problem is also a Missing Label problem since for a sample $(\mathbf{X}, \mathbf{Y})$, not all $\boldsymbol{y}_j$ are known or annotated (compare Sect. 3).

In our experiments, we assume that all labels can be independently predicted for each instance. Under this assumption, the MIML problem decomposes into L (20 for OpenMIC dataset) instantiations of Multi-Instance Learning (MIL) [8,41] problems, one for each label in the dataset.

Note that exploiting label-correlation in multi-label classification has shown to significantly improve the classification performance [28, 28, 34, 40]. However, exploring ways to incorporate label-correlation for instrument recognition in the OpenMIC dataset has the additional challenge of missing and sparse labels [3]. Also, as is prevalent in most MIL approaches [8], we assume independence among different instances in a bag. Neighboring instances in a bag representing a polyphonic music snippet will, however, likely have high correlation. Relaxing the aforementioned assumptions about independence among labels, and instances in a bag is left for future work since in our current work, we focus on the impact of attention for aggregating instance-level predictions.

4.2.2 Multi-Instance Learning

In the MIL setting, a bag label is produced through a score function $S(\mathbf{X})$. Under the assumption of independence

among instances, $S(\mathbf{X})$ admits a parametrization of the form

$$S(\mathbf{X}) = \mu\Big(f(\boldsymbol{x}) \Big) \tag{1}$$

where $f(.)$ is a score function for an instance $\boldsymbol{x}$, and $\mu(.)$ is a permutation-invariant aggregation operation for instance scores $f(\boldsymbol{x})$ [37]. This parameterization induces a natural approach to classify a bag of instances: (i) to produce scores for each instance in the bag using an instance-level scoring function $f(\boldsymbol{x})$, and (ii) to aggregate the scores across different instances in the bag using the aggregation function $\mu(.)$. In our approach, we use a classification function to produce instance-level scores $f(\boldsymbol{x})$, which are essentially the probabilities of a label being present for each instance. The max and avg functions are two commonly used permutation-invariant operations to aggregate instance-level scores to bag-level scores. McFee et al. found that *learning* an aggregation operation, however, significantly improved performance over fixed predefined operations like max and avg. We choose to represent our aggregation operation $\mu(.)$ as a weighted sum of instance-level scores, i.e.,

$$S(\mathbf{X}) = \sum_{\boldsymbol{x} \in \mathbf{X}} w_{\boldsymbol{x}}\ f(\boldsymbol{x}) \tag{2}$$

where $w_{\boldsymbol{x}}$ is a learnable weight for instance $\boldsymbol{x}$. Our choice of $f(.)$ and $\mu(.)$ has the two advantages that (i) the resulting $S(.)$ is the probability of a label being present in the bag and can be directly used to make a prediction and (ii) the learned weights for each instance add interpretability to the MIL models by encoding beliefs placed by the MIL model on the score of each instance.

4.2.3 Attention Mechanism

The learnable aggregation operation is equivalent to attention. Given a bag $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_r\}$ of r instances, the instance level scoring function $f(.)$ produces a bag $\{f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_r)\}$ of instance scores. The bag-level score $S(\mathbf{X})$ is then computed using Eq. (2).

We further impose the restriction that instance weights w_x should sum to 1, i.e., $\sum_{\boldsymbol{x} \in \mathbf{X}} w_{\boldsymbol{x}} = 1$. This ensures that the aggregation operation is invariant to the size of the bag, thus allowing the model to work with sound clips of arbitrary length. Furthermore, this normalization leads to a probabilistic interpretation of the instance weights which can then be used to infer the relative contribution of each instance towards $S(\mathbf{X})$. For an instance $\boldsymbol{x} \in \mathbf{X}$, the weight $w_{\boldsymbol{x}}$ is thus parametrized as

$$w_{\boldsymbol{x}} = \frac{\sigma(\boldsymbol{v}^\top h(\boldsymbol{x}))}{\sum_{\boldsymbol{x}' \in \mathbf{X}} \sigma(\boldsymbol{v}^\top h(\boldsymbol{x}'))} \tag{3}$$

where $h(\boldsymbol{x})$ is a learned embedding of the instance $\boldsymbol{x}$, $\boldsymbol{v}$ are the learned parameters of the attention layer, and $\sigma(.)$ is the *sigmoid* non-linearity.

This corresponds to the attention mechanism traditionally used in sequence modeling [2, 35]. For example, Raffel and Ellis [29] produced attention weights in a manner similar to Eq. (3) with the only difference being the use of *softmax* operation to perform normalization of weights across the instances.

4.3 Model Architecture

Computing bag-level scores $S(.)$ involves computing instance-level scores $f(.)$ and aggregating the scores across instances using a learned set-operator $\mu(.)$ which performs weighted averaging with the weights computed with Eq. (3). For our experiments, we represent the scores, both instance level $f(.)$ and bag-level $S(.)$, as the probability estimate of the instance or bag being a positive sample for a given label. We first pass each instance $\boldsymbol{x}$ through an embedding network of three fully connected layers to project each instance to a suitable embedding space. Next, instance-level scores $f(.)$ are computed from the output of embedding network with another fully connected layer. Similarly, attention weights are computed by normalizing the outputs of a fully connected layer, the weights of which correspond to parameters $\boldsymbol{v}$ in Eq. (3). Note that the output dimension of these two parallel fully connected layers is equal to the number of labels, i.e., 20. Figure 1 illustrates the model architecture. In the embedding layer, the number of hidden units is 128. We also found that adding a skip connection from the input to the final embedding stabilized the training across different random seeds. We use batch normalization, ReLU activations, and a dropout of 0.6 after each embedding layer. The model has 55336 learnable parameters.

4.4 Loss Function and Training Procedure

Our model performs a multi-label classification over 20 labels given an input. However, as we point out earlier, the OpenMIC dataset does not contain all labels for each instance. This leads to missing ground truth labels for training with loss functions such as binary cross-entropy (BCE). To account for this, we utilize the partial binary cross-entropy (BCE_p) loss function introduced for handling missing labels [7]:

$$\begin{aligned} \mathrm{BCE}_p(\boldsymbol{y}, \boldsymbol{q}) &= \frac{g(p_y)}{L} \sum_{l \in L^o} \boldsymbol{y}_l \log \boldsymbol{q} + (1 - \boldsymbol{y}_l) \log(1 - \boldsymbol{q}) \\ g(p_y) &= \alpha p_y^\gamma + \beta \end{aligned} \tag{4}$$

Here $g(p_y)$ is a normalization function, p_y is the proportion of observed labels for the current data point, L is the total number of labels, L^o is the list of observed labels for the input data, $\boldsymbol{y}_l \in \{0, 1\}$ is the ground truth (absent or present) for label l, and $\boldsymbol{q}$ is the model's probability output for the label l being present in the input data $\mathbf{X}$. The hyperparameters in Eq. (4) are α, β, and γ. Note that in the absence of $g(p_y)$, data points with few observed labels will have a lower contribution in loss computation than those with several observed labels. This is undesirable behavior and the inclusion of a normalization factor, dependent on the proportion of observed labels, is important. Therefore, we set α, β, and γ to 1, 0, and -1, respectively. This normalizes the loss for a data point by the number of observed labels and is equivalent to only computing the loss for observed labels.

Finally, the Adam optimization algorithm [20] is used for training with a batch size of 128 and learning rate of $5e^{-4}$ for 250 epochs. We checkpoint the model at the epoch with the best validation loss.

5. EVALUATION

In this section we describe the experimental setup including the dataset, the baseline methods, and evaluation metrics.

5.1 Dataset

We use the OpenMIC dataset for the experiments in this paper. In addition to the audio and label annotations, the data repository contains pre-computed features extracted from the publicly available VGG-ish model for audio classification. We utilize those features in our experiments to strictly focus on handling the weak labels and avoid further complexity by having to learn features from the raw data or spectrogram representations. Pilot experiments for feature learning showed that CNN architectures based on state-of-the-art instrument recognition models were unable to outperform the baseline model of 20 instrument-wise random forest classifiers trained using the pre-computed features. For reproducibility and comparability, we utilize the training and testing split released with the dataset. Additionally, we randomly sample and separate 15% data from the training split to create a validation set.

5.2 Experiments

We compare the attention model (ATT) with the following models:

1. RF_BR: This model is the baseline random forest model in [17]. A binary-relevance transformation is applied to convert the multi-label classification task into 20 independent binary classification tasks [39].
2. FC: A 3-layer fully connected network trained to predict the presence or absence of all instruments for a given data instance. Here, the input features of dimension 10×128 are flattened into a single feature vector for classification. Dropout is used for regularization and the Leaky ReLU (0.01 slope) is used. The model has 986772 parameters.
3. FC_T: This model serves as an ablation study to observe the benefits of the attention mechanism. FC_T uses the same embedding layer as ATT. However, the aggregation of predictions in time is simply performed with average-pooling. The model has 52116 parameters.
4. RNN: A 3-layer bi-directional gated recurrent unit model with 64 hidden units per direction. The model processes the input features and produces a single embedding which is then fed to a classifier for all 20 instruments. The model has 226068 parameters.

Source code for the Pytorch implementation of the neural network models is publicly available. [1] For each model, we train 10 randomly initialized instances with different random seeds and compute the classification metrics for each.

[1] https://github.com/SiddGururani/AttentionMIC

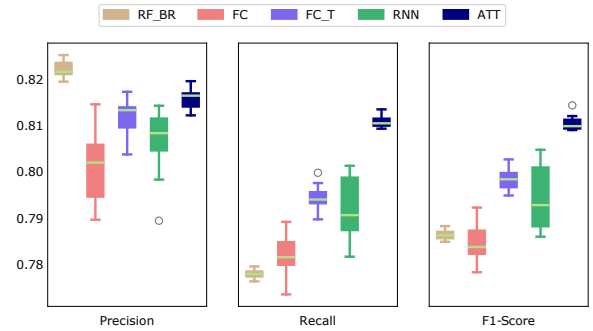


Figure 2: Precision, recall, and F1-score for different models

This gives us a distribution of each model's performance. One benefit of ATT over the FC and RNN models is its small size. Both the ATT and FC_T utilize weight-sharing for embedding instances from the bags. This leads to significantly fewer learnable parameters compared to FC and RNN while performing better than both of these models.

5.3 Metrics

While the total number of clips per instrument label in the OpenMIC dataset is balanced, the number of positive and negative examples is not well balanced for each instrument label. Therefore, we separately compute the precision, recall and F1-score for the positive and negative class. Thereafter, we compute the macro-average of these metrics to report the final instrument-wise metrics, meaning that positive and negative examples are weighted equally. We call these the instrument-wise precision, recall, and F1-score. Additionally, to measure the overall performance of a classifier, we macro-average the instrument-wise precision, recall and F1-score. We use a fixed threshold of 0.5 to convert the outputs into binary predictions for computing the classification metrics.

6. RESULTS AND DISCUSSION

Figure 2 shows the overall performance of ATT compared to the baseline models with box plots for the macro-averaged precision, recall, and F1-score. Additionally, we compare the instrument-wise F1-score for each model in Figure 3. Note that we only show the mean instrument-wise F1-score across 10 seeds in Figure 3 for improved visibility.

We observe that while the attention mechanism does not lead to an improvement in precision compared to the other models, the recall is improved significantly and consequently the F1-score is also improved. We also observe that ATT performs better than RF_BR in almost every instrument label, especially for the labels with high positive-negative class imbalance, such as clarinet, flute, and organ. This ties to the observation made about improved recall, as ATT is able to overcome this imbalance possibly due to the ability to localize the relevant instances for the minority class. In the case of an imbalanced instrument label, the recall for the minority class greatly suffers for RF_BR.

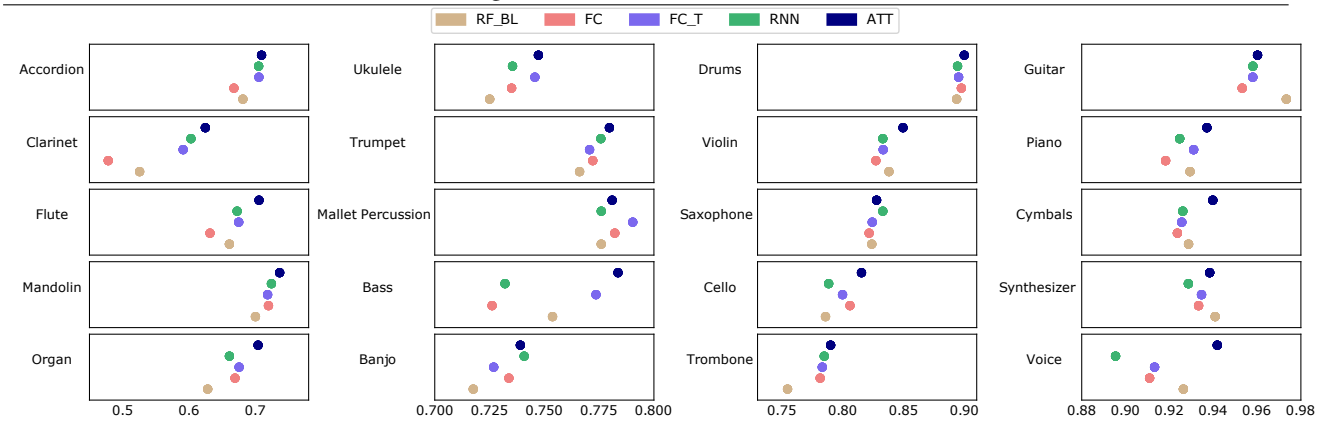


Figure 3: Instrument-wise F1-scores

While this problem is easily mitigated in standard multi-class problems by using balanced sampling, it is difficult to address with multi-label data. Comparing to FC_T, we can attribute the better performance of ATT to better aggregation of instance-level predictions. FC_T is essentially the same model as ATT using mean pooling instead of attention, and ATT outperforms it for most instrument classes, especially the generally more difficult to classify instruments. The RNN model also beats the RF_BR baseline. In polyphonic music, the instances in a bag are structured and highly correlated and hence using a recurrent network to model the temporal structure in the instance sequence leads to a powerful embedding of the bag, incorporating useful information from each instance.

We visualize the attention weights for two example clips in Figure 4. The left clip is from the test set and starts with the vocals fading out until 2 seconds. From 5 second onwards, the vocals grow in loudness until the end of the clip. The violin plays throughout but is the pre-dominant instrument only for a few seconds between 3 and 6 seconds, as visualized in the corresponding attention weights as well. The right clip is from the training set and contains vocals starting from 6 second onwards. The attention weights for vocals directly coincides with that. It is interesting to note that the annotation for vocals was missing for this clip.

7. CONCLUSION

Weakly labeled datasets for instrument recognition in polyphonic music are easier to develop or annotate than strongly labeled datasets. This calls for a paradigm shift in the approaches towards supervised learning approaches better suited for weakly labeled data. We formulate the instrument recognition task as a MIML problem and introduce an attention-based model, evaluated on the OpenMIC dataset for 20 instruments, and compared against several other baseline models including: (i) binary-relevance random forest, (ii) fully connected networks, and (iii) recurrent neural networks, We find that the attention mechanism improves the overall performance as well as the instrument-wise performance of the model while keeping the model light-weight. The example visualizations show that the model indeed is

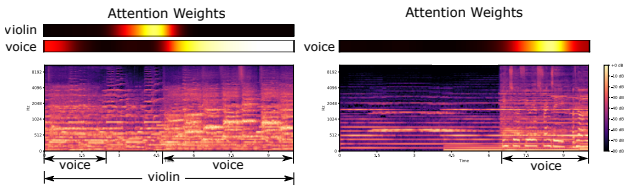


Figure 4: Attention Weight Visualization: The horizontal bars above the mel-spectrogram represent the attention weights across the instances of the clip for the respective instruments.

able to attend to relevant sections on a clip.

Some of the assumptions made in the formulation of the MIML problem are strong and may be worth relaxing due to the nature of musical data. We plan to further explore the task of instance-level embeddings using recurrent networks or using self-attention mechanisms as used in Transformer networks [35]. Additionally, we plan to address the problem of missing labels or label sparsity in the OpenMIC dataset using the curriculum learning-based methods proposed in [7]. Our concern is that the dataset is not large enough with enough labels for strictly supervised learning approaches to significantly improve the results much further than what we achieve with the attention mechanism, and we therefore plan to tackle the problem from other angles, such as handling missing labels or data augmentation.

8. ACKNOWLEDGEMENTS

This research is partially funded by Gracenote, Inc. We thank them for their generous support and meaningful discussions. We also thank Nvidia Corporation for their donation of a Titan V awarded as part of the GPU grant program.

9. REFERENCES

[1] Sharath Adavanne, Pasi Pertilä, and Tuomas Virtanen. Sound event detection using spatial features and convolutional recurrent neural network. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 771–775, New Orleans, LA, USA, 2017.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. of the International Conference on Learning Representations, (ICLR)*, San Diego, CA, USA, 2015.

[3] Wei Bi and James Kwok. Multilabel classification with label correlations and missing labels. In *Proc. of the AAAI Conference on Artificial Intelligence*, pages 1680–1686, Québec City, Québec, Canada, 2014.

[4] Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. Medleydb: A multitrack dataset for annotation-intensive MIR research. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 155–160, Taipei, Taiwan, 2014.

[5] Juan J Bosch, Jordi Janer, Ferdinand Fuhrmann, and Perfecto Herrera. A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 559–564, Porto, Portugal, 2012.

[6] Emre Cakır, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, 25(6):1291–1303, 2017.

[7] Thibaut Durand, Nazanin Mehrasa, and Greg Mori. Learning a deep convnet for multi-label classification with partial labels. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 647–657, Long Beach, CA, USA, 2019.

[8] James Foulds and Eibe Frank. A review of multi-instance learning assumptions. *The Knowledge Engineering Review*, 25(1):1–25, 2010.

[9] Ferdinand Fuhrmann. *Automatic musical instrument recognition from polyphonic music audio signals*. PhD thesis, Universitat Pompeu Fabra, 2012.

[10] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780, New Orleans, LA, USA, 2017.

[11] Siddharth Gururani and Alexander Lerch. Mixing secrets: A multi-track dataset for instrument detection in polyphonic music. In *Late Breaking Demo (Extended Abstract), Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017.

[12] Siddharth Gururani, Cameron Summers, and Alexander Lerch. Instrument activity detection in polyphonic music using deep neural networks. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 569–576, Paris, France, 2018.

[13] Yoonchang Han, Jaehun Kim, and Kyogu Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 25(1):208–221, 2017.

[14] Yoonchang Han, Subin Lee, Juhan Nam, and Kyogu Lee. Sparse feature learning for instrument identification: Effects of sampling and pooling methods. *The Journal of the Acoustical Society of America*, 139(5):2290–2298, 2016.

[15] Perfecto Herrera-Boyer, Geoffroy Peeters, and Shlomo Dubnov. Automatic classification of musical instrument sounds. *Journal of New Music Research*, 32(1):3–21, 2003.

[16] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss, and Kevin Wilson. CNN architectures for large-scale audio classification. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, USA, 2017.

[17] Eric Humphrey, Simon Durand, and Brian McFee. Openmic-2018: An open dataset for multiple instrument recognition. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 438–444, Paris, France, 2018.

[18] Yun-Ning Hung, Yi-An Chen, and Yi-Hsuan Yang. Multitask learning for frame-level instrument recognition. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 381–385, Brighton, UK, 2019.

[19] Yun-Ning Hung and Yi-Hsuan Yang. Frame-level instrument recognition by timbre and pitch. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 135–142, Paris, France, 2018.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conference on Learning Representations, (ICLR)*, San Diego, CA, USA, 2015.

[21] Tetsuro Kitahara, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. Instrument identification in polyphonic music: Feature weighting to minimize influence of sound overlaps. *EURASIP Journal on Applied Signal Processing*, 2007(1):155–155, 2007.

[22] Qiuqiang Kong, Yong Xu, Wenwu Wang, and Mark Plumbley. Audio set classification with attention model: A probabilistic perspective. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 316–320, Calgary, Canada, 2018.

[23] Qiuqiang Kong, Changsong Yu, Turab Iqbal, Yong Xu, Wenwu Wang, and Mark D. Plumbley. Weakly labelled audioset classification with attention neural networks. *CoRR*, abs/1903.00765, 2019.

[24] Anurag Kumar and Bhiksha Raj. Audio event detection using weakly labeled data. In *Proc. of the 24th ACM International Conference on Multimedia (ACMMM)*, pages 1038–1047, Amsterdam, The Netherlands, 2016.

[25] Peter Li, Jiyuan Qian, and Tian Wang. Automatic instrument recognition in polyphonic music using convolutional neural networks. *CoRR*, abs/1511.05520, 2015.

[26] Vincent Lostanlen, Joakim Andén, and Mathieu Lagrange. Extended Playing Techniques: The Next Milestone in Musical Instrument Recognition. In *Proc. of the International Conference on Digital Libraries for Musicology (DLfM)*, pages 1–10, Paris, France, 2018.

[27] Brian McFee, Justin Salamon, and Juan Pablo Bello. Adaptive pooling operators for weakly labeled sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, 26(11):2180–2193, 2018.

[28] Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, Tao Mei, and Hong-Jiang Zhang. Correlative multi-label video annotation. In *Proc. of the ACM International Conference on Multimedia (ACMMM)*, pages 17–26, Augsburg, Germany, 2007.

[29] Colin Raffel and Daniel P. W. Ellis. Feed-forward networks with attention can solve some long-term memory problems. *CoRR*, abs/1512.08756, 2015.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. of the International Conference on Learning Representations, (ICLR)*, San Diego, CA, USA, 2015.

[31] Dan Stowell, Dimitrios Giannoulis, Emmanouil Benetos, Mathieu Lagrange, and Mark D. Plumbley. Detection and classification of acoustic scenes and events. *IEEE Transactions on Multimedia*, 17(10):1733–1746, 2015.

[32] Takumi Takahashi, Satoru Fukayama, and Masataka Goto. Instrudive: A Music Visualization System Based on Automatically Recognized Instrumentation. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 561–568, Paris, France, 2018.

[33] John Thickstun, Zaïd Harchaoui, and Sham Kakade. Learning features of music from scratch. In *Proc. of the International Conference on Learning Representations, (ICLR)*, Toulon, France, 2017.

[34] Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis P Vlahavas. Multi-label classification of music into emotions. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 325–330, Philadelphia, PA, USA, 2008.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008. Curran Associates, Inc., Long Beach, CA, USA, 2017.

[36] Chih-Wei Wu and Alexander Lerch. From labeled to unlabeled data – on the data challenge in automatic drum transcription. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

[37] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3391–3401. Curran Associates, Inc., Long Beach, CA, USA, 2017.

[38] Zheng-Jun Zha, Xian-Sheng Hua, Tao Mei, Jingdong Wang, Guo-Jun Qi, and Zengfu Wang. Joint multi-label multi-instance learning for image classification. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Anchorage, AK, USA, 2008.

[39] Min-Ling Zhang, Yu-Kun Li, Xu-Ying Liu, and Xin Geng. Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, 12(2):191–202, 2018.

[40] Min-Ling Zhang and Kun Zhang. Multi-label learning by exploiting label dependency. In *Proc. of the ACM International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)*, pages 999–1008, Washington, DC, USA, 2010.

[41] Zhi-Hua Zhou and Min-Ling Zhang. Neural networks for multi-instance learning. In *Proc. of the International Conference on Intelligent Information Technology*, pages 455–459, Beijing, China, 2002.

[42] Zhi-Hua Zhou and Min-Ling Zhang. Multi-instance multi-label learning with application to scene classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1609–1616. Curran Associates, Inc., Vancouver, BC, Canada, 2007.

[43] Zhi-Hua Zhou, Min-Ling Zhang, Sheng-Jun Huang, and Yu-Feng Li. Multi-instance multi-label learning. *Artificial Intelligence*, 176(1):2291 – 2320, 2012.

MIDI–SHEET MUSIC ALIGNMENT USING BOOTLEG SCORE SYNTHESIS

Thitaree Tanprasert[1*] **Teerapat Jenrungrot**[1*] **Meinard Müller**[2] **TJ Tsai**[1]
[1] Harvey Mudd College, Claremont, CA USA
[2] International Audio Laboratories Erlangen, Germany
{ttanprasert, mjenrungrot, ttsai}@hmc.edu, meinard.mueller@audiolabs-erlangen.de

ABSTRACT

MIDI–sheet music alignment is the task of finding correspondences between a MIDI representation of a piece and its corresponding sheet music images. Rather than using optical music recognition to bridge the gap between sheet music and MIDI, we explore an alternative approach: projecting the MIDI data into pixel space and performing alignment in the image domain. Our method converts the MIDI data into a crude representation of the score that only contains rectangular floating notehead blobs, a process we call bootleg score synthesis. Furthermore, we project sheet music images into the same bootleg space by applying a deep watershed notehead detector and filling in the bounding boxes around each detected notehead. Finally, we align the bootleg representations using a simple variant of dynamic time warping. On a dataset of 68 real scanned piano scores from IMSLP and corresponding MIDI performances, our method achieves a 97.3% accuracy at an error tolerance of one second, outperforming several baseline systems that employ optical music recognition.

1. INTRODUCTION

This paper tackles the problem of MIDI–sheet music synchronization. Given a symbolic music representation and its scanned sheet music, the goal is to determine the alignment between each time instant in the symbolic representation and its corresponding pixel location in the sheet music.

Many tools for alignment have been developed in the context of audio synchronization. The goal of audio synchronization is to find the temporal alignment between two different audio recordings of the same musical piece. The main technique used to solve this alignment problem is called dynamic time warping (DTW) [3] [10] [15]. DTW consists of four steps: (1) extracting a sequence of features from both audio recordings, (2) computing a cost matrix C, where C_{ij} indicates the dissimilarity between the i^{th} frame of recording 1 and the j^{th} frame of recording 2, (3) using dynamic programming to calculate a cumulative cost matrix D and backtrace matrix B, where D_{ij} indicates the optimal path score from $(0, 0)$ to (i, j) given a set of allowable transitions and transition weights, and where B_{ij} indicates the penultimate element in the optimal path, and (4) backtracing through B to determine the lowest cost path through the entire matrix. Many works have proposed ways to extend or improve upon this basic method, including doing the time warping in an online fashion [4] [18], estimating the alignment at multiple granularities [22] [26], handling repeats and jumps [12], handling subsequences or partial alignments [20] [28], dealing with fixed memory constraints [23], and utilizing multiple recordings [1] [31].

Several previous works have studied the problem of finding correspondences between audio and sheet music. There are two general approaches to the problem. The first approach is to use an existing optical music recognition (OMR) system to convert the sheet music into a symbolic (MIDI-like) representation, to collapse the pitch information across octaves to get a chroma representation, and then to compare this representation to chroma features extracted from the audio. This approach has been applied to synchronizing audio and sheet music [2] [16] [27], identifying audio recordings that correspond to a given sheet music representation [13], and finding the audio segment corresponding to a fragment of sheet music [11]. A different approach has been explored in recent years: convolutional neural networks (CNNs). This approach attempts to learn a multimodal CNN that can embed a short segment of sheet music and a short segment of audio into the same feature space, where similarity can be computed directly. This approach has been explored in the context of online sheet music score following [5], sheet music retrieval given an audio query [6] [7], and offline alignment of sheet music and audio [7]. Dorfer et al. [8] have also recently shown promising results formulating the score following problem as a reinforcement learning game.

In this paper, we consider the task of MIDI–sheet music synchronization, which can be seen as a variant of the audio–sheet music synchronization scenario. As symbolic (MIDI-like) representations often serve as a bridge between audio and sheet music, MIDI–sheet music synchronization can be regarded as an important intermediate step for more general cross-modal alignment. As men-

[*] The first two authors had equal contribution.

tioned above, traditional approaches typically apply OMR to bridge the modality gap—a step that often introduces severe errors. On the other side, deep learning approaches that try to extract shared feature representations directly from waveforms (audio) and images (sheet music) are promising, but are still in their infancy [21]. As the main contribution of this paper, we introduce an approach that avoids an explicit OMR step by working with an explicitly known sparse, binary representation in the image domain. As we will demonstrate, we can convert both sheet image data and MIDI data into this representation using simple logic coupled with a notehead detector. Based on this common binary representation, we show how the alignment problem can then be solved using a simple variant of DTW. In a sense, our approach mimics a deep learning approach, but explicitly introduces a mid-level representation. We hope that our contribution not only sheds a new light into cross-modal alignment, but may also serve as a non-trivial baseline approach for future fully automated procedures.

The paper is organized as follows. Section 2 describes the proposed algorithm. Section 3 explains the experimental results. Section 4 provides an analysis of system performance. Section 5 concludes the work.

2. SYSTEM DESCRIPTION

There are two inputs to our system: a MIDI file and its corresponding sheet music. Similar to recent work [6–8], we assume that the sheet music is presented as a sequence of image strips, where each image strip contains a single line of music. We focus exclusively on piano music in this work, so each line of music consists of a single grand staff containing an upper staff (treble clef) and a lower staff (bass clef). The image strips may be different sizes, and the staff lines may appear at a different location on each strip.

Our proposed method has three main steps. The first step is to convert each image strip into a sparse, binary representation in pixel space (A_i in Figure 1). We perform this conversion by applying a notehead detector and filling in the predicted bounding boxes around each detected notehead. This representation is a very crude representation of the score that only contains rectangular floating notehead blobs. Accordingly, we call this a bootleg representation. The second step is to project the MIDI data into the same bootleg space (B_i in Figure 1). We perform this projection by converting MIDI note onsets into floating notehead blobs that are appropriately placed in pixel space. The third step is to align the bootleg representations using a variant of DTW (Figure 3). These three steps are described in detail in the next three subsections. [1]

2.1 Notehead Detection

The first step is to convert each image strip into a bootleg representation. As shown in Figure 1 (left side), we accomplish this by applying a notehead detector and filling in the predicted bounding boxes around each detected notehead. The remainder of this subsection describes our notehead detection.

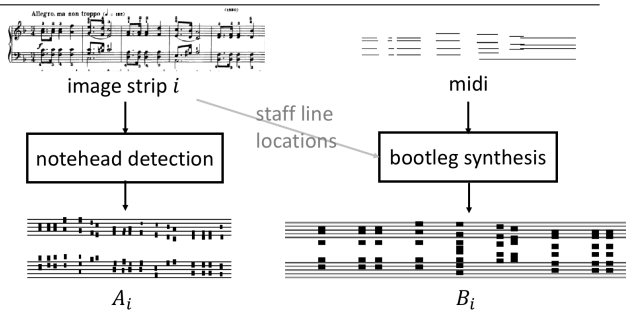


Figure 1. Projecting data to bootleg space. We convert the image strips and MIDI data into a very crude approximation of the sheet music that only contains floating notehead blobs. The staff lines in A_i and B_i are shown as a visual aid, but are not included in the bootleg representation.

Our notehead detector is based on the deep watershed detector recently proposed by Tuggener et al. [30] for musical object detection in sheet music. The deep watershed detector is a fully convolutional network [17] modified to predict three outputs: (a) a quantized energy output map which indicates the likelihood of having an object at each pixel location, (b) a class output map which predicts which type of object is present at each pixel location (e.g. filled notehead, staff line, treble clef, sharp, quarter rest, etc.), and (c) a bounding box output map which indicates the width and height of an object at that pixel location. Figure 2 illustrates the overall architecture of the deep notehead detection network. The reader is referred to [30] for more details. In [29] and [30], Tuggener et al. show that fully convolutional networks are more suitable for semantic segmentation and detection of tiny objects in sheet music, tasks where (large) object detection methods like Fast R-CNN [14], Faster R-CNN [25], and YOLO [24] fail miserably.

We trained our network on the DeepScores dataset [29]. This dataset contains approximately 300,000 full pages of synthetically generated musical scores and pixel-level ground truth labels for 124 different symbol classes. The inputs to the network are 500x500 grayscale image patches that are randomly sampled from the full page images. The loss function is a linear combination of the losses from the quantized energy output map (cross entropy loss), class output map (cross entropy loss), and bounding box output map (mean squared error).

After training on DeepScores, we fine-tune the network on real scanned sheet music. For fine-tuning, we manually annotated the location and type of approximately 2200 noteheads in 30 different pages of piano music downloaded from IMSLP. [2] These 30 pages of music were selected to maximize diversity across composers and music publishers, and they are a completely separate set from the data used to evaluate alignment. Because we only care about detecting noteheads, we disregard all other musical objects

[1] Our code and data are available at `https://github.com/ttanprasert/sheet-midi-sync`.

[2] `https://imslp.org`

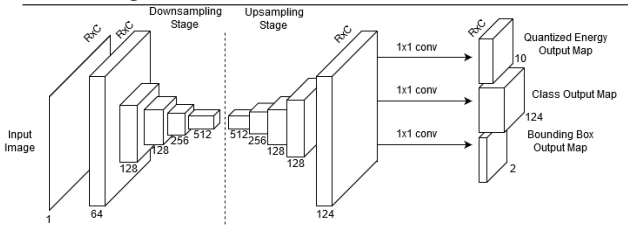


Figure 2. Architecture of the deep watershed notehead detector. The number below each layer indicates the number of feature maps. In the downsampling and upsampling stages, the length and width of the feature maps change by a factor of two in each successive layer. We train on the DeepScores dataset [29] and fine-tune on a small set of manually labeled noteheads in real scanned music.

in the fine-tuning process. Because the real scanned music contains a variety of font sizes, we scale each input image to match the staff line spacing in the DeepScores data. After fine-tuning, the notehead detector achieves a training mean average precision (mAP) of 0.4201 for all notehead types (black notehead, half notehead, and whole notehead). For reference, in normal object detection tasks (not tiny objects), the state-of-the-art mAP is around 0.4 to 0.6. [3] Figure 5 (top half) shows an example of the notehead detector predictions on a section of Brahms Intermezzo Op. 117 No. 2.

2.2 Bootleg Synthesis

The second step is to convert the MIDI data into a bootleg representation. As shown in Figure 1 (right side), we accomplish this by converting note onsets into appropriately placed floating notehead blobs. This process consists of three key parts.

The first part is determining the staff line coordinate system. For each image strip, we would like to determine the location of the staff lines in the upper staff and lower staff. We can accomplish this by computing the row sum of image pixels, convolving the result with comb filters of various sizes (each containing 5 regularly spaced impulses), and identifying the comb filter that yields the strongest response at two non-overlapping staff locations. This gives us the staff line coordinate system for the upper and lower staves.

The second part is synthesizing and placing noteheads. Given the coordinate system from an image strip, we convert each MIDI note onset into one or more floating rectangular noteheads. Note that there is ambiguity when converting from a MIDI note number to a location on a staff. For example, a MIDI note number of 68 might appear as a G-sharp or an A-flat, which correspond to two different staff locations. To handle this ambiguity, we can place a larger-than-normal rectangular notehead which overlaps both possible locations. Furthermore, since notes in the middle register could appear in the right hand or left hand staves, we can simply place two different floating noteheads at both possible locations. In the visualization of B_i in Figure 1, for example, you can see that the first chord contains a C4, which produces a notehead in both the upper and lower staves.

[3] http://cocodataset.org/#detection-leaderboard

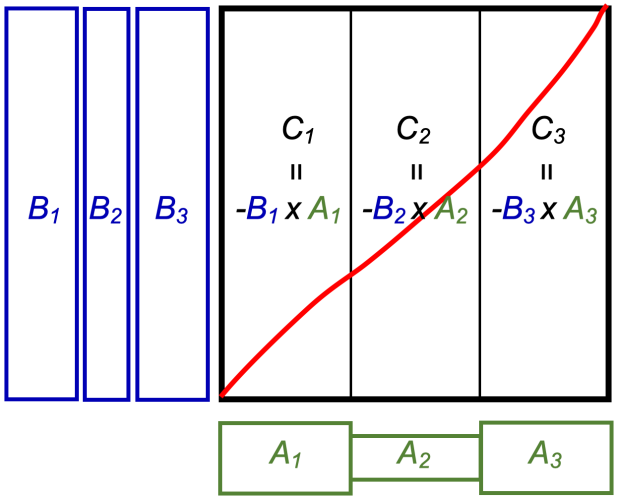


Figure 3. Aligning MIDI-generated bootleg scores with the sheet image-generated bootleg strips. Each bootleg strip A_i is compared to its corresponding bootleg score B_i to yield a cost matrix block C_i. Note that each B_i is a MIDI-generated bootleg score of the *entire* piece projected onto the staff line coordinate system of strip A_i. The red line indicates an alignment path.

The third part is handling timing issues. One issue is the choice of sampling rate. When converting the MIDI data into the bootleg representation, we must choose how much time corresponds to a single pixel column. To avoid extreme time warping, we select this parameter to ensure that the bootleg representation is approximately the same length as the image strips concatenated end-to-end. Another issue is shortening long pauses. When there is a fermata in the score, for example, the MIDI performance may slow down in tempo by a factor of three or four. The sheet music, however, does not reflect this, i.e., the length of the measure in pixels is not elongated by a factor of three or four. To mitigate this issue, we simply shorten long gaps greater than a fixed threshold to the length of the threshold. In experiments, we find that system performance is relatively insensitive to this threshold (across an order of magnitude).

Because each sheet image strip A_i has a different size and a different staff line coordinate system, we generate one bootleg score B_i of the *entire* MIDI performance for each image strip A_i. In other words, B_i is the bootleg score representation of the entire MIDI performance projected onto the staff line coordinate system of image strip A_i. A schematic illustration of this process is shown in Figure 3 for the case of three image strips A_1, A_2, and A_3. Note that the duration of each B_i ($i \in \{1, 2, 3\}$) semantically corresponds to the total duration of the concatenation $A_1 A_2 A_3$, while the pixel height of each B_i matches the height of image strip A_i.

2.3 Block DTW

The third step is to determine the alignment between the bootleg representations. Figure 3 shows a graphical depiction of this process. In this example, the sheet music contains three image strips A_1, A_2, and A_3. The alignment is carried out in two substeps.

The first substep is to calculate the cost matrix (C_i) between each bootleg image strip (A_i) and its corresponding MIDI-generated bootleg score (B_i). In choosing a suitable cost metric, we must consider the nature of the bootleg representations. The MIDI-generated bootleg score will have many redundant notes, where (for example) a C4 will appear in both the left and right hand staff systems in order to handle both possibilities. For this reason, we do not want to penalize the two bootleg representations when they disagree—we only want to reward them when they agree. One simple cost metric that meets this criteria is a negative inner product, i.e. the $(k, \ell)^{\text{th}}$ element of C_i indicates (-1 times) the number of overlapping black pixels in the k^{th} pixel column of B_i and the ℓ^{th} pixel column of A_i. When black ink shows up in the same vertical pixel position in two pixel columns, it will make the cost more negative.

The second substep is to perform global DTW. We assemble the constituent cost matrices C_i into a single global cost matrix (represented as a bold black rectangle in Figure 3). We then apply DTW with step transitions $\{(1,1),(1,2),(2,1)\}$ and corresponding weights $\{2,3,3\}$. This set of step transitions and weights is a robust, common choice in alignment tasks (e.g. see [19]). The lowest cost path through the global cost matrix is the estimated alignment between the MIDI performance and the sheet music. The estimated alignment is shown as a red line in Figure 3.

3. EXPERIMENTS

We now summarize our experiments, where we evaluate and compare our bootleg alignment approach with several baseline approaches. In Section 3.1, we describe our experimental setup introducing the dataset, the manually generated reference annotations, and the evaluation measure. In Section 3.2, we then present and discuss our quantitative results.

3.1 Experimental Setup

The data consists of sheet music scans and MIDI representations for 22 compositions from 8 different composers. The pieces are all for solo piano, contain no repeats or structural jumps, and span a variety of eras, styles, and lengths. The sheet music is downloaded from IMSLP and contains digital scans of printed sheet music editions in the public domain. Note that the choice of using scans of real printed sheet music is a significant departure from other works that focus on synthetically rendered sheet music representations (e.g. [7, 8]). In total, there are 68 sheet music scores. For each composition, we also collected one MIDI performance from online websites. [4] The MIDI performances are symbolic score representations that have been modified to sound like expressive, realistic human performances. Table 1 summarizes the dataset.

Piece	Sh	Meas	Strips
Brahms Fantasia Op117No2	4	86	20,25
Brahms Fantasia Op116No6	3	64	12,15
Chopin Mazurka Op30No2	6	64	9,12
Chopin Mazurka Op63No3	6	76	10,12
Chopin Mazurka Op68No3	6	60	8,12
Clementi Sonata Op36No1 mv3	2	70	8,8
Clementi Sonata Op36No2 mv3	2	111	14,14
Clementi Sonata Op36No3 mv3	2	82	11,11
Debussy Children's Corner mv1	3	76	24,25
Debussy Children's Corner mv3	3	124	23,29
Debussy Children's Corner mv6	3	128	25,25
Mendelssohn Op19No2	5	91	12,14
Mendelssohn Op62No3	3	48	8,10
Mendelssohn Op62No5	3	59	12,13
Mozart Sonata No13 mv3	4	225	42,50
Mozart Sonata No9 mv3	3	269	50,60
Schubert Impromptu Op90No1	2	204	41,60
Schubert Impromptu Op90No3	2	86	42,42
Schubert Op94No2	2	92	17,20
Tchaikovsky The Seasons - Jan	2	102	29,29
Tchaikovsky The Seasons - Jun	2	99	38,40
Tchaikovsky The Seasons - Aug	2	198	24,24

Table 1. Summary of dataset. For each piece, the table indicates the number of sheet music versions (Sh), the number of measures (Meas), and the minimum & maximum number of image strips (i.e. lines of music) across the different sheet music versions.

The ground truth consists of beat-level annotations. For the sheet music, we annotate the horizontal pixel location of a subset of beats in each piece, along with the measure number and image strip number. Because pixel-level annotation of beat locations is very time-consuming, we annotate the beats in $N = 40$ measures equally spaced throughout each piece. For the MIDI performances, we estimate the ground truth beat locations using `pretty-midi` [5] and manually correct any errors.

To evaluate the system performance, we compare the predicted alignment to the ground truth annotations. At the ground truth beat locations in the sheet music, we compare the predicted corresponding times in the MIDI to the ground truth timestamps. Given a fixed error tolerance, we define the error rate to be the percentage of predictions that fall outside of the allowable error tolerance. By considering a range of error tolerances, we can characterize the tradeoff between error rate and error tolerance.

In total, we have 68 MIDI-score pairings, and the resulting alignments are evaluated at $10,913$ ground truth beat locations. Our choice to use scans of real published musical scores has a tradeoff: it places a constraint on the size of the dataset due to the time-consuming nature of annotation, but it is more representative of performance "in the wild" compared to large synthetic datasets like the Multimodal Sheet Music Dataset (MSMD) [9]. We also evaluate

[4] `www.piano-midi.de` and `www.mazurka.org.uk`

[5] `https://github.com/craffel/pretty-midi`

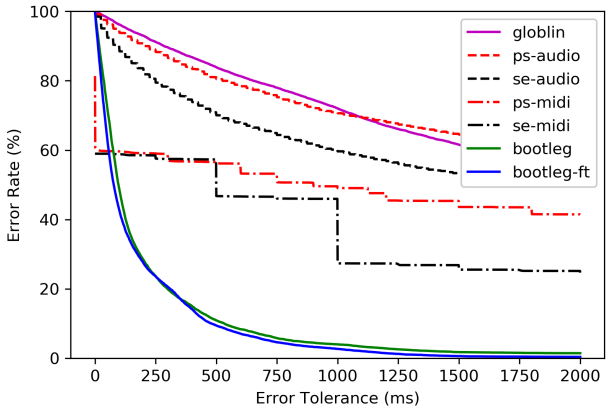


Figure 4. Comparison of baselines to the bootleg system with and without fine-tuning. The legend lists the systems in order of performance from worst to best. For a description of the baseline systems, see Section 3.2.

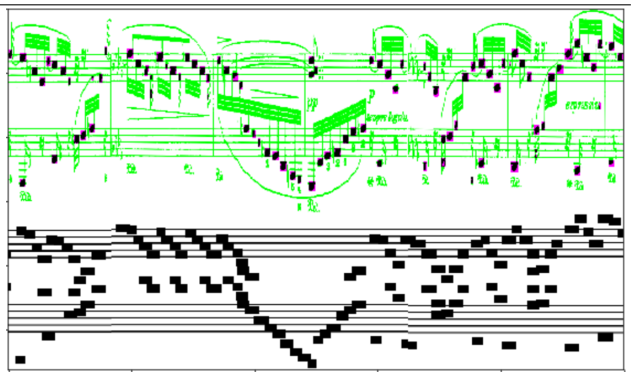


Figure 5. An example of the predicted alignment produced by the bootleg system. The upper half of the figure shows the original score with the detected noteheads overlaid. The bottom half of the figure shows the aligned MIDI-generated bootleg score. This figure is best visualized in color.

our system on MSMD as a point of comparison.

3.2 Results

We compare our bootleg method (with and without fine-tuning the notehead detector) to five baseline systems. The first baseline system ('globlin') simply assumes a global linear correspondence between the concatenated sheet image strips and the MIDI performance. The second and third baseline systems use two different commercial OMR systems (Photoscore [6] and SharpEye [7]) to convert the sheet music to MIDI, synthesize the MIDI to audio, and then perform audio–audio alignment using DTW on chroma features with 25 ms hop size. [8] These baselines are abbreviated as 'ps-audio' and 'se-audio' in Figure 4. The fourth and fifth baseline systems use the same two OMR systems to convert the sheet music to MIDI, estimate the beat locations using `pretty-midi`, and then assume a 1-to-1 correspondence between beat locations in both MIDI files. Note that the OMR system can have many recognition errors but still have perfect alignment if it can simply interpret barlines and beats correctly. These two baselines are abbreviated as 'ps-midi' and 'se-midi' in Figure 4. [9]

There is one important issue to mention about evaluating the OMR baseline systems. Because PhotoScore and SharpEye do not retain the connection between sheet music pixel location and corresponding MIDI time, there is no reliable way to automatically infer ground truth beat locations in the OMR-generated MIDI. Thus, it was necessary to manually annotate the ground truth beat locations in the OMR-generated MIDI on every sheet music score, so that the predicted alignment can be evaluated. This is a very time-consuming process, and clearly not sustainable for large-scale evaluations. However, the benefit of these results is a fair comparison to commercial OMR systems over a reasonably diverse data set.

Figure 4 compares the performance of our proposed bootleg approach (with and without fine-tuning) against the five baseline systems. There are three things to notice about Figure 4.

First, the bootleg method ('bootleg') outperforms the baselines by a wide margin. For example, at 500 ms error tolerance the bootleg systems achieve error rates around 10%, whereas the best performing baseline system achieves an error rate of 47%. Similarly, at 1000 ms error tolerance the bootleg systems achieve $3-4\%$ error rate, whereas the best baseline system has a 27% error rate.

Second, fine-tuning the notehead detector on sheet music scans ('bootleg-ft') shows demonstrable improvement. For example, at 1000 ms error tolerance the fine-tuning improves the error rate from 4.0% to 2.7%, and at 100 ms error tolerance the fine-tuning improves the error rate from 48.8% to 42.8%. We already know that fine-tuning will always improve results. The key observation here is that we can significantly improve results even with an extremely small dataset (2200 noteheads).

Third, the bootleg systems achieve very low asymptotic error rates. Whereas the best-performing baseline system achieves an error rate of 24.7% at a 2000 ms error tolerance, the fine-tuned bootleg system achieves a 0.4% error rate. So, the bootleg alignments are reliable, at least on the data in our experiments.

4. FURTHER ANALYSIS

In this section, we further investigate the proposed system through three different types of analyses.

4.1 Visualization of Alignments

The first method of analysis is to create a visualization that shows the predicted alignment between the bootleg repre-

[6] `https://www.neuratron.com/photoscore.htm`
[7] `http://www.visiv.co.uk`
[8] We also experimented with doing DTW directly on a piano roll representation of the MIDI data, but found that the results were always worse than synthesizing to audio and aligning chroma features.
[9] Note that the stairstep shape of the 'se-midi' system comes from the fact that SharpEye always renders its OMR-generated MIDI at 120 BPM, so that missing or extra beats correspond to errors at integer multiples of 500 ms.

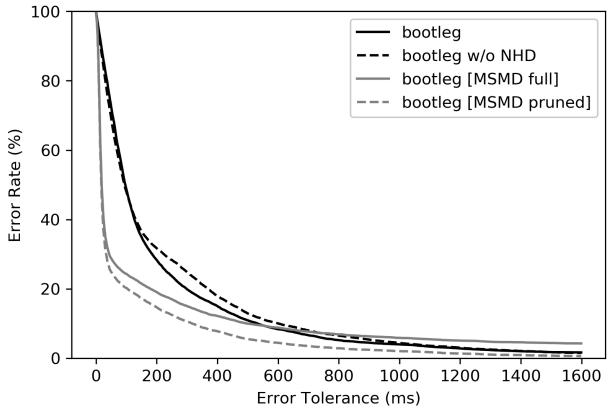


Figure 6. This figure shows two comparisons: (a) The performance of the bootleg system on real scanned sheet music vs. synthetically generated sheet music (see Section 4.2). (b) The performance of the bootleg system with and without notehead detection (see Section 4.3).

sentations. Figure 5 shows an example from a section of Brahms Intermezzo Op. 117 No. 2. In the upper half, the detected notehead regions are overlaid on top of the original score for ease of visualization. The bottom half contains the aligned MIDI-generated bootleg score. We can see that most noteheads are correctly detected, and the two bootleg representations match well.

By looking at example visualizations, we discovered two weaknesses in our system. The first weakness is that the notehead detector performs poorly on half noteheads and whole noteheads. This is due to the fact that the training data was highly imbalanced towards filled noteheads. The second weakness is that the staff line detection occasionally fails, which leads to poor alignments on the entire strip. Interestingly, the system is fairly robust to clef changes, mainly because clef changes usually only occur in one staff but not in both staffs simultaneously.

4.2 Synthetic vs. Real Data

The second analysis investigates the question: “How well does the proposed system work on synthetic data vs. real data?" As mentioned before, we chose to evaluate the baseline systems on real data (i.e. scans of music from IMSLP) rather than synthetically rendered data so that we can assess performance “in the wild." The primary drawback of using real data is that it needs to be manually annotated, which is very costly and limits the practical size of the evaluation dataset. To see how well our system performs on synthetic data, we also ran a large-scale evaluation on the Multimodal Sheet Music Dataset (MSMD) [9].

Figure 6 compares the performance of the bootleg system on the real dataset (black solid line) and the test set from MSMD (gray solid line). We can see that the synthetic data is “easier" to align, especially at lower error tolerances. However, the performance on MSMD leveled off at a much higher error rate for large error tolerances (e.g. 4.4% error at 1500 ms tolerance vs. 1.8% error on the real dataset). Upon further investigation, we found that these errors came from a small set of six pieces that all had one of two peculiar characteristics: (a) they consisted of almost all half or whole notes throughout the entire piece (e.g. Erik Satie’s *Gymnopedies*), or (b) they had very frequent time signature changes (on average every 1.4 measures for the two relevant pieces). The first characteristic will cause the bootleg system to fail because of the imbalance in notehead detection training data, and the second will cause extreme time warping in the alignment stage. While these pieces might be considered extreme or unusual in this regard, they nonetheless provide additional insight into failure modes of the bootleg approach. When we removed this set of six pieces from evaluation, the error curve falls significantly (dotted gray line in Figure 6). We can interpret the gap between the dotted gray curve and the solid black curve as the performance loss when transitioning from synthetic data to real scanned sheet music.

4.3 Importance of Notehead Detection

The third analysis investigates the question: “How much does system performance depend on notehead detection?" To answer this question, we simply removed the notehead detection from our system and directly aligned the MIDI-generated bootleg scores to the raw image strips. We expect the performance to be worse without notehead detection because the raw sheet music will contain many symbols that the MIDI-generated bootleg score does not have: note stems, rests, accidentals, etc. These additional symbols introduce noise that can lead to poor alignments.

Figure 6 compares the performance of the bootleg system with (solid black line) and without (dotted black line) notehead detection on the real dataset. Surprisingly, the system without notehead detection performs only marginally worse, and it approaches approximately the same error rate at high error tolerances. This suggests a way to significantly reduce the complexity of the system without sacrificing much performance.

5. CONCLUSION

We investigate the MIDI–sheet music synchronization problem as an important intermediate step for cross-modal alignment. Because OMR is a difficult task and may not be needed for music alignment, we avoid the need for OMR by introducing a mid-level representation called a bootleg score. We project the MIDI data into bootleg space using the rules of Western musical notation, and we project the sheet music into bootleg space by applying a deep watershed notehead detector. Once the MIDI and sheet music have been projected to this bootleg representation, the alignment can be performed using a simple variant of DTW. We evaluate the proposed system on scans of real published piano scores. Our results indicate that the proposed approach works well for piano music, and it outperforms several baseline systems based on optical music recognition. Our approach may serve as a non-trivial baseline approach for future end-to-end learning approaches.

6. ACKNOWLEDGMENTS

Thitaree Tanprasert and Teerapat Jenrungrot were supported by the Class of 1989 Summer Experiential Learning Fund, the Vandiver Summer Experiential Learning Fund, and the Norman F. Sprague III, M.D. Experiential Learning Fund established by the Jean Perkins Foundation. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research. Meinard Müller was supported by the German Research Foundation (DFG MU 2686/12-1). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

7. REFERENCES

[1] Andreas Arzt and Gerhard Widmer. Real-time music tracking using multiple performances as a reference. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 357–363, Málaga, Spain, 2015.

[2] David Damm, Christian Fremerey, Frank Kurth, Meinard Müller, and Michael Clausen. Multimodal presentation and browsing of music. In *Proc. of the International Conference on Multimodal Interfaces (ICMI)*, pages 205–208, Chania, Crete, Greece, October 2008.

[3] Roger B. Dannenberg and Ning Hu. Polyphonic audio matching for score following and intelligent audio editors. In *Proc. of the International Computer Music Conference (ICMC)*, pages 27–34, San Francisco, USA, 2003.

[4] Simon Dixon. Live tracking of musical performances using on-line time warping. In *Proc. of the 8th International Conference on Digital Audio Effects*, pages 92–97. Citeseer, 2005.

[5] Matthias Dorfer, Andreas Arzt, Sebastian Böck, Amaury Durand, and Gerhard Widmer. Live score following on sheet music images. In *Late Breaking Demo at the International Conference on Music Information Retrieval (ISMIR)*, 2016.

[6] Matthias Dorfer, Andreas Arzt, and Gerhard Widmer. Towards score following in sheet music images. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 789–795, New York City, New York, USA, 2016.

[7] Matthias Dorfer, Andreas Arzt, and Gerhard Widmer. Learning audio-sheet music correspondences for score identification and offline alignment. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 115–122, Suzhou, China, 2017.

[8] Matthias Dorfer, Florian Henkel, and Gerhard Widmer. Learning to listen, read, and follow: Score following as a reinforcement learning game. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 784–791, 2018.

[9] Matthias Dorfer, Jan Hajič Jr., Andreas Arzt, Harald Frostel, and Gerhard Widmer. Learning audio-sheet music correspondences for cross-modal retrieval and piece identification. *Transactions of the International Society for Music Information Retrieval*, 1(1):22–33, 2018.

[10] Sebastian Ewert, Meinard Müller, and Peter Grosche. High resolution audio synchronization using chroma onset features. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1869–1872, Taipei, Taiwan, April 2009.

[11] Christian Fremerey, Michael Clausen, Sebastian Ewert, and Meinard Müller. Sheet music-audio identification. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 645–650, Kobe, Japan, October 2009.

[12] Christian Fremerey, Meinard Müller, and Michael Clausen. Handling repeats and jumps in score-performance synchronization. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 243–248, Utrecht, The Netherlands, 2010.

[13] Christian Fremerey, Meinard Müller, Frank Kurth, and Michael Clausen. Automatic mapping of scanned sheet music to audio recordings. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 413–418, Philadelphia, USA, September 2008.

[14] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[15] Ning Hu, Roger B. Dannenberg, and George Tzanetakis. Polyphonic audio matching and alignment for music retrieval. In *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA, 2003.

[16] Frank Kurth, Meinard Müller, Christian Fremerey, Yoon-Ha Chang, and Michael Clausen. Automated synchronization of scanned sheet music with audio recordings. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 261–266, Vienna, Austria, September 2007.

[17] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[18] Robert Macrae and Simon Dixon. Accurate real-time windowed time warping. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 423–428, 2010.

[19] Meinard Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer, 2015.

[20] Meinard Müller and Daniel Appelt. Path-constrained partial music synchronization. In *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 65–68, Las Vegas, Nevada, USA, April 2008.

[21] Meinard Müller, Andreas Arzt, Stefan Balke, Matthias Dorfer, and Gerhard Widmer. Cross-modal music retrieval and applications: An overview of key methodologies. *IEEE Signal Processing Magazine*, 36(1):52–62, 2019.

[22] Meinard Müller, Henning Mattes, and Frank Kurth. An efficient multiscale approach to audio synchronization. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 192–197, Victoria, Canada, October 2006.

[23] Thomas Prätzlich, Jonathan Driedger, and Meinard Müller. Memory-restricted multiscale dynamic time warping. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 569–573, Shanghai, China, 2016.

[24] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[26] Stan Salvador and Philip Chan. FastDTW: Toward accurate dynamic time warping in linear time and space. In *Proc. of the KDD Workshop on Mining Temporal and Sequential Data*, 2004.

[27] Verena Thomas, Christian Fremerey, Meinard Müller, and Michael Clausen. Linking sheet music and audio – challenges and new approaches. In Meinard Müller, Masataka Goto, and Markus Schedl, editors, *Multimodal Music Processing*, volume 3 of *Dagstuhl Follow-Ups*, pages 1–22. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2012.

[28] T.J. Tsai, Steven Tjoa, and Meinard Müller. Make your own accompaniment: Adapting full-mix recordings to match solo-only recordings. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 79–86, Suzhou, China, 2017.

[29] Lukas Tuggener, Ismail Elezi, Jürgen Schmidhuber, Marcello Pelillo, and Thilo Stadelmann. DeepScores – a dataset for segmentation, detection and classification of tiny objects. In *Proc. of the IEEE International Conference on Pattern Recognition*, 2018.

[30] Lukas Tuggener, Ismail Elezi, Jürgen Schmidhuber, and Thilo Stadelmann. Deep watershed detector for music object recognition. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 271–278, 2018.

[31] Siying Wang, Sebastian Ewert, and Simon Dixon. Robust and efficient joint alignment of multiple musical performances. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):2132–2145, 2016.

MIRDATA: SOFTWARE FOR REPRODUCIBLE USAGE OF DATASETS

Rachel M. Bittner[*1], **Magdalena Fuentes**[*2,3], **David Rubinstein**[1], **Andreas Jansson**[1]
Keunwoo Choi[1], **Thor Kell**[1]
[1] Spotify, USA [2] L2S, CNRS–Univ.Paris-Sud–CentraleSupélec, France
[3] LTCI, Télécom Paris, Institut Polytechnique de Paris, France
[*]Equal contribution

ABSTRACT

There are a number of efforts in the MIR community towards increased reproducibility, such as creating more open datasets, publishing code, and the use of common software libraries, e.g. for evaluation. However, when it comes to datasets, there is usually little guarantee that researchers are using the exact same data in the same way, which among other issues, makes comparisons of different methods on the "same" datasets problematic. In this paper, we first show how (often unknown) differences in datasets can lead to significantly different experimental results. We propose a solution to these problems in the form of an open source library, `mirdata`, which handles datasets in their current distribution modes, but controls for possible variability. In particular, it contains tools which: (1) validate if the user's data (e.g. audio, annotations) is consistent with a canonical version of the dataset; (2) load annotations in a consistent manner; (3) download or give instructions for obtaining data; and (4) make it easy to perform track metadata-specific analysis.

1. INTRODUCTION

Music Information Retrieval (MIR) systems are often software or algorithms which are evaluated and compared based on their performance according to appropriate metrics on chosen datasets. These systems are becoming increasingly complex; reproducing systems presented in academic publications requires access to the software and data [23]. As outlined in [23], some of the common elements of an MIR system are (1) Data (Audio and Annotations) (2) Codecs and Parsing (3) Modeling and (4) Evaluation. The reproducibility of each of these elements poses challenges, but efforts are being made to reduce potential inconsistencies.

For evaluation, different implementations of evaluation metrics can result in substantially different results, motivating the need for `mir_eval` - a common and transparent evaluation software [29]. For modeling, slightly different implementations of the same algorithm can result in very different results [23]. Recently, this has been mitigated by the availability of software with tools for popular MIR tasks. Some examples are `librosa` [25] and `essentia` [6] - tools for MIR related signal processing, simple models and commonly used algorithms; Scikit-Learn [28] - tools for training simple machine learning algorithms; and `madmom` [5] - deep learning and machine learning models for common MIR tasks such as chord recognition, beat and downbeat tracking.

It is very difficult to get licenses to distribute music recordings openly. As a result, the majority of datasets available do not have freely available audio files; the exchange of this data is often done manually, which can result in varying data versions. When working with pairs of audio and annotation files, it is important that the audio files used are the same files that were used to create the annotations. When audio files are released separately from annotations, unknown differences between the original and other versions of the audio can create reproducibility issues. Websites such as Zenodo [1] and Figshare [2] provide permanent hosting and versioning of datasets, increasing reproducibility, but many datasets used in MIR are not available on such websites, and (often unknown) differences in data can adversely affect downstream performance.

Additionally, the annotations that come with each of these datasets exist in a huge variety of formats. Among these formats, some provide very complete information e.g. in the form of a JAMS file [17, 22], while others lack crucial information needed to accurately use the data, such as the time stamps associated with different observations. Most of the time, researchers write their own code for parsing the specific annotation files they use for a particular dataset. This is both inefficient and error-prone; what was found for evaluation and modeling is also true for data parsing: small differences in annotation loading code can result in huge differences in results downstream. Finally, the pairing of audio and annotation files is often done manually each time, usually by matching on filename substrings. In addition to being cumbersome, this can also lead to mismatched audio and annotation files.

To summarize, obtaining datasets and writing code to

[1] `https://zenodo.org`
[2] `https://figshare.org`

process them is both time consuming and error-prone. As a result, researchers are less inclined to use multiple datasets for their task, and instead develop or test their models on single datasets, reducing the reliability of their results [33]. We believe that the two current biggest blockers of reproducibility in the MIR community are (1) the lack of open datasets, resulting in a lack of transparency as to the consistency of data across publications and (2) the lack of an open library for consistently loading annotations in various formats in common datasets.

In this paper, we introduce an open source library, `mirdata`, which provides tools for using common MIR datasets. We aim to create a useful tool for researchers, which will increase reproducibility, and facilitate and encourage the use of several datasets for evaluation. In particular, it contains tools for loading dataset-specific annotations in a consistent manner, validating if a dataset copy the user has is consistent with a canonical version of the dataset, downloading datasets, and linking audio and annotation files along with track level metadata. We demonstrate the need for this tool by highlighting inconsistencies in common practices when loading annotations and in the data itself (annotations and audio) for three popular MIR datasets, namely iKala, Salami, and the Beatles dataset.

The library is publicly available on Github at `github.com/mir-dataset-loaders/mirdata`.

2. RELATED WORK

Data utility libraries exist for other fields such as text, video and image analysis [1, 10, 27, 28] which allow a user to download a dataset and load it into memory for ease of use in experimentation and consistent results. TensorFlow [1], a deep learning framework, includes a variety of datasets covering images, text, language translation and video [3]. In order to ensure the integrity of the data, TensorFlow hard codes expected file sizes and SHA256 checksums of each file in their library, as well as paths of the data included with the dataset. If the expected values do not match what is downloaded, the local dataset is not considered valid and is not available for usage in the library.

Scikit-Learn [28], a popular machine learning library for Python, includes their own set of dataset loading utilities [4]. Some small datasets, known as "toy datasets", are included directly in the library. Larger datasets, known as "real-world datasets" are downloaded and stored in a "data home" directory on the local machine. Like TensorFlow, Scikit-Learn checks for dataset integrity based on a SHA256 checksum, but only checks the downloaded zip or tar file itself. This approach requires that users download the entirety of the Scikit-Learn library in order to use the dataset loaders.

MLDatasets [5] acts as a specification for how datasets should be managed. DataDeps [36] specifies dataset metadata that conforms to a management system. This method allows for multiple people to maintain their own dataset repositories or for a single organization to set up multiple, independent libraries, each for one dataset.

There are few examples of dataset utility libraries for music. However, the python library Nussl [19] for source separation contains some dataset utilities. Unlike the other libraries previously mentioned, `Nussl` does not include any utilities for dataset retrieval and expects the user to have the datasets locally on the machine before use. However, it includes expected checksums for the dataset audio and logic to check for validity and existence of the dataset as well as simple utilities for loading the data.

Software also exists for loading particular types of (music-centric) annotations, including `pretti_midi` [6] for MIDI data, JAMS [17] for data released in JAMS format, and Music21 [7] for MIDI and MusicXML data. When annotations are released in these formats, custom loading code is less necessary. However, many annotations are released in other formats and require custom loading code.

3. AUDIO FILES IN MIR DATASETS

Datasets in MIR suffer from a unique constraint: most music is protected under copyright. Datasets which are built on copyrighted materials are not typically available for open download. There are several common levels of access for the audio files for different MIR datasets:

1. Open Access
2. Restricted Access (e.g. password protected)
3. "Do it yourself" Access (e.g. YouTube links)
4. No Access

We surveyed 128 MIR datasets from the "Audio Content Analysis" website [8] in April 2019 and determined their access levels. By our estimate, 80 were "open access", 19 were "restricted access", 15 were "DIY" Access, 14 were "no access", meaning that 22.8% of the total list is not openly available. These limited access datasets include historically popular datasets such as RWC [13], AudioSet [12], CAL10k [32], the Beatles dataset [16], iKala [8], the Million Song Dataset [2], and Salami [31].

The more restrictive the access level, the more room there is for "dataset telephone"; when it is difficult to access a particular dataset from a common repository, researchers may share their personal copies with each other, which may contain perturbations from when they first received it. Additionally, since the audio and annotations are sometimes released separately, if the audio is incorrect, the annotations will not correspond to the audio files, resulting in inconsistencies during model development and evaluation. As a result, researchers are performing experiments and computing metrics on datasets they believe are the same as others versions, but may be quite different in reality.

[3] `www.tensorflow.org/datasets/datasets`
[4] `https://scikit-learn.org/stable/datasets`
[5] `https://github.com/JuliaML/MLDatasets.jl`
[6] `github.com/craffel/pretty_midi`
[7] `http://web.mit.edu/music21/doc/index.html`
[8] `https://www.audiocontentanalysis.org/data-sets/`

In an ideal case, the audio files used for a dataset should be the same as those used to create the annotation files. There are a number of popular datasets for which the audio is difficult to obtain. For example, the 7-digital preview clips of the million song dataset [2] have often been used for music classification tasks. While the clips were previously available through an API, it has since been shut down and the clips are no longer available. In the Beatles dataset [16], audio is not released, but instead, catalog numbers and release years of the albums used are provided to prevent differences in audio versions used. Regardless, we found that different versions of the dataset have been used by researchers (see Section 4.1). In the case of AudioSet [12], audio is provided in the form of YouTube video identifiers, adding a new challenge in data reproducibility. However, the availability of the linked YouTube videos changes over time, and accessibility varies by country.

4. EXPERIMENTS - WHY A COMMON TOOL IS NEEDED

Differences in audio or annotations, or in the code used to load data into memory can have a huge impact on downstream results. In this section, we examine the effect of real differences we found on evaluation metrics in instances of three popular MIR datasets.

4.1 The Beatles Dataset

The Beatles dataset [16] contains annotations for beats, downbeats, sections, and chords for the nearly entire Beatles' collection. However, as the audio is copyrighted, only the annotations are released as part of the dataset. The researchers are asked to use their personal copy of the Beatles' catalog and match the audio files with the annotations.

The annotations were created using a particular version of the audio, and they may not correspond well with other versions. For this dataset in particular, it is quite easy to end up with different versions of the same Beatles track, since there are several releases of every album, including remastered versions.

To evaluate these potential differences, we first compared checksums across four different researcher's copies of the audio files corresponding to the Beatles dataset. Out of the four versions, three had identical checksums, while one had invalid checksums on every single audio file, indicating that the audio is completely different between the two versions. Upon further examination of the differences, we found inconsistencies in the number of channels, the duration, and the average RMS of the audio files between the two versions.

The differences go even further than channels, duration and volume. In Figure 1, we first normalize a pair of audio files to have the same peak level and compute the absolute difference in their spectrograms. In the low frequencies, in particular, there are major differences between the frequency content of the two versions, despite sounding similar.

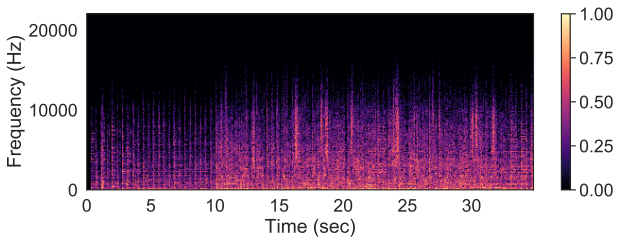


Figure 1. Normalized absolute difference between two spectrograms of the first 30 seconds of "Across the Universe" computed on audio files from two versions of the Beatles dataset audio.

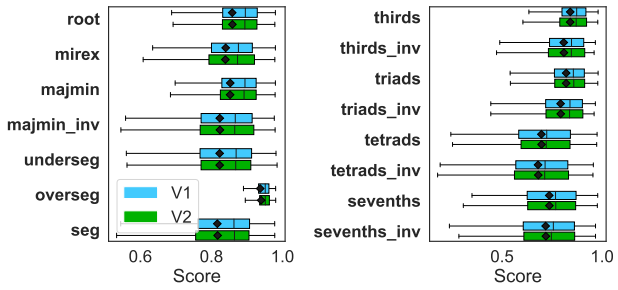


Figure 2. Chord metrics for chord estimates computed on two different versions of the Beatles audio, compared with the Beatles dataset reference chord annotations.

Next, we ran a chord recognition algorithm [21] on the two different versions of the audio collection, and computed the standard chord recognition metrics as implemented in `mir_eval` using the dataset's (public) reference chord annotations. The differences in the metrics are shown in Figure 2. While only one of these metrics had statistically significantly different results ("overseg", according to a paired t-test), we see that the same chord recognition algorithm produces results which are different enough to affect the metrics.

4.2 The iKala Dataset

The iKala dataset [8] is commonly used for melody estimation, vocal activity detection, and source separation. It contains isolated vocals and instruments (provided as left and right channels of a stereo audio file), along with vocal f_0 annotations and lyrics.

We performed the same checksum experiment as for the Beatles dataset and compared checksums for four different researcher's copies of the iKala dataset. We found that all four versions (audio and annotations) were identical.

One challenge with the iKala dataset is that the vocal f_0 annotations are provided as newline-separated files with the pitch, but without timestamps, which must be inferred upon load. On the dataset's website, they state that the hop size is 0.032 seconds, but it does not state the alignment of the first time frame (left aligned or center aligned). The dataset's website also provides code for loading the annotation files, which uses a different hop size of 0.03125 seconds and center aligned frames (with the first time stamp

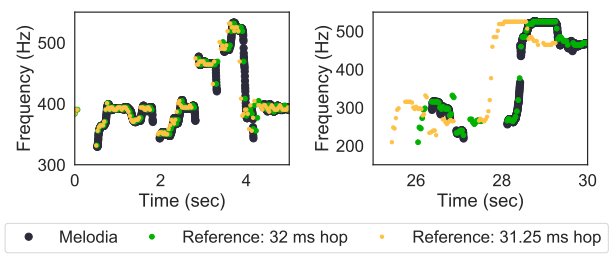


Figure 3. iKala reference annotations loaded using two different hop sizes (32 ms and 31.25 ms) versus the output of Melodia. (Left) the first 5 seconds of the track. (Right) the last 5 seconds of the track.

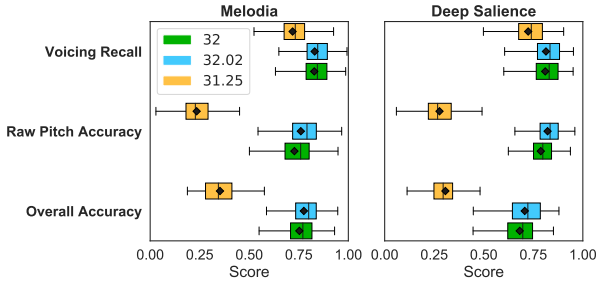


Figure 4. Melodia and Deep Salience melody metrics when evaluated against iKala's reference data loaded with 3 different hop sizes.

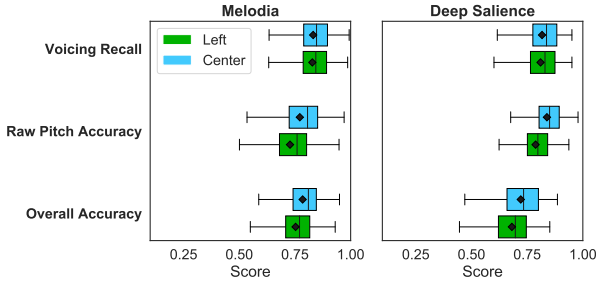


Figure 5. Melodia and Deep Salience melody metrics when evaluated against iKala's reference data loaded with left and center-aligned time stamps.

at 0.5 * hop seconds).

To see how users infer the iKala time stamps, we performed a search of public code on Github.com for code which loads the iKala pitch annotations. We found 5 unique ways of loading the time stamps, consisting of 3 different hop sizes (0.032 s and 0.03125 as listed on the dataset website, and 0.032017, inferred from the duration of the audio files), and two different alignments (left and center). By far, the most common combination was using a hop size of 0.032 s and left aligned frames.

The differences in hop size have a major effect on the alignment of the audio and the annotation, especially over time. Figure 3 shows an example of the annotations loaded with two of the hop sizes, and the estimate of a melody extraction algorithm (Melodia [30]) for comparison. In the first 5 seconds, the differences are small, but in the last 5 seconds, we see a visible misalignment between the loaded annotations and the audio.

To investigate the severity of these differences, we ran two melody extraction algorithms, Melodia [30] and Deep Salience [3], on the iKala audio. We then compute melody evaluation metrics using `mir_eval` with reference times computed using the three different hop sizes, $h = 32$ ms, $h = 32.017$ ms and $h = 31.25$ ms, using *left* aligned frames. In Figure 4, we show boxplots of the results across tracks in the dataset for each of these reference hop sizes. The results for different hop sizes are quite different, and drastically so for the smallest hop size, 0.03125 s. Even the difference between $h = 32$ and $h = 32.017$ in Overall Accuracy is substantial for both datasets - a difference that is historically enough to claim state of the art over another algorithm. A paired T-test shows statistically significant differences for all pairs of hop sizes for each metric, with the exception of Voicing Recall for $h = 32$ and $h = 32.017$

Next, we compute the melody metrics for the same melody extraction algorithms using a hop size of $h = 32$ ms and compare left vs center aligned frames in the reference. Figure 5 shows the results per track of the two different reference alignments. The difference in metrics is smaller than for the hop size differences, but left alignment is statistically significantly worse than right alignment for Overall Accuracy and Raw Pitch Accuracy under a paired T-test.

This begs the question: which is the correct way to load the timestamps? Since it is quite unlikely that an incorrectly time aligned reference would produce higher scores than a correct alignment, it is likely that, despite the norm of using left-aligned frames, the annotations are intended to have center aligned timestamps. Indeed, if we look at a specific example of left vs. center aligned timestamps for a short excerpt compared with two different algorithm estimates, as in Figure 6, we see that the reference is better aligned with both estimates when using center aligned time frames. Note that in Figure 4, the reference hop of 32.02 ms resulted in higher metrics than the hop of 32 ms (both with left aligned frames), and a 32 ms hop with center aligned frames has higher metrics than all of the left-aligned hops. The data loaded with a hop size of 32.02 ms starts off misaligned but over time, approaches a center alignment, explaining the "better" score with this incorrect hop size.

The shocking result of this set of experiments is that every single example we found publicly – including the code found on the dataset's website – appears to be loading the f_0 data incorrectly, either with an incorrect hop size or an incorrect alignment - no example we found had both a hop size of 32 ms and center alignment.

4.3 Salami Dataset

The Salami dataset [31] is a popular dataset used for music structural segmentation. It consists of 1359 tracks across

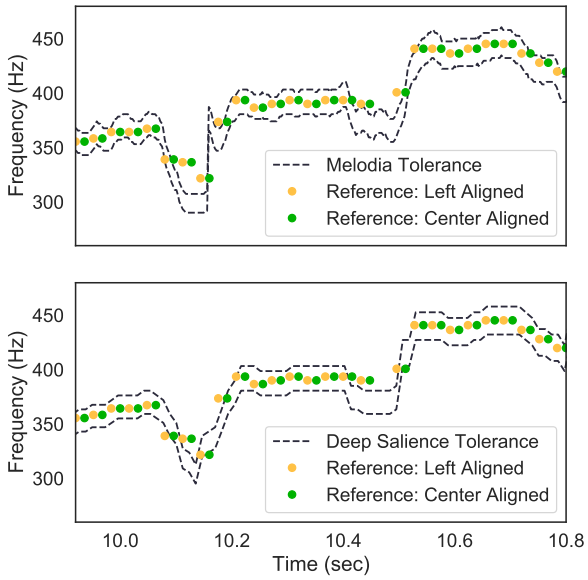


Figure 6. Left and center aligned time stamps for a track in iKala versus algorithm estimates from Melodia and Deep Salience. The dashed lines show the distance from the estimate where the algorithm would be considered correct in the standard melody extraction metrics.

a wide variety of genres, namely classical, jazz, popular, world, among others. Each track has annotations of 'coarse' and 'fine' segments, and among the annotated files, a subset of 884 tracks was annotated by two distinct annotators. The complete set of annotations was released in version 2.0 of the dataset, increasing the volume of the dataset with respect to previous versions. For instance, from version 1.9 to 2.0 additional annotations related to 539 tracks were added, 390 with multiple annotations and 189 with single annotations. Both versions are available in the dataset's main repository. However, as in other cases when a dataset is updated, there is no centralized version control that is transparent and ensures the awareness of the community to these changes.

Data-driven models are increasingly popular for addressing MIR tasks, including the task of boundary detection using the Salami dataset [15, 35]. One of the main reproducibility-related issues about data-driven models is their training, in particular, the amount of annotated data is crucial. When using Salami, it is important to avoid using an old version of the annotations, which could have a negative impact in a model's performance in comparison with the same model trained on the newest version of the dataset. In particular, if different data is used for training two different models with the aim of comparing their performance afterward, it is difficult to isolate possible causes of performance differences. An example of this situation is shown in [11], where the authors use a different subset of Salami than previous works, obtaining substantially different results when intending to re-implement other authors' model (e.g. 0.246 instead of the previously reported 0.523 F-measure with a ± 0.5 s tolerance window).

Another possible source of inconsistency with the use of this dataset relates to contributions from people other than the dataset creators. The authors in [24] manually edited 171 of the annotations in version 2.0, to correct formatting errors and enforce consistency with the annotation guide proposed by the dataset creators. However, this "corrected" version of the annotations was not included in the dataset's main repository. This third version of annotations is used in recent works [24, 34]; comparison of systems without awareness of the difference with the version 2.0 released annotations may lead to differences in performance that are beyond models' design.

5. MIR DATASET LOADERS

In this section, we describe the `mirdata` python library, our proposed solution to the current reproducibility issues with dataset versions and loaders. A driving philosophy of this library is to work with the imperfect situation we are faced with, with regards to the limited openness of MIR data. In an ideal scenario, all data would be freely sharable and version controlled; since this is not the case, we do our best to create tools to maximize reproducibility given the current constraints. Most importantly, we aimed to create a clean, transparent and easy to use interface to encourage reuse and contributions. The first release of the library will include loaders for Orchset [7], iKala [8], MedleyDB Melody and Pitch subsets [4], the Beatles dataset [16], Salami [31], the Million Song Dataset [2], Medley Solos-DB [18], RWC [9, 13, 14, 20], DALI [26], and GuitarSet [37].

5.1 Dataset Indexes and Checksums

Datasets, by their definition, are collections of data. In the case of MIR datasets, the data is often a collection of separate files, some of which correspond to e.g. a particular audio file. For example, the Beatles dataset contains four separate text files containing chord, beat, section, and key annotations for each audio file. Since each of these four annotation files are related to the same audio file, it is desirable to have a common way of linking them. In `mirdata`, we use a *dataset index* to link related files, in the form of a JSON file. This index contains a unique identifier for each group (for example, the name of the audio file), which is mapped to its corresponding file paths and their expected *checksums*, for example:

```
{
    "track1": {
        "audio": [
            "example/audio/track1.wav",
            "912ec803b2ce49e4a541068d495ab570"
        ],
        "annotation": [
            "example/annotations/track1.csv",
            "2cf33591c3b28b382668952e236cccd5"
        ]
    },
    ...
}
```

The use of an index for each dataset is advantageous for a number of reasons. First, it groups related data files in

a transparent way, avoiding audio-annotation pairing mistakes, and removing the need for custom filename substring matching per dataset. Second, it gives a version-controlled record of all expected files in the dataset, preventing inconsistencies due to missing or extra files; we can check if all the expected files are present in a local copy of a dataset, and we load data to memory based on the index, ignoring files not included in it. Finally, it provides a way to verify if a local copy of a dataset is consistent with a canonical version on a file-by-file basis.

For each dataset, we also provide a `validate()` function, which checks for the existence of files locally and compares the expected checksums with checksums of the local copy. A checksum is a representation of a digital file similar to a fingerprint and usually computed by taking a hash of the bits in a file. The smaller (in size) representation created allows for efficient file comparisons at the bit level. If two files have the same checksum then a user can assume that the two files are exactly the same with high confidence. If the checksums between two files differ or the computed checksum is different from an expected checksum, then the user is at a minimum aware of discrepancies and can take appropriate action. In the `mirdata` use case, checksums allow users who have a local copy of a dataset to know whether or not they are using the same data as others, simply by running the validation function.

Note that there is not always one "correct" version of a dataset, so it is difficult to decide which version of a dataset should be used to create the reference checksums. For this library, we compute checksums on the version that is as close to the "original" as possible, for example by obtaining a version from dataset creators.

5.2 Dataset Downloading

It can often be difficult or unclear how to get access to a particular dataset, and same data often exists in multiple places and may not be identical. In `mirdata`, we provide a `download()` function for each dataset. When data is openly available online, the function automatically downloads the data, and this version of the data is same the version used to create the checksums. When the data has been downloaded, we run validation to ensure it matches and warn the user if not (e.g. in the case of an incomplete download). When data is not openly available online, we provide instructions for how to obtain the data (e.g. by requesting access on a particular website). Once the data has been obtained, the user can then run validation to ensure the data is consistent.

5.3 Annotation Loaders

As highlighted previously, differences in implementations for loading annotation data to memory can have big effects on the resulting data. In `mirdata`, we remove the need for users to manually write loaders per dataset and annotation type by providing functions for loading all annotations for each dataset. These implementations are shared and transparent, allowing users to permanently correct mistakes in the way data is loaded.

As an example, for some of the beat annotations in the Salami dataset, the beat position is missing for only a few observations. These missing positions can be inferred from the neighboring information (e.g. beats 1 and 3 have labels, and the one in between is absent), and in `mirdata`'s implementation we fill in this information on load.

5.4 Track Metadata

More often than not, in addition to data files containing e.g. time varying annotations, datasets provide track-level metadata. When loading annotations, we also link any available track level metadata with each track id group. This can be particularly useful when splitting data, such as for creating unbiased train-validation-test splits [23], or for analyzing evaluation metrics over different splits of a dataset.

6. CONCLUSIONS

Although data distribution challenges remain, we believe that the use of `mirdata` will result in reproducible usage of datasets in research moving forward. Future iterations of `mirdata` will include support for large (out of memory) datasets and an increased number of supported datasets. As datasets become more open and annotation formats standardize, the scientific need for this library will lessen, but it will remain a useful tool for ease of working with datasets.

Importantly, we designed `mirdata` to have a low barrier to entry for contributions. New datasets can be easily included independently with minimal interfacing with the rest of the library. With active community participation, we believe that `mirdata` can help ensure that MIR datasets are used in a consistent, reproducible manner moving forward.

7. REFERENCES

[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida*, pages 591–596, 2011.

[3] Rachel M Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan P. Bello. Deep salience representations

for f0 estimation in polyphonic music. In *18th International Society of Music Information Retrieval (ISMIR) Conference*, October 2017.

[4] Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan P Bello. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *International Society of Music Information Retrieval (ISMIR)*, October 2014.

[5] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: a new python audio and music signal processing library. In *Proceedings of the 24th ACM International Conference on Multimedia*, ACMMM, 2016.

[6] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez Gutiérrez, Sankalp Gulati, Perfecto Herrera Boyer, Oscar Mayor, Gerard Roma Trepat, Justin Salamon, José Ricardo Zapata González, and Xavier Serra. Essentia: An audio analysis library for music information retrieval. In *14th Conference of the International Society for Music Information Retrieval (ISMIR)*. International Society for Music Information Retrieval (ISMIR), 2013.

[7] Juan J Bosch, Ricard Marxer, and Emilia G'omez. Evaluation and combination of pitch estimation methods for melody extraction in symphonic classical music. *Journal of New Music Research*, 45(2):101–117, 2016.

[8] Tak-Shing Chan, Tzu-Chun Yeh, Zhe-Cheng Fan, Hung-Wei Chen, Li Su, Yi-Hsuan Yang, and Roger Jang. Vocal activity informed singing voice separation with the ikala dataset. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 718–722. IEEE, 2015.

[9] Taemin Cho and Juan P Bello. A feature smoothing method for chord recognition using recurrence plots. In *12th International Society for Music Information Retrieval Conference*, ISMIR, 2011.

[10] François Chollet et al. Keras. `https://keras.io`, 2015.

[11] Alice Cohen-Hadria and Geoffroy Peeters. Music structure boundaries estimation using multiple self-similarity matrices as input depth of convolutional neural networks. In *Audio Engineering Society Conference: 2017 AES International Conference on Semantic Audio*. Audio Engineering Society, 2017.

[12] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2017.

[13] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. Rwc music database: Popular, classical and jazz music databases. In *ISMIR*, volume 2, pages 287–288, 2002.

[14] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. Rwc music database: Music genre database and musical instrument sound database. 2003.

[15] Thomas Grill and Jan Schlüter. Music boundary detection using neural networks on combined features and two-level annotations. In *ISMIR*, pages 531–537, 2015.

[16] Christopher Harte. *Towards automatic extraction of harmony information from music signals*. PhD thesis, 2010.

[17] Eric J Humphrey, Justin Salamon, Oriol Nieto, Jon Forsyth, Rachel M Bittner, and Juan Pablo Bello. Jams: A json annotated music specification for reproducible mir research. In *ISMIR*, pages 591–596, 2014.

[18] Vincent Lostanlen, Carmine-Emanuele Cella, Rachel Bittner, and Slim Essid. Medley-solos-DB: a cross-collection dataset for musical instrument recognition, September 2018.

[19] Ethan Manilow, Prem Seetharaman, and Bryan Pardo. The northwestern university source separation library. Proceedings of the 19th International Society of Music Information Retrieval Conference (ISMIR 2018), Paris, France, September 23-27, 2018.

[20] Matthias Mauch, Hiromasa Fujihara, Kazuyoshi Yoshii, and Masataka Goto. Timbre and melody features for the recognition of vocal activity and instrumental solos in polyphonic music. In *ISMIR*, ISMIR, 2011.

[21] Brian McFee and Juan P. Bello. Structured training for large-vocabulary chord recognition. In *18th International Society for Music Information Retrieval Conference*, ISMIR, 2017.

[22] Brian McFee, Eric J Humphrey, Oriol Nieto, Justin Salamon, Rachel Bittner, Jon Forsyth, and Juan P Bello. Pump up the jams: V0. 2 and beyond. *Music and Audio Research Laboratory, New York University, Tech. Rep*, 2015.

[23] Brian McFee, Jong Wook Kim, Mark Cartwright, Justin Salamon, Rachel M Bittner, and Juan Pablo Bello. Open-source practices for music signal processing research: Recommendations for transparent, sustainable, and reproducible audio research. *IEEE Signal Processing Magazine*, 36(1):128–137, 2019.

[24] Brian McFee, Oriol Nieto, Morwaread M Farbood, and Juan Pablo Bello. Evaluating hierarchical structure in music annotations. *Frontiers in psychology*, 8:1337, 2017.

[25] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25, 2015.

[26] Gabriel Meseguer-Brocal, Alice Cohen-Hadria, and Geoffroy Peeters. Dali: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm. In *19th International Society for Music Information Retrieval Conference*, 2018.

[27] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[28] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[29] Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel PW Ellis, and C Colin Raffel. mir_eval: A transparent implementation of common mir metrics. In *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. Citeseer, 2014.

[30] Justin Salamon and Emilia Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio. Speech. Lang. Processing*, 20(6):1759–1770, 2012.

[31] Jordan Bennett Louis Smith, John Ashley Burgoyne, Ichiro Fujinaga, David De Roure, and J Stephen Downie. Design and creation of a large-scale database of structural annotations. In *ISMIR*, volume 11, pages 555–560. Miami, FL, 2011.

[32] Derek Tingle, Youngmoo E Kim, and Douglas Turnbull. Exploring automatic music annotation with acoustically-objective tags. In *Proceedings of the international conference on Multimedia information retrieval*, pages 55–62. ACM, 2010.

[33] Antonio Torralba, Alexei A Efros, et al. Unbiased look at dataset bias. In *CVPR*, volume 1, page 7. Citeseer, 2011.

[34] Christopher J Tralie and Brian McFee. Enhanced hierarchical music structure annotations via feature level similarity fusion. *arXiv preprint arXiv:1902.01023*, 2019.

[35] Karen Ullrich, Jan Schlüter, and Thomas Grill. Boundary detection in music structure analysis using convolutional neural networks. In *ISMIR*, pages 417–422, 2014.

[36] Lyndon White, Roberto Togneri, Wei Liu, and Mohammed Bennamoun. Datadeps.jl: Repeatable data setup for replicable data science. *CoRR*, abs/1808.01091, 2018.

[37] Qingyang Xi, Rachel M Bittner, Johan Pauwels, Xuzhou Ye, and Juan Pablo Bello. Guitarset: A dataset for guitar transcription. In *ISMIR*, pages 453–460, 2018.

COVER DETECTION USING DOMINANT MELODY EMBEDDINGS

Guillaume Doras
Sacem & Ircam Lab, CNRS, Sorbonne Université
guillaume.doras@sacem.fr

Geoffroy Peeters
LTCI, Telecom Paris, Institut Polytechnique de Paris
geoffroy.peeters@telecom-paris.fr

ABSTRACT

Automatic cover detection – the task of finding in an audio database all the covers of one or several query tracks – has long been seen as a challenging theoretical problem in the MIR community and as an acute practical problem for authors and composers societies. Original algorithms proposed for this task have proven their accuracy on small datasets, but are unable to scale up to modern real-life audio corpora. On the other hand, faster approaches designed to process thousands of pairwise comparisons resulted in lower accuracy, making them unsuitable for practical use.

In this work, we propose a neural network architecture that is trained to represent each track as a single embedding vector. The computation burden is therefore left to the embedding extraction – that can be conducted offline and stored, while the pairwise comparison task reduces to a simple Euclidean distance computation. We further propose to extract each track's embedding out of its dominant melody representation, obtained by another neural network trained for this task. We then show that this architecture improves state-of-the-art accuracy both on small and large datasets, and is able to scale to query databases of thousands of tracks in a few seconds.

1. INTRODUCTION

Covers are different interpretations of the same original musical work. They usually share a similar melodic line, but typically differ greatly in one or several other dimensions, such as their structure, tempo, key, instrumentation, genre, etc. Automatic cover detection – the task of finding in an audio database all the covers of one or several query tracks – has long been seen as a challenging theoretical problem in MIR. It is also now an acute practical problem for copyright owners facing continuous expansion of user-generated online content.

Cover detection is not *stricto sensu* a classification problem: due to the ever growing amount of musical works (the classes) and the relatively small number of covers per work, the actual question is not so much "to which work this track belongs to ?" as "to which other tracks this track is the most similar ?".

Formally, cover detection therefore requires to establish a similarity relationship S_{ij} between a query track A_i and a reference track B_j. It implies the composite of a feature extraction function f followed by a pairwise comparison function g, expressed as $S_{ij} = g(f(A_i), f(B_j))$. If f and g are independent, the feature extraction of the reference tracks B_j can be done offline and stored. The online feature extraction cost is then linear in the number of queries, while pairwise comparisons cost without optimisation scales quadratically in the number of tracks [16].

Efficient cover detection algorithms thus require a fast pairwise comparison function g. Comparing pairs of entire sequences, as DTW does, scales quadratically in the length of the sequences and becomes quickly prohibitive. At the opposite, reducing g to a simple Euclidean distance computation between tracks embeddings is independent of the length of the sequences. In this case, the accuracy of the detection entirely relies on the ability of f to extract the common musical facets between different covers.

In this work, we describe a neural network architecture mapping each track to a single embedding vector, and trained to minimize cover pairs Euclidean distance in the embeddings space, while maximizing it for non-cover pairs. We leverage on recent breakthroughs in dominant melody extraction, and show that the use of dominant melody embeddings yield promising performances both in term of accuracy and scalability.

The rest of the paper is organized as follow: we review in §2 the main concepts used in this work. We detail our method in §3, and describe and discuss in §4 and §5 the different experiments conducted and their results. We finally present a comparison with existing methods in §6. We conclude with future improvements to bring to our method.

2. RELATED WORK

We review here the main concepts used in this study.

2.1 Cover detection

Successful approaches in cover detection used an input representation preserving common musical facets between different versions, in particular dominant melody [19, 27, 40], tonal progression – typically a sequence of chromas [10, 12, 33, 39] or chords [2], or a fusion of both [11, 29]. Most of these approaches then computed a similarity score between pairs of melodic and/or harmonic sequences, typically a cross-correlation [10], a variant of the DTW algorithm [12, 20, 33, 39], or a combination of both [25].

These approaches lead to good results when evaluated on small datasets – at most a few hundreds of tracks, but are not scalable beyond due to their expensive comparison function. Faster methods have recently been proposed, based on efficient comparison of all possible subsequences pairs between chroma representations [34], or similarity search between 2D-DFT sequences derived from CQTs overlapping windows [31], but remain too costly to be scalable to query large modern audio databases.

Another type of method has been proposed to alleviate the cost of the comparison function and to shift the burden to the audio features extraction function – which can be done offline and stored. The general principle is to encode each audio track as a single scalar or vector – its embedding – and to reduce the similarity computation to a simple Euclidean distance between embeddings. Originally, embeddings were for instance computed as a single hash encoding a succession of pitch landmarks [3], or as a vector obtained by PCA dimensionality reduction of a chromagram's 2D-DFT [4] or with locality-sensitive hashing of melodic excerpts [19].

As for many other MIR applications, ad-hoc – and somewhat arbitrary – hand-crafted features extraction was progressively replaced with data-driven automatic feature learning [15]. Different attempts to learn common features between covers have since been proposed: in particular, training a k-means algorithm to learn to extract an embedding out of chromagram's 2D-DFT lead to significant results improvements on large datasets [16]. Similar approaches, commonly referred to as *metric learning* approaches, have been used in different MIR contexts, such as music recommendation [21, 41], live song identification [38], music similarity search [24], and recently cover detection [23].

2.2 Metric learning

Although the concept can be traced back to earlier works [1, 8], the term of metric learning was probably coined first in [43] to address this type of clustering tasks where the objective is merely to assess whether different samples are similar or dissimilar. It has since been extensively used in the image recognition field in particular [14, 36, 37].

The principle is to learn a mapping between the input space and a latent manifold where a simple distance measure (such as Euclidean distance) should approximate the neighborhood relationships in the input space. There is however a trivial solution to the problem, where the function ends up mapping all the examples to the same point. Contrastive Loss was introduced to circumvent this problem, aiming at simultaneously *pulling* similar pairs together and *pushing* dissimilar pairs apart [13].

However, when the amount of labels becomes larger, the number of dissimilar pairs becomes quickly intractable. It was moreover observed in practice that once the network has become reasonably good, negative pairs become relatively easy to discern, which stalls the training of the discriminative model. *Pair mining* is the strategy of training the model only with hard pairs, i.e. positive (resp. negative) pairs with large (resp. small) distances [35]. Further improvement was introduced with the triplet loss, which is used to train a model to map each sample to an embedding that is closer to all of its positive counterparts than it is to all of its negative counterparts [30]. Formally, for all triplets $\{a,\ p,\ n\}$ where a is an anchor, and p or n is one of its positive or negative example, respectively, the loss to minimize is expressed as $\ell = \max(0, d_{\mathrm{ap}} + \alpha - d_{\mathrm{an}})$, where α is a margin and d_{ap} and d_{an} are the distances between each anchor a and p or n, respectively.

2.3 Dominant melody extraction

Dominant melody extraction has long been another challenging problem in the MIR community [18, 28, 42]. A major breakthrough was brought recently with the introduction of a convolutional network that learns to extract the dominant melody out of the audio Harmonic CQT [7]. The HCQT is an elegant and astute representation of the audio signal in 3 dimensions (time, frequency, harmonic), stacking along the third dimension several standard CQTs computed at different minimal multiple frequencies. Harmonic components of audio signal will thus be represented along the third dimension and be localized at the same location along the first and second dimensions. This representation is particularly suitable for melody detection, as it can be directly processed by convolutional networks, whose 3-D filters can be trained to localize in the time and frequency plan the harmonic components.

In a recent work [9], we suggested in an analogy with image processing that dominant melody extraction can be seen as a type of image segmentation, where contours of the melody have to be isolated from the surrounding background. We have thus proposed for dominant melody estimation an adaptation of U-Net [26] – a model originally designed for medical image segmentation – which slightly improves over [7].

3. PROPOSED METHOD

We present here the input data used to train our network, the network architecture itself and its training loss.

3.1 Input data

We have used as input data the dominant melody 2D representation (F0-CQT) obtained by the network we proposed in [9]. The frequency and time resolutions required for melody extraction (60 bins per octave and 11 ms per time frame) are not needed for cover detection. Moreover, efficient triplet loss training requires large training batches, as we will see later, so we reduced data dimensionality as depicted on Figure 2.

The F0-CQT is **a)** trimmed to keep only 3 octaves around its mean pitch (180 bins along the frequency axis), and only the first 3 minutes of the track (15500 time frames) – if shorter, the duration is not changed. The resulting matrix is then **b)** downsampled via bilinear 2D interpolation with a factor 5. On the frequency axis, the semi-tone resolution is thus reduced from five to one bin,

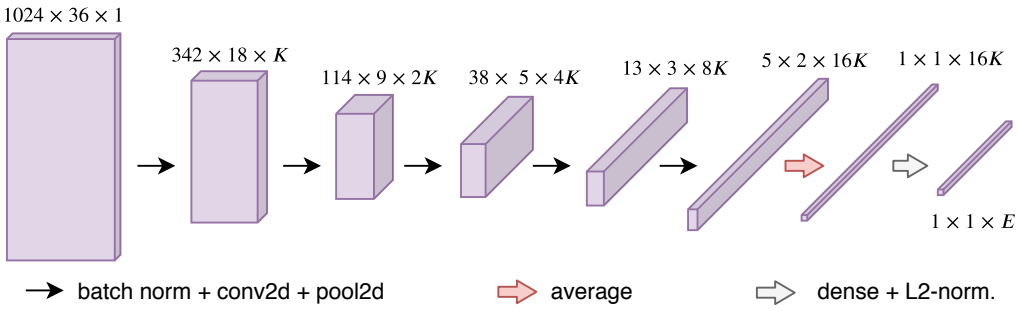


Figure 1: Convolutional model (time on the first dimension, frequency on the second dimension).

which we considered adequate for cover detection. On the time axis, it is equivalent to a regular downsampling.

Finally, as the representation of different tracks with possibly different durations shall be batched together during training, the downsampled F0-CQT is **c)** shrunk or stretched along the time axis by another bilinear interpolation to a fixed amount of bins (1024). This operation is equivalent to a tempo change: for the 3 minutes trimmed, shrinking is equivalent to multiply the tempo by a factor 3. We argue here that accelerated or decelerated version of a cover is still a cover of the original track.

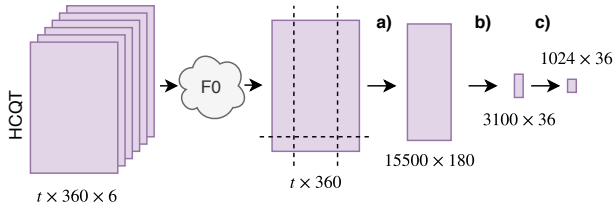


Figure 2: Input data pre-processing: dominant melody is extracted from HCQT, then a) F0 output is trimmed, b) downsampled by a factor 5 and c) resized time-wise to 1024 bins (time bins on first dimension, frequency bins on second dimension).

3.2 Model

The proposed model is a simple convolutional network pictured in Figure 1. As we are constrained by the input data shape, whose time dimension is much larger than its frequency dimension, only five layers blocks are needed. Each layer block consists of a batch normalization layer, a convolution layer with 3×3 kernels and a mean-pooling layer with a 3×2 kernel and 3×2 stride in order to reduce time dimensionality faster than frequency dimensionality. A dropout rate of $0.1, 0.1, 0.2$ and 0.3 is applied to the blocks 2, 3, 4 and 5, respectively.

The first convolutional layer has K kernels, and this number is doubled at each level (i.e. the deeper layer outputs 2^4K-depth tensors). The penultimate layer averages along frequency and time axes to obtain a vector. A last dense layer outputs and L2-normalizes the final embedding vector of size E.

Our assumption behind the choice of this convolutional architecture is that we expect it to learn similar patterns in the dominant melody, at different scales (tempo invariance) and locations (key and structure invariance).

3.3 Objective loss

We use a triplet loss with online semi-hard negative pairs mining as in [30]. In practice, triplet mining is done within each training batch: instead of using all possible triplets, each track in the batch is successively considered as the anchor, and compared with all its covers in the batch. For each of these positives pairs, if there are negatives such as $d_{\text{an}} < d_{\text{ap}}$, then only the one with the highest d_{an} is kept. If no such negative exist, then only the one with the lowest d_{an} is kept. Other negatives are not considered.

Model is fit with Adam optimizer [17], with initial learning rate at $1e^{-4}$, divided by 2 each time the loss on the evaluation set does not decrease after 5k training steps. Training is stopped after 100k steps, or if the learning rate falls below $1e^{-7}$. The triplet loss was computed using squared Euclidean distances (i.e. distances are within the $[0, 4]$ range), and the margin was set to $\alpha = 1$.

3.4 Dataset

As metric learning typically requires large amount of data, we fetched from internet the audio of cover tracks provided by the SecondHandSongs website API [1]. Only works with 5 to 15 covers, and only tracks lasting between 60 and 300 seconds where considered, for a total of $W = 7460$ works and $T = 62310$ tracks.

The HCQT was computed for those 62310 tracks as detailed in [7], i.e. with f_{min} = 32.7 Hz and 6 harmonics. Each CQT spans 6 octaves with a resolution of 5 bins per semi-tone, and a frame duration of ~11 ms. The implementation was done with the Librosa library [22].

The dominant melody was extracted for these 62310 HCQT with the network we described in [9], and the output was trimmed, downsampled and resized as described in §3.1.

4. PRELIMINARY EXPERIMENTS

We present here some experiments conducted to develop the system. The 7460 works were split into disjoint train and evaluation sets, with respectively 6216 and 1244 works and five covers per work. The evaluation set represents ~20% of the training set, which we considered fair enough given the total amount of covers. The same split has been used for all preliminary experiments.

[1] https://secondhandsongs.com/

4.1 Metrics

Ideally, we expect the model to produce embeddings such that cover pair distances are low and non-cover pair distances are high, with a large gap between the two distributions. In the preliminary experiments, we have thus evaluated the separation of the cover pairs distance distribution $p_c(d)$ from the non-cover pairs distance distribution $p_{nc}(d)$ with two metrics:

- the ROC curve plots the true positives rate (covers, TPR) versus the false positive rate (non-covers, FPR) for different distance d thresholds. We report the area under the ROC curve (AuC), which gives a good indication about the distributions separation. We also report the TPR corresponding to an FPR of 5% (TPR@5%), as it gives an operational indication about the model's discriminative power.

- we also report the Bhattacharyya coefficient (BC), expressed as $\sum_d \sqrt{p_c(d)p_{nc}(d)}$, as it directly measures the separation between the distributions (smaller is better) [6].

4.2 Influence of input data

We first compared the results obtained for different inputs data: chromas and CQT computed using Librosa [22], and the dominant melody computed as described in 3.1. As shown on Figure 3 (left), dominant melody yields the best results. It does not imply that melody features are more suited than tonal features for cover detection, but shows that convolutional kernels are better at learning similar patterns at different scales and locations across different tracks when the input data is sparse, which is not the case for chromas and CQT.

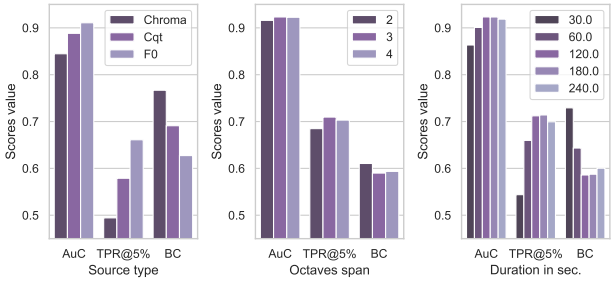


Figure 3: Scores obtained on evaluation set for a model trained - with chromas, CQT or F0 (left). - on the F0 with various octaves spans (middle). - on the F0 with various durations (right).

Results obtained when trimming the F0-CQT with various octaves and time spans are also shown Figure 3. It appears that keeping 3 octaves around the mean pitch of the dominant melody and a duration of 2 to 3 minutes yields the best results. Smaller spans do not include enough information, while larger spans generate confusion.

All other results presented below are thus obtained with the dominant melody 2D representation as input data, and a span of 3 octaves and 180 seconds for each track.

4.3 Influence of model and training parameters

We then compared the results obtained for different numbers of kernels in the first layer (K) and the corresponding sizes of the embeddings (E). As shown on Figure 4 (left), results improve for greater K, which was expected. However, increasing K above a certain point does not improve the results further, as the model has probably already enough freedom to encode common musical facets.

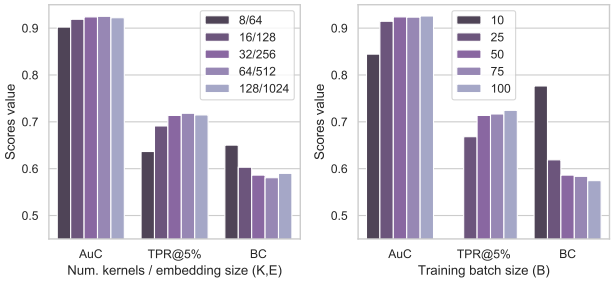


Figure 4: Scores obtained on evaluation set for a model trained - with various K/E (left) - with various B (right).

We have then compared the results obtained for different sizes of training batches (B). As shown on Figure 4 (right), results improve with larger B: within larger batches, each track will be compared with a greater number of non-covers, improving the separation between clusters of works. A closer look at the distances shows indeed that the negative pairs distance distribution $p_{nc}(d)$ gets narrower for larger batches (not showed here). Due to GPU memory constraints, we have not investigated values above B=100.

All other results presented below are obtained with K=64, E=512 and B=100.

5. LARGE SCALE LOOKUP EXPERIMENTS

We now present experiments investigating the realistic use case, i.e. large audio collections lookup. When querying an audio collection, each query track can be of three kinds: **a)** it is already present in the database, **b)** it is a cover of some other track(s) already in the database, or **c)** it is a track that has no cover in the database. The case **a)** corresponds to the trivial case, where the query will produce a distance equal to zero when compared with itself, while case **c)** corresponds to the hard case where neither the query or any cover of the query have been seen during training. We investigate here the case **b)**, where the query track itself has never been seen during training, but of which at least one cover has been seen during training.

5.1 Metrics

In these experiments, we are interested in measuring our method's ability to find covers in the reference set when queried with various unknown tracks. This is commonly addressed with the metrics proposed by MIREX [2] for the cover song identification task: the mean rank of first correct result (MR1), the mean number of true positives in the top ten positions (MT10) and the Mean Average Precision (MAP). We refer the reader to [32] for a detailed review of these standard metrics. We also report here the TPR@5%, already used in the premilinary experiments.

[2] https://www.music-ir.org/mirex/wiki/2019

5.2 Structuring the embeddings space

We study here the role of the training set in structuring the embeddings space, and in particular the role of the number of covers of each work. More precisely, we tried to show evidence of the *pushing* effect (when a query is pushed away from all its non-covers clusters) and the *pulling* effect (when a query is pulled towards its unique covers cluster).

To this aim, we built out of our dataset a query and a reference set. The query set includes 1244 works with five covers each. The reference set includes P of the remaining covers for each of the 1244 query works, and N covers for each other work not included in the query set (Figure 5).

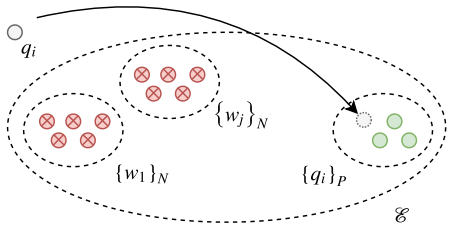


Figure 5: Big dotted oval represents the embeddings space $\mathcal{E}$. Smaller dotted ovals represent work clusters of covers. Red crossed circles represent the positions in the manifold of the reference tracks w_j that *are not* covers of query track q_i (N per cluster). Green circles represent the positions of the tracks that *are* covers of query track q_i (P per cluster).

Pushing covers We first train our model on the reference set with fixed P=5. We compute query tracks embeddings with the trained model, compute pairwise distances between query and reference embeddings, as well as the different metrics. We repeat this operation for different values of $N \in [2, ..., 10]$, and report results on Figure 6 (left). We report MR1's percentile (defined here as MR1 divided by the total of reference tracks, in percent) instead of MR1, because the number of reference tracks varies with N.

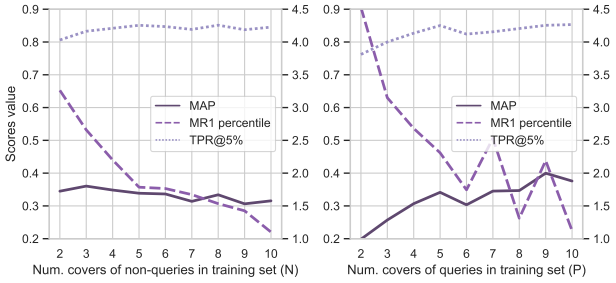


Figure 6: Query scores for different training/reference sets. Left: increasing N, number of covers of non-queries, P fixed. Right: increasing P, number of covers of queries, N fixed. (Y-axes scale MAP and TPR on the left, and the MR1 percentile on the right).

The MAP only slightly decreases as N increases, which indicates that the precision remains stable, even though the number of examples to sort and to rank is increasing. Moreover, the MR1 percentile and the TPR@5% clearly improve as N increases. As P is fixed, it means that the ranking and the separation between covers and non-covers clusters is improving as the non-queries clusters are consolidated, which illustrates the expected *pushing* effect.

Pulling covers We reproduce the same protocol again, but now with N=5 fixed and for different values of $P \in [2, ..., 10]$. We report results on Figure 6 (right). It appears clearly that all metrics improve steadily as P increases, even though the actual query itself has never been seen during training. As N is fixed, this confirms the intuition that the model will get better in locating unseen tracks closer to their work's cluster if trained with higher number of covers of this work, which illustrates the expected *pulling* effect.

5.3 Operational meaning of $p_c(d)$ and $p_{nc}(d)$

We now investigate further the distance distributions of cover and non-cover pairs. To this aim, we randomly split our entire dataset into a query and a reference set with a 1:5 ratio (resp. 10385 and 51925 tracks). Query tracks are thus not seen during training, but might have zero or more covers in the reference set.

Covers probability Computing queries vs. references pairwise distances gives the distributions $p_c(d)$ and $p_{nc}(d)$ shown on Figure 7 (left). Using Bayes' theorem, it is straightforward to derive from $p_c(d)$ and $p_{nc}(d)$ the probability for a pair of tracks to be covers given their distance d (Figure 7, right). This curve has an operational meaning, as it maps a pair's distance with a probability of being covers without having to rank it among the entire dataset.

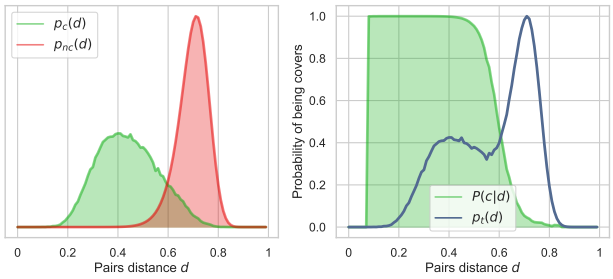


Figure 7: Left: separation of $p_c(d)$ (green) and $p_{nc}(d)$ (red). Right: probability of being a covers pair given the distance d (green) and total pairs distance distribution $p_c(d)+p_{nc}(d)$ (blue).

Easy and hard covers We repeat the previous test five times with random splits, and report metrics in Table 1. At first sight, MR1 and MT@10 could seem inconsistent, but a closer look at the results gives an explanation. To illustrate what happens, imagine a set of five queries where the first query ranks ten covers correctly in the first ten positions, e.g. because they are all very similar, while all other four queries have their first correct answer at rank 100. This would yield to MT@10=2.0, and MR1=80.2. This kind of discrepancy between MR1 and MT@10 reflects the fact that some works in our dataset have similar covers that are easily clustered, while other are much more difficult to discriminate. This can be observed on the positive pairs distribution $p_c(d)$ on Figure 7 (left), which is spread over a large range of distances.

	MAP	MT@10	MR1	BC	AuC	TPR@5%
Proposed	0.39 (<0.01)	2.90 (0.03)	581 (59)	0.40 (<0.01)	0.97 (<0.01)	0.88 (<0.01)

Table 1: Results of query set lookup in reference set.

6. COMPARISON WITH OTHER METHODS

6.1 Comparison on small dataset

We first compared with two recent methods [31, 34], who reported results for a small dataset of 50 works with 7 covers each. The query set includes five covers of each work (250 tracks), while the reference set includes each work's remaining two covers (100 tracks). As this dataset is not publicly available anymore, we have mimicked it extracting randomly 350 tracks out of own dataset [3] .

Our data-driven model can however not be trained with only 100 tracks of the reference set, as it would overfit immediately. We have thus trained our model on our full dataset, with two different setups: **a)** excluding the 350 tracks reserved for the query and reference sets. **b)** excluding the 250 tracks of the query set, but including the 100 tracks of the reference set. We repeated this operation ten times for each setup, and report the mean and standard deviation on Table 2 for the same metrics used in [31, 34], as well as the p-value obtained by a statistical significance t-test carried out on results series.

	MAP	P@10	MR1
[34]	0.591	0.140	7.910
[31]	0.648	0.145	8.270
Proposed **a)**	**0.675**	**0.165**	**3.439**
	(0.04), p=.29	(0.005), p<.001	(1.062), p<.001
Proposed **b)**	**0.782**	**0.179**	**2.618**
	(0.104), p<.01	(0.014), p<.001	(1.351), p<.001

Table 2: Comparison between recent method [31, 34] and our proposed method on a small dataset (precision at 10 P@10 is reported instead of MT@10. As there are only two covers per work in the reference set, P@10 maximum value is 0.2).

Our method significantly improve previous results: for the hardest case **a)** where the model has not seen any queries work during training, embeddings space has been sufficiently structured to discriminate the unseen works from the other training clusters (*pushing* effect). For the easier case **b)**, the *pulling* effect from the known queries covers provides further improvement.

6.2 Comparison on large dataset

We also compared with [16], who is to our knowledge the last attempt to report results for thousands of queries and references – a more realistic use case. This paper reported results on the SecondHandSong (SHS) subset of the MillionSong dataset (MSD) [5] for two experiments: **a)** only the training set of 12960 covers of 4128 works was used both as the query and reference sets. **b)** the SHS MSD test set of 5236 covers of 1726 works was used to query the entire MSD used as reference.

The SHS MSD is not available anymore. However, as our dataset has also been built from the SHS covers list, we consider that results can be compared [3] . We have therefore randomly generated out of our dataset a training and a test set mimicking the original ones. We trained our model on the training set, and perform the pairwise distances computation between the query and reference sets (as the query set is included in the reference set, we excluded for comparison the pairs of the same track). For experiment **b)**, we have used our entire dataset as reference set as we do not have one million songs. We have repeated this operation five times and report in Table 3 the mean and standard deviations for the same metrics used in [16], as well as MR1, MT@10 and the p-value of the t-test carried out.

		MAP	MR	MT@10	MR1
a)	[16]	0.285	1844	-	-
	Proposed	**0.936**	**78**	2.010	33
		(0.001), p<.001	(6), p<.001	(<0.001)	(3)
b)	[16]	0.134	173117	-	-
	Proposed	**0.220**	**3865**	1.622	430
		(0.007), p<.001	(81), p<.001	(0.003)	(19)

Table 3: Comparison between method [16] and our proposed method on a large dataset (MR=Mean rank). For **b)**, the MR percentile should be compared, as our reference set does not have 1M tracks (6^{th} vs. 17^{th} percentile for [16]).

Our method significantly improve previous results. For case **a)**, results are notably good, which is not surprising as the model has already seen all the queries during the training. Case **b)** is on the other hand the hardest possible configuration, where the model has not seen any covers of the queries works during training, and clusterisation of unseen tracks entirely relies on the *pushing* effect.

As to our method's computation times, we observed on a single Nvidia GPU Titan XP for a ~3 mn audio track: ~10 sec for F0 extraction, ~1 sec for embeddings computation, and less than 0.2 sec for distances computation with the full dataset embeddings (previously computed offline).

7. CONCLUSION

In this work, we presented a method for cover detection, using a convolutional network which encodes each track as a single vector, and is trained to minimize cover pairs Euclidean distance in the embeddings space, while maximizing it for non-covers. We show that extracting embeddings out of the dominant melody 2D representation drastically yields better results compared to other spectral representations: the convolutional model learns to identify similar patterns in the dominant melody at different scales and locations (tempo, key and structure invariance).

We have also shown that our method scales to audio databases of thousands of tracks. Once trained for a given database, it can be used to assess the probability for an unseen track to be a cover of any known track without having to be compared to the entire database. We have finally shown that our method improves previous methods both on small and large datasets.

In the future, we plan to grow our training dataset to address the realistic use case where collections of millions of tracks should be queried: as for many other data-driven problems, will the cover detection problem be solved if the embeddings space is sufficiently structured?

[3] We acknowledge that, strictly speaking, results can not be directly compared to rank methods, as original datasets of [34], [31] and [16] are not available any longer. They however reflect the level of performance of the proposed method.

Acknowledgements.

The dominant melody representations of tracks used as training dataset in this work is available upon request.

We are grateful to Xavier Costaz at Sacem for his suggestions and support, and to Philippe Esling at Ircam for fruitful discussions.

8. REFERENCES

[1] Pierre Baldi and Yves Chauvin. Neural networks for fingerprint recognition. *Neural Computation*, 5(3):402–418, 1993.

[2] Juan Pablo Bello. Audio-based cover song retrieval using approximate chord sequences: Testing shifts, gaps, swaps and beats. In *Proceedings of ISMIR (International Society of Music Information Retrieval)*, 2007.

[3] Thierry Bertin-Mahieux and Daniel PW Ellis. Large-scale cover song recognition using hashed chroma landmarks. In *Proceedings of IEEE WASPAA (Workshop on Applications of Signal Processing to Audio and Acoustics)*, pages 117–120. IEEE, 2011.

[4] Thierry Bertin-Mahieux and Daniel PW Ellis. Large-scale cover song recognition using the 2d fourier transform magnitude. In *Proceedings of ISMIR (International Society of Music Information Retrieval)*, 2012.

[5] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. *Proceedings of ISMIR (International Society of Music Information Retrieval)*, 2011.

[6] Anil Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.*, 35:99–109, 1943.

[7] Rachel M Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan P Bello. Deep salience representations for f0 estimation in polyphonic music. In *Proceedings of ISMIR (International Society of Music Information Retrieval)*, 2017.

[8] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744, 1994.

[9] Guillaume Doras, Philippe Esling, and Geoffroy Peeters. On the use of u-net for dominant melody estimation in polyphonic music. In *International Workshop on Multilayer Music Representation and Processing (MMRP)*, pages 66–70. IEEE, 2019.

[10] Daniel PW Ellis and Graham E Poliner. Identifying-cover songs with chroma features and dynamic programming beat tracking. In *Proceedings of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*. IEEE, 2007.

[11] Rémi Foucard, Jean-Louis Durrieu, Mathieu Lagrange, and Gäel Richard. Multimodal similarity between musical streams for cover version detection. In *Proceedings of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*. IEEE, 2010.

[12] Emilia Gómez and Perfecto Herrera. The song remains the same: identifying versions of the same piece using tonal descriptors. In *Proceedings of ISMIR (International Society of Music Information Retrieval)*, 2006.

[13] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of IEEE CVPR (Conference on Computer Vision and Pattern Recognition)*, volume 2, pages 1735–1742. IEEE, 2006.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[15] Eric J Humphrey, Juan Pablo Bello, and Yann LeCun. Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In *Proceedings of ISMIR (International Society of Music Information Retrieval)*, 2012.

[16] Eric J Humphrey, Oriol Nieto, and Juan Pablo Bello. Data driven and discriminative projections for large-scale cover song identification. In *Proceedings of ISMIR (International Society of Music Information Retrieval)*, 2013.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] Anssi Klapuri. Multiple fundamental frequency estimation by summing harmonic amplitudes. In *Proceedings of ISMIR (International Society of Music Information Retrieval)*, 2006.

[19] Matija Marolt. A mid-level representation for melody-based retrieval in audio collections. *IEEE Transactions on Multimedia*, 10(8):1617–1625, 2008.

[20] Benjamin Martin, Daniel G Brown, Pierre Hanna, and Pascal Ferraro. Blast for audio sequences alignment: a fast scalable cover identification. In *Proceedings of ISMIR (International Society of Music Information Retrieval)*, 2012.

[21] Brian McFee, Luke Barrington, and Gert Lanckriet. Learning content similarity for music recommendation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(8):2207–2218, 2012.

[22] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25, 2015.

[23] Xiaoyu Qi, Deshun Yang, and Xiaoou Chen. Triplet convolutional network for music version identification. In *International Conference on Multimedia Modeling*, pages 544–555. Springer, 2018.

[24] Colin Raffel and Daniel PW Ellis. Pruning subsequence search with attention-based embedding. In *Proceedings of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*. IEEE, 2016.

[25] Suman Ravuri and Daniel PW Ellis. Cover song detection: from high scores to general classification. In *Proceedings of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*. IEEE, 2010.

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[27] Christian Sailer and Karin Dressler. Finding cover songs by melodic similarity. *MIREX extended abstract*, 2006.

[28] Justin Salamon and Emilia Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1759–1770, 2012.

[29] Justin Salamon, Joan Serra, and Emilia Gómez. Tonal representations for music retrieval: from version identification to query-by-humming. *International Journal of Multimedia Information Retrieval*, 2(1):45–58, 2013.

[30] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of IEEE CVPR (Conference on Computer Vision and Pattern Recognition)*, pages 815–823, 2015.

[31] Prem Seetharaman and Zafar Rafii. Cover song identification with 2d fourier transform sequences. In *Proceedings of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*. IEEE, 2017.

[32] Joan Serrà. *Music similarity based on sequences of descriptors tonal features applied to audio cover song identification*. PhD thesis, Universitat Pompeu Fabra, Spain, 2007.

[33] Xavier Serra, Ralph G Andrzejak, et al. Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11(9):093017, 2009.

[34] Diego F Silva, Chin-Chin M Yeh, Gustavo Enrique de Almeida Prado Alves Batista, Eamonn Keogh, et al. Simple: assessing music similarity using subsequences joins. In *Proceedings of ISMIR (International Society of Music Information Retrieval)*, 2016.

[35] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 118–126, 2015.

[36] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, 2016.

[37] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of IEEE CVPR (Conference on Computer Vision and Pattern Recognition)*, pages 4004–4012. IEEE, 2016.

[38] TJ Tsai, Thomas Prätzlich, and Meinard Müller. Known artist live song id: A hashprint approach. In *Proceedings of ISMIR (International Society of Music Information Retrieval)*, 2016.

[39] Wei-Ho Tsai, Hung-Ming Yu, Hsin-Min Wang, et al. Query-by-example technique for retrieving cover versions of popular songs with similar melodies. In *Proceedings of ISMIR (International Society of Music Information Retrieval)*, 2005.

[40] Wei-Ho Tsai, Hung-Ming Yu, Hsin-Min Wang, and Jorng-Tzong Horng. Using the similarity of main melodies to identify cover versions of popular songs for music document retrieval. *Journal of Information Science & Engineering*, 24(6), 2008.

[41] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013.

[42] Emmanuel Vincent, Nancy Bertin, and Roland Badeau. Adaptive harmonic spectral decomposition for multiple pitch estimation. *IEEE Transactions on Audio, Speech and Language Processing*, 18(3):528–537, 2010.

[43] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, pages 521–528, 2003.

IDENTIFYING EXPRESSIVE SEMANTICS IN ORCHESTRAL CONDUCTING KINEMATICS

Yu-Fen Huang[1] **Tsung-Ping Chen**[1] **Nikki Moran**[2] **Simon Coleman**[3] **Li Su**[1]
[1]Music and Culture Technology Lab, Institute of Information Science, Academia Sinica, Taiwan
[2]Reid School of Music, University of Edinburgh, UK
[3]Institute for Sport, Physical Education and Health Sciences, University of Edinburgh, UK
yfhuang@iis.sinica.edu.tw

ABSTRACT

Existing kinematic research on orchestral conducting movement contributes to beat-tracking and the delivery of performance dynamics. Methodologically, such movement cues have been treated as distinct, isolated events. Yet as practicing musicians and music pedagogues know, conductors' expressive instructions are highly flexible and dependent on the musical context. We seek to demonstrate an approach to search for effective descriptors to express musical features in conducting movement in a valid music context, and to extract complex expressive semantics from elementary conducting kinematic variations. This study therefore proposes a multi-task learning model to jointly identify dynamic, articulation, and phrasing cues from conducting kinematics. A professional conducting movement dataset is compiled using a high-resolution motion capture system. The ReliefF algorithm is applied to select significant features from conducting movement, and recurrent neural network (RNN) is implemented to identify multiple movement cues. The experimental results disclose key elements in conducting movement which communicate musical expressiveness; the results also highlight the advantage of multi-task learning in the complete musical context over single-task learning. To the best of our knowledge, this is the first attempt to use recurrent neural network to explore multiple semantic expressive cuing in conducting movement kinematics.

1. INTRODUCTION

During orchestral conducting, conductors use their body movements to guide musicians' expressions of various features such as the tempo, dynamics, and articulation in music. Through these delicate nuances in their body movement, conductors are able to communicate their refined interpretations and expressive intentions of the musical work in question. As documented in pedagogical literature, 'conducting semantics' - a codified repertoire of movements bearing specific musical instructions - are broadly accepted as core knowledge, fundamental to conducting practice. [11,15,21,31]. However, in the scenario of actual conducting performances, conductors' movement styles are remarkably diverse. Moreover, while it is conventional common sense for experienced musicians to identify distinct musical features (e.g. phrasing, dynamics, articulation), these features associate with one another in the complex musical context, and conductors tend to communicate similar musical features using diverse strategies depending on the musical context, e.g., dynamic and phrase cuing may vary when conducted with different articulation patterns. As a result, existing music-movement coupling models are limited in some aspects [16, 28, 33]. The straightforward association which presumes that certain movement features can communicate specific musical traits, as stated in conducting pedagogy, has not been observed in such models.

The major challenges to overcome for current conducting movement research are: 1) to identify interpretable quantitative descriptors to represent movement features; 2) to construct a generalisable model, which is robust in identifying features such as phrase, dynamic, and articulation cuing in complex musical context, and which is tolerant to the flexibility of conducting performance and the differences between individual conductors. In this study, we approach both of the challenges by adopting machine learning algorithms. More specifically, to determine potential movement descriptors relevant to expressive musical features, we first applied a supervised feature selection technique, ReliefF [17–19], to three-dimensional body movement data. Based on such selected features, we then sought to jointly identify different types of expressive cuing in conducting movement, and thus trained a recurrent neural network (RNN) on the body movement data from various conductors to perform multi-task learning (MTL).

This study identifies the effective descriptors in conducting movement which are used to communicate musical features. As the pioneering attempt to apply RNN to music-movement coupling in conducting, we also verify the advantage for RNN framework with MTL to probe potential complex connections between various movement and musical elements, especially compared to previous works using other models. In the subsequent section, re-

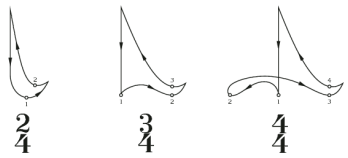


Figure 1. Basic metrical patterns in musical conducting. (reproduced from http://www.purposeful primarymusic.com).

lated research in musical conducting movement will be reviewed. Our dataset and model will be introduced in section 3 and 4 respectively. The experimental designs will be reported and the results will be discussed in section 5, followed by the conclusion in section 6.

2. RELATED WORK

For orchestral performance, a fundamental function of conducting is to coordinate musicians' playing. Conductors regulate the musical timing using basic beat patterns (e.g. two-beat, three-beat, or four-beat cycles etc., see Figure 1), and such beating movements communicate both the tempo arrangement and the metrical structure of the performed musical piece [11, 15, 21, 31]. The beating pattern in conducting has been substantially investigated, and it is found that the vertical action is the major component to lead the beat timing [4, 22, 24, 34, 35]. The tracking of beat is the most efficient under the constraint of movement bounding box area [35] and using dynamic time warping technique [34]. Based on these findings, interactive systems such as Pinocchio, vMaestro, and Personal Orchestra has built, in which the tempo of recorded orchestral audio is manipulated by the user's body movement, and the music automatically aligns with the beat timing identified in the movement [4, 22, 24].

Yet conducting movement is far more complicated than simple beating. Built on these basic metrical patterns, conductors take a step further to communicate their musical interpretations via refined variations in their movement. Pedagogical sources present a stock of movements frequently used by conductors to instruct their expressive intentions regarding articulation, dynamics, and phrasing [11, 15, 21, 31]. As summarised by the authors' previous work, specific conducting movements carrying expressive intentions, i.e. conducting semantics, are understood to comprise distinct combinations of hand position (high/ low/ away from the body/ close to the body), movement size (large/ small), speed (quick/ slow), acceleration (sudden/ gradual change of movement), smoothness (smooth/ jerky), trajectory shape (straight/ curved), and palm direction (upward/ downward/ facing musicians/ facing the conductor) [13, 14]. These qualitative, subjective descriptions of movement features match with quantitative, empirical analysis of conducting kinematics, in which the palm directions, movement size, hand positions and velocity reflect the dynamic change in music [8, 33, 35].

That conductors use their body movement to communicate musical expressiveness is self-evident, and is a truth demonstrated by both musical pedagogy and empirical analysis of conducting movement. However, existing music-movement coupling models generate only low to moderate correlations between musical and movement features [16, 33], and the computational approach which automatically classifies conductors' expressive intentions based on their movement has also produced unsatisfactory results [28]. These findings demonstrate that the expressive semantics in conducting are highly context-dependent. Different aspects of musical expression, such as dynamics, articulation, and phrasing are entangled in actual instances of musical performance, and their study would benefit from the ecologically-valid, intact musical contexts.

In music information retrieval (MIR) research, vast efforts have been devoted to extracting semantic representations such as pitch, beat, and harmony from audio signals. The machine learning approach has gained great success in modelling the semantics from music audio recordings [7]. During performance, musical sound is usually generated by the execution of body movement. There is a pressing need, therefore, for an equivalent effort to explore the expressive semantics carried by musical movement. The recurrent neural network (RNN) is widely used to analyse music semantics in a sequential way [25, 30, 32, 38]. RNN also allows the multi-task learning (MTL) setting, which has been proven successful in music information retrieval [12, 39]. We therefore propose a RNN framework with MTL approach in this study, and test the model on our conducting movement corpus.

3. DATA

This study is based on a professional conducting movement dataset collected by the authors, which contains conductors' upper body movement recorded by a high-resolution motion capture system, together with annotations of beat timing, phrase, dynamic, and articulation.

3.1 Collection of motion capture data

The motion capture data were recorded in the Biomechanical Laboratory at the Institute for Sport, Physical Education and Health Sciences (ISPEHS), University of Edinburgh, UK. Conductors' movement were collected using a nine-camera optical motion capture system (Qualisys, Pro-Reflex, Sweden) at a sampling frequency of 120 frames per second. The captured area was calibrated using the Qualisys 300 mm wand kit with the average residual being lower than 2 mm. Twenty-seven 12 mm optical markers were attached to the conductor's upper body and baton following the Golem Upper Body Model in Visual3D documentation [6]. The locations of 27 markers were illustrated in Figure 2 and listed in Table 1.

Six conductors (3 professional conductors and 3 advanced conducting students) participating the collection were all right-handed males, with an average conducting experience for 10.6 years (SD = 9.37), and conducted for 4.4 hours per week on average (SD = 2.38) at the time of participation. Five string musicians accustomed to musi-

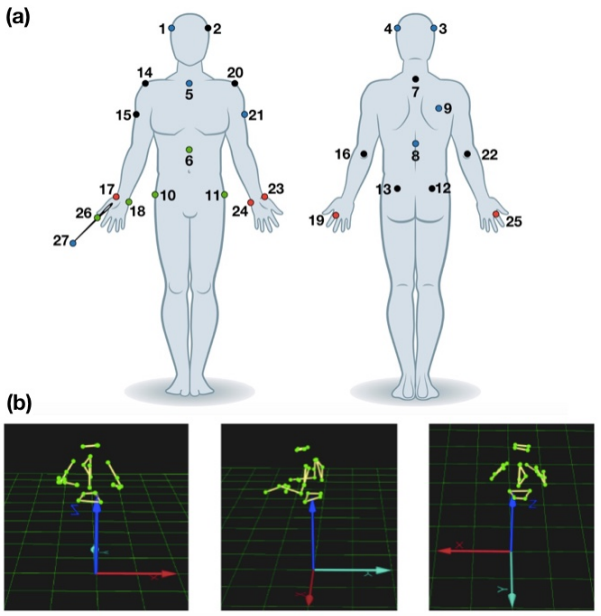


Figure 2. (a) The location of 27 optical markers placed on the conductor's upper body from the frontal and rear viewpoints. The marker colours represent the ranking from the feature selection procedure ReliefF: The top 15 elements (red), no 16-30 elements (green), no 31-45 elements (blue), no 46-81 elements (black). (b) The 27 markers captured by the system from the frontal, lateral, and rear viewpoints.

#	Name	Description
1	RFHD	Right temple
2	LFHD	Left temple
3	RBHD	Right back head
4	LBHD	Left back head
5	CLAV	Jugular Notch
6	STRN	Xiphoid process of the Sternum
7	C7	Spinous process of the 7th Cervical vertebrae
8	T10	Spinous process of the 10th thoracic vertebrae
9	RBAK	Middle of the right Scapula
10	RASI	Right Anterior Superior Iliac Spine
11	LASI	Left Anterior Superior Iliac Spine
12	RPSI	Right Posterior Superior Iliac Spine
13	LPSI	Left Posterior Superior Iliac Spine
14	RSHO	Right Acromio-clavicular joint
15	RUPA	Right upper arm
16	RELB	Right elbow joint
17	RWRA	Right wrist thumb side
18	RWRB	Right wrist pinkie side
19	RFIN	The 2nd Metacarpal of the right forefinger
20	LSHO	Left Acromio-clavicular joint
21	LUPA	Left upper arm
22	LELB	Left elbow joint
23	LWRA	Left wrist thumb side
24	LWRB	Left wrist pinkie side
25	RFIN	The 2nd Metacarpal of the right forefinger
26	BASH	Baton shaft
27	BAEN	Baton end

Table 1. The locations for 27 optical markers placed on the conductor's upper body

cal conductors' directions were recruited from University ensembles; the average number of years' instrumental experience was 15.6 (SD = 2.30) and average experience of playing in orchestra was 12.2 (SD = 3.11).

Each conductor rehearsed with musicians for 30 minutes. In the subsequent recording session, the conductor's movements were collected in a repeated measures design. The 6 conductors performed 3 different excerpts of Western classical orchestral repertoire: 1) W. A. Mozart, *Serenade in G major, K.525* (first movement, bars 1-55), 2) A. Dvořák, *Serenade in E Major, Op.22* (first movement, bars 1-53), and 3) B. Bartók, *Divertimento for String Orchestra, Sz. 133* (third movement, bars 1-183). The excerpts represent different compositional styles [5], yet share aspects of metrical structure, which lends them to comparison. Each excerpt is roughly 1 minute long according to constraints of the motion capture equipment. Individual conductors recorded 3 instances of each excerpt, with excerpt performance order counterbalanced across the 6 individuals. As a result, 54 conducting performances were collected in the corpus in total (3 performances x 3 musical excerpts x 6 conductors). The complete dataset was uploaded as the C3D file format for motion capture analysis to the DataShare open access data repository at the University of Edinburgh: https://datashare.is.ed.ac.uk/handle/10283/2906.

3.2 Data pre-processing and labelling

3.2.1 Pre-processing of motion capture data

The collected motion capture data were exported from Qualisys Tracker Manager (version 2.7, Pro-Reflex, Sweden) and imported to Visual3D (standard version 4.93, C-motion, USA) and Python (version 3.6.8) for further analysis. The original movement data containing the position on x-, y-, and z- axes of 27 markers were smoothed by the fourth-order low-pass Butterworth filter with a cut-off frequency of 10 Hz. The speed on x-, y-, z- axes of each marker was defined as the first derivative of x-, y-, z- position respectively, divided by the sampling interval (1/120 s). Speed was considered to carry important information in previous studies on musical conducting movement [13, 14, 26, 27], and thus was chosen as the variable to be investigated. The speed data were normalised based on the mean and standard deviation in each performance trial, and were converted to the z-scores of speed.

The z-scores of speed were then imported in the subsequent feature selection procedure ReliefF. Moreover, a generalisable element is preferable for RNN model. Considering that the minor fluctuations within 1/10 beats (roughly 5 frames) has only trivial effect on our target musical features – which are phrase, dynamic, and articula-

tion structure in bar level, and such minor fluctuations may, on the contrary, have negative effect on generalisation, the simple moving averages of speed data for per 5 frames were thus taken before being imported in RNN model.

In order to align motion capture data with expressive labels extracted from musical scores, the timing of beats and bars in conducting movement was assessed. The beat timing in movement was estimated using the motion capture analysis software Visual3D, and was defined as the time when the lowest position of baton tip on the z-axis (vertical axis) occurred within a beat period according to previous studies [13, 14, 26, 27]. The initiation of a musical bar is then defined as the onset of the first beat in the given bar.

3.2.2 Annotation of expressive labels

The label for dynamics, articulation, and phrasing were annotated by three experienced music theorists (with an average experience for music analysis = 20.7 years, SD = 2.89) based on designated edition of musical scores [2, 9, 29]. Where the annotators disagreed with each other, the opinion of the majority was taken. The dynamic and articulation labels were appointed to each musical bar. Phrasing labels, on the other hand, indicate the initiation timing of phrases, which is dissimilar to dynamic and articulation status that prolongs for a period of time. The phrase initiation was thus determined as the timing of the first beat in each phrase, and then entered the model with a window size of +/- 60 frames, which is equivalent to +/- 0.5 seconds of the phrase initiation in the given sample frequency 120 fps, and roughly +/- 1 beat of the phrase initiation according to 120 bpm instructed in musical scores. The labels are illustrated in Figure 3. The annotation process complied with the following principles:

1) Dynamics: The dynamic annotation contains 6 classes including: *pp, p, mp, mf, f, ff* in the 3 music excerpts. Dynamic levels were labelled according to the expressive terminology in the scores. Where there is no dynamic instruction specified in the given bar, the dynamic level in the previous bar was taken. Where the dynamic level changes within a bar, the dynamic level for the majority of beats was taken. Where different dynamic levels exist in equal number of beats within a bar, the first dynamic level occurring in the given bar was taken. Where crescendo or diminuendo is marked, the gradual change of dynamics was divided into equal levels and was designated to the bars in the crescendo or diminuendo process.

2) Articulation: The articulation annotation contains 3 classes including: *legato, neutral, staccato*. The legato label was assigned to where the slur is printed; the staccato label was assigned to where the staccato mark is printed in the score, or where a note equals to or shorter than a quaver is followed by a rest; the neutral label was assigned to bars where no specific legato or staccato marks was printed. Where there are more than one type of articulation terms printed within a bar, the articulation type for the majority of beats was taken. Where different articulation types exist in equal number of beats within a bar, the first articulation type occurring in the given bar was taken.

3) Phrasing: The phrasing annotation specifies the initiation of phrases, and it contains 2 classes including: *the phrase onset, none*. The phrase structure was determined according to conventional music analysis techniques [3]. Where different melodies interlocked, the phrase in the main melody is taken.

4. DATA ANALYSIS AND MODEL

The aforementioned data set was analysed in two steps: 1) the supervised feature selection technique ReliefF was applied to identify effective descriptors in conducting movement to communicate musical expressiveness; 2) the RNN architecture with MTL setting were constructed to model generalisable rules to associate multiple movement and musical features.

4.1 Data representation

The input data is represented as the z-score of speed on x-, y-, z- axes of 27 markers (3 x 27 = 81 features) with the sample frequency of 120 fps. As shown in Figure 4, each segment fed into the bidirectional long-short-term memory (BLSTM) network contains 81 features with 121 frames (which is equivalent to roughly 1 second or 2 beats); the hop size for consecutive segments is 30 frames (0.25 seconds or 0.5 beats); each input clip contains 64 segments (16.75 seconds or roughly 32 beats; 8 bars in Mozart and Dvořák; 32 bars in Bartók). In addition, we also imported the top 15 movement elements selected by ReliefF (15 features x 121 frames per segment) into RNN, to examine if our recognition model is capable of identifying musical expressions from a small assemble of crucial features.

4.2 Feature selection for movement data

Orchestral conducting consists of movements from different body parts. Conducting pedagogy mainly describes the expressive guidance instructed by hand movements, yet no evidence has been provided by previous motion capture studies to verify such emphasis on hands. To this end, it is essential to know which parts of the body and what kinds of movements are relevant to the expressive intentions in music. The process of finding such relevance can be regarded as a feature selection problem. The ReliefF algorithm, one of the most commonly used supervised feature selection algorithms for music related movement [23], is applied here as an exploratory study. ReliefF is an extension from the original Relief [17–19] with higher reliability and is applicable to multi-class datasets. The algorithm searches every class of features x_i for their k-nearest neighbours from the same class (nearest hit) and from a different class (nearest miss) to score how well such feature to distinguish data from different classes [19]: $\text{Score}_i \sim -\sum_k d(x_i, x_{\text{NearHit}}) + \sum_k d(x_i, x_{\text{NearMiss}})$, where $d(\cdot,\cdot)$ is a distance measure. The exact form of ReliefF can be found in [23]. In this research, we set $k = 20$ as suggested by Urbanowicz *et al.* [37].

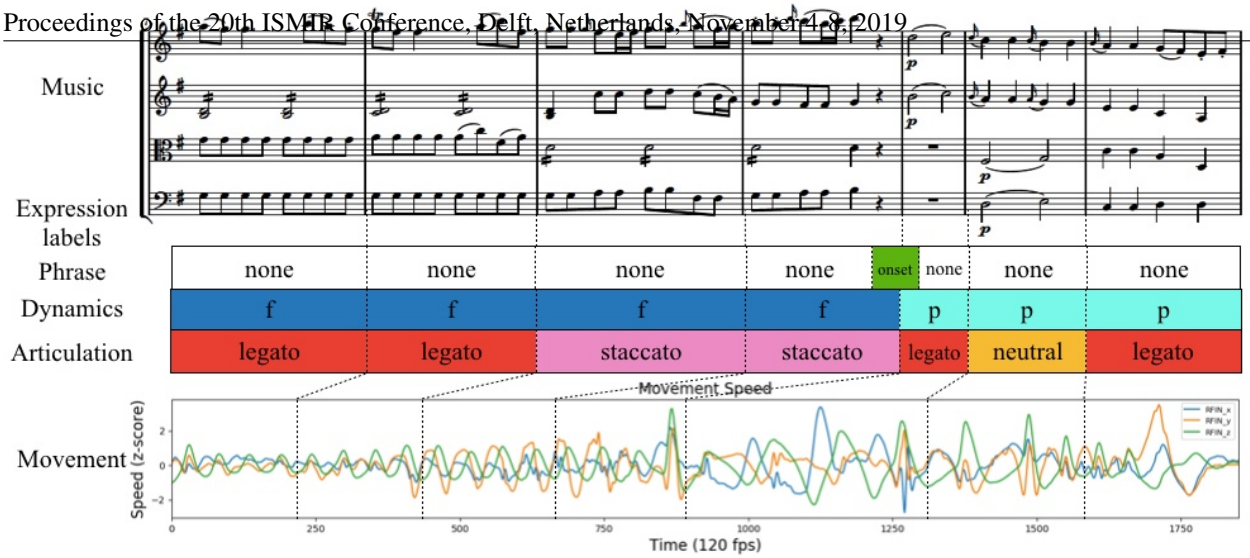


Figure 3. The example of musical scores, 3 types of expression label (phrase onset, dynamic level, articulation), and the movement data from right finger marker.

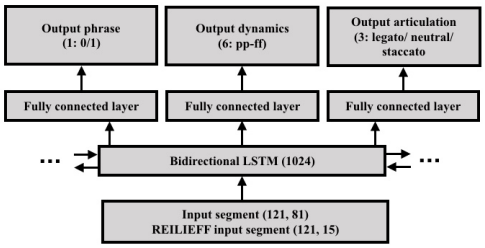


Figure 4. The multi-task model built on the RNN

4.3 Recognition model for music expression

We applied a recurrent neural network (RNN) with bidirectional long-short-term memory (BLSTM) cells to carry out multi-task learning (MTL) for expressive cuing in conducting movement. Such architecture is advantageous to determine sequential features in musical signals [7, 30, 32, 38], yet to the best of our knowledge, this approach has not been applied to musical movement in any existing study.

As shown in Figure 4, the model has a shared BLSTM layer with 1024 hidden units, and a task-specific fully-connected layer. The outputs from forward and backward LSTMs are concatenated as a 2-by-1024 matrix in the segment level, which is then flattened to link the fully-connected layer. The output layer is a 10-D vector containing the classes for the three tasks. Sigmoid is applied to output phrase initiation (1D); Softmax is applied to output the dynamic level (6D) and the articulation type (3D).

To examine the advantage of MTL, we also constructed single-task learning (STL) models, where the same RNN framework and BLSTM cells are applied to phrase, dynamics, and articulation tasks respectively.

5. EXPERIMENT

5.1 Experimental settings

The 54 conducting performance trials (810 input clips) are divided into training set (48 trials, 720-722 clips) and testing set (6 trials, 88-90 clips). The 9-fold cross-validation is performed in a way that per 6 trials are in turns assigned to the testing set in 9 experiments. Each clip fed into the BLSTM network contains 64 segments (# of feature x 121 frames per segment); the hop size for consecutive clips are 32 segments. To prevent the problem of over-fitting, data augmentation is performed in a way that random Gaussian noise is added to the original data with the level of 0.1 standard deviation of the input data.

All networks are implemented using TensorFlow [1], and are trained using stochastic gradient descent with Adam optimiser. The cross-entropy between output and labels are computed for training purpose. To avoid over-fitting, L2 regularisation is applied. The dropout rate for both the input and output of BLSTM cells is 0.7. The learning rate is 0.0001 with 4800 training steps. This procedure is performed on 4 models: the MTL model, STL models for phrase, dynamics, and articulation tasks respectively.

5.2 Evaluation metrics

For the phrase recognition task, The Precision, Recall, and F1 measures (the balanced harmonic mean of Precision and Recall) [10] are computed in segment level (hop size = 30 frames; 0.25 seconds; roughly 0.5 beats). A detected phrase cuing event is considered as a true positive if it lies within a tolerance window +/- 60 frames (0.5 second and roughly 1 beat) from the ground truth annotation. If there are two or more phrase cuing detected within this tolerance window, one of the detections is considered as a true positive, and others are considered as false positives. For dynamic and articulation recognition tasks, the accuracies (acc) are computed in segment level: acc = (# of true posi-

tive segments/ # of total segments).

5.3 Result

5.3.1 Results of movement feature selection

The top 15 relevant movement features selected by ReliefF algorithm to fit 3 types of musical features (phrase, dynamics, articulation) are illustrated in Figure 2. It appears that the elements from right and left fingers, and right and left wrists are prominent in communicating all three types of musical expressions. These top 15 elements are inputted in the subsequent RNN model.

It is an unexpected outcome that the elements from the baton end are not included in the top list. As suggested in conducting pedagogy, the baton end has been considered as an important reference to communicate expressive information in conducting [11, 15, 21, 31]. In the ReliefF analysis, the elements from baton end are ranked as no 31-33 within 81 elements. It could be the effect that our target musical features are in bar level rather than in beat level. According to eigenvalue and eigenvector analysis in previous research on musical movement, the movements in extremity usually correspond to lower-level musical elements with a shorter period (such as per beat), whereas movements in torso tend to associate with higher-level musical features with a longer period (such as per bar or longer) [36]. The right finger and wrist are in the intermediate location between the body and baton tip, and are the key body part for conductors to manipulate the baton, which could be the reason that why they contribute the most to our expressive targets.

5.3.2 Results of music expression recognition

The results of using RNN framework to perform MTL and STL are reported in Table 2. It is manifest that the accuracy produced by all models is higher than the random incidence (0.166 for 6-class dynamics; 0.333 for 3-class articulation). All the t-tests comparing the models with the feature selection procedure (# of feature = 15) and with full feature sets (# of feature = 81) do not reach the significant p-value of 0.05, which suggests that the top 15 features selected by ReliefF are effective descriptors regarding the expressiveness in conducting movement. It is evident from t-tests that the MTL model shows its advantage over STL models. It appears that these three musical features are intertwined in the musical context. Identifying phrase cuing can be the most challenging one among the three tasks. Considering the characteristics of movement kinematic signal, the phrase cuing can be easily confused with local-level beat cuing. Yet in the multi-task model, the dynamic and articulation information can help the recognition of phrase in one way or another.

The majority of previous research on conducting movement focuses on the beating pattern in basic level [24, 34, 35], and in this study, we make effort to explore the higher-level expressive semantics in conducting. There are several previous attempts to tackle the semantic-level in conducting, but exclusively target on the dynamic instructions, and tend to consider such gestures as isolated events regardless

Model	#	Phrase			Dync.	Artc.
		P	R	F	acc	acc
MTL	81	**60.39** **	**42.86** *	**48.48**	71.49 ***	76.45 ***
MTL	15	59.94 ***	42.10	47.63	**73.03** ***	**76.97** ***
STL	81	52.35 **	38.15 *	44.07	65.16 ***	70.25 ***
STL	15	48.05 ***	40.78	43.87	64.46 ***	68.95 ***

Table 2. Recognition results (in %) using RNN with multi-task setting (MTL) and single-task setting (STL): The precision (P), recall (R), F1 (F) measures for the phrase task, and the accuracy (acc) for dynamics (dync.) and articulation (artc.) tasks, comparing all movement features (# = 81) and the top 15 features selected by ReliefF (# = 15). Asterisks indicate the p-value of t-test of MTL and STL counterparts: * for $p < 0.05$; ** for $p < 0.01$; *** for $p < 0.001$.

the musical context [4, 8]. Previous studies examined the correlations of movement-music dynamic feature pairs and yield moderate r^2 ranging from 0.4 - 0.5 [33]. Our model takes another approach and is competent to investigate the complex inter-connections among multiple factors, and is able to produce further and solid results. As we expect from previous MIR research using MTL settings [12, 39], our MTL model demonstrates the advantage to consider multiple musical and movement elements together to investigate the complex inter-connections in the communication process during conducting performance. Moreover, as suggested by the previous research, musicians may perform the dynamics and articulation differently from the notated scores [20], the connection between the conducting movement and the performed sound can be further investigated using a similar approach presented in this study.

6. CONCLUSION AND FUTURE WORK

In this paper, we describe results from our investigation into the expressive semantics in conductors' movement used to communicate the phrase, dynamics, and articulation in music. The supervised feature selection technique ReliefF provides insights into effective descriptors in elementary kinematic signals of conducting movement. The movement elements from the right and left hand, and right and left wrists appear to carry important information regarding the conductor's expressive intentions. These selected descriptors are further investigated using recurrent neural network with multi-task learning. The RNN architecture yields improved results compared to previous works using other analysis techniques. Particularly, the multi-task learning model demonstrates a promising approach to examine the complex interactions among multiple musical and movement elements.

As the pioneering investigation on conducting movement using RNN, we highlight the potential for this framework to be applied to further explore other issues in music conducting, such as the connection between the conducting movement and the performance sound. The findings of such studies can enlighten the musical education for both conductors and orchestral musicians.

7. ACKNOWLEDGEMENT

Yu-Fen Huang is supported by the Postdoctoral Fellows Program of Academia Sinica, Taiwan. This work is partially supported by the Automatic Music Concert Animation (AMCA) project of the Institute of Information Science, Academia Sinica, Taiwan.

8. REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] B. Bartók. *Divertimento for String Orchestra, Sz.113*. Boosey & Hawkes, 1999.

[3] I. Bent. *Music Analysis in the Nineteenth Century: Volume 2, Hermeneutic Approaches*, volume 2. Cambridge University Press, 2005.

[4] B. Bruegge, C. Teschner, P. Lachenmaier, E. Fenzl, D. Schmidt, and S. Bierbaum. Pinocchio: conducting a virtual symphony orchestra. In *Proc. of the international conference on Advances in computer entertainment technology*, pages 294–295, 2007.

[5] P. Burkholder and D. Grout. *A History of Western Music: Ninth International Student Edition*. WW Norton & Company, 2014.

[6] C-motion. *Visual3D Wiki Documentation*. https://www.c-motion.com/v3dwiki/index.php?title=Tutorial:_Plug-In_Gait_Full-Body, access 2018.

[7] T.-P. Chen and L. Su. Functional harmony recognition of symbolic music data with multi-task recurrent neural networks. In *Proc. of the 19th International Society for Music Information Retrieval Conference, ISMIR*, pages 90–97, 2018.

[8] D. Dansereau, N. Brock, and J. Cooperstock. Predicting an orchestral conductor's baton movements using machine learning. *Computer Music Journal*, 37(2):28–45, 2013.

[9] A. Dvořák. *Serenade No.1, Op.22*. Dover Publications, 2001.

[10] C. Goutte and E. Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *European Conference on Information Retrieval*, pages 345–359. Springer, 2005.

[11] E. Green, M. Gibson, and N. Malko. *The Modern Conductor*. Pearson Education, Upper Saddle River, NJ, 2004.

[12] P. Hamel, M. Davies, K. Yoshii, and M. Goto. Transfer learning in mir: Sharing learned latent representations for music audio classification and similarity. In *Proc. of the 14th International Society for Music Information Retrieval Conference, ISMIR*, pages 9–14, 2013.

[13] Y.-F. Huang. *Connecting Conductor Gesture to Compositional Features and Conductors' Expressive Intentions: An Exploratory Kinematic Study*. PhD Thesis, University of Edinburgh, 2018.

[14] Y.-F. Huang, S. Coleman, E. Barnhill, R. MacDonald, and N. Moran. How do orchestral conducting movement kinematics communicate musical structures and interpretational intentions. *Psychomusicology*, 27(3):148–157, 2017.

[15] D. Hunsberger and R. Ernst. *The Art of Conducting*. McGraw-Hill, New York, NY, 1992.

[16] T. Kelkar, U. Roy, and A. Jensenius. Evaluating a collection of sound-tracing data of melodic phrases. In *Proc. of the 19th International Society for Music Information Retrieval Conference, ISMIR*, pages 74–81, 2018.

[17] K. Kira and L. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proc. of AAAI*, volume 2, pages 129–134, 1992.

[18] K. Kira and L. Rendell. A practical approach to feature selection. In *Machine Learning Proc.*, pages 249–256. Elsevier, 1992.

[19] I. Kononenko. Estimating attributes: analysis and extensions of relief. In *Proc. of European Conference on Machine Learning*, pages 171–182. Springer, 1994.

[20] K. Kosta, O. Bandtlow, and E. Chew. Dynamics and relativity: Practical implications of dynamic markings in the score. *Journal of New Music Research*, 47(5):438–461, 2018.

[21] J. Labuta. *Basic Conducting Techniques*. Pearson Education, Upper Saddle River, NJ, 2003.

[22] E. Lee, H. Kiel, S. Dedenbach, I. Grüll, T. Karrer, M. Wolf, and J. Borchers. Isymphony: an adaptive interactive orchestral conducting system for digital audio and video streams. In *CHI'06 Extended Abstracts on Human Factors in Computing Systems*, pages 259–262. ACM, 2006.

[23] J. Li, K. Cheng, S. Wang, F. Morstatter, R. Trevino, J. Tang, and H. Liu. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):94, 2018.

[24] Y. K. Lim and W. S. Yeo. Smartphone-based music conducting. In *Proc. of the International Conference on New Interfaces for Musical Expression , NIME*, pages 573–576, 2014.

[25] Z. Lipton, J. Berkowitz, and C. Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[26] G. Luck and S. Nte. An investigation of conductors' temporal gestures and conductor—musician synchronization, and a first experiment. *Psychology of Music*, 36(1):81–99, 2008.

[27] G. Luck and P. Toiviainen. Ensemble musicians' synchronization with conductors' gestures: An automated feature-extraction analysis. *Music Perception: An Interdisciplinary Journal*, 24(2):189–200, 2006.

[28] P. Modler and T. Myatt. Recognition of separate hand gestures by time-delay neural networks based on multistate spectral image patterns from cyclic hand movements. In *Proc. of IEEE International Conference on Systems, Man and Cybernetics*, pages 1539–1544, 2008.

[29] W. Mozart. *Eine Kleine Nachtmusik K.525*. Dover Publications, 2012.

[30] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan. This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, pages 1–13, 2018.

[31] M. Rudolf. *The Grammar of Conducting: A practical Study of Modern Baton Technique*. Schirmer, New York, NY, 1995.

[32] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017.

[33] Á. Sarasúa and E. Guaus. Dynamics in music conducting: A computational comparative study among subjects. In *Proc. of the International Conference on New Interfaces for Musical Expression , NIME*, pages 195–200, 2014.

[34] R. Schramm, C. Jung, and E. Miranda. Dynamic time warping for music conducting gestures evaluation. *Proc. of IEEE Transactions on Multimedia*, 17(2):243–255, 2015.

[35] L.-W. Toh, W. Chao, and Y.-S. Chen. An interactive conducting system using kinect. In *Proc. of IEEE International Conference on Multimedia and Expo, ICME*, pages 1–6. IEEE, 2013.

[36] P. Toiviainen, G. Luck, and M. Thompson. Embodied meter: hierarchical eigenmodes in music-induced movement. *Music Perception: An Interdisciplinary Journal*, 28(1):59–70, 2010.

[37] R. Urbanowicz, M. Meeker, W. La Cava, R. Olson, and J. Moore. Relief-based feature selection: introduction and review. *Journal of biomedical informatics*, 85:189–203, 2018.

[38] C. Walder and D. Kim. Neural dynamic programming for musical self similarity. *arXiv preprint arXiv:1802.03144*, 2018.

[39] M.-H. Yang, L. Su, and Y.-H. Yang. Highlighting root notes in chord recognition using cepstral features and multi-task learning. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA*, pages 1–8, 2016.

THE ROMANTEXT FORMAT: A FLEXIBLE AND STANDARD METHOD FOR REPRESENTING ROMAN NUMERAL ANALYSES

Dmitri Tymoczko[1] **Mark Gotham**[2] **Michael Scott Cuthbert**[3] **Christopher Ariza**[4]
[1] Princeton University, NJ [2] Cornell University, NY [3] M.I.T., MA [4] Independent
dmitri@princeton.edu, mark.gotham@cornell.edu, cuthbert@mit.edu

ABSTRACT

Roman numeral analysis has been central to the Western musician's toolkit since its emergence in the early nineteenth century: it is an extremely popular method for recording subjective analytical decisions about the chords and keys implied by a passage of music. Disagreements about these judgments have led to extensive theoretical debates and ongoing controversies. Such debates are exacerbated by the absence of a public corpus of expert Roman numeral analyses, and by the more fundamental lack of an agreed-upon, computer-readable syntax in which those analyses might be expressed. This paper specifies such a standard, along with an associated code library in `music21`, and a preliminary set of example corpora. To frame the project, we review some of the motivations for doing harmonic analysis, some reasons why it resists automation, and some prospective uses for our tools.

1. INTRODUCTION AND MOTIVATION

Roman numeral analysis represents tertian chords by their triad type (major, minor, diminished, augmented), their position relative to the tonic (specified by the scale degree of their root), their bass note or inversion, and the presence of sevenths or other added or altered notes. The practice emerged in the early nineteenth century, with Gottfried Weber's *Versuch einer geordneten Theorie der Tonsetzkunst* [19] drawing on Rameau's earlier concept of the fundamental bass. Weber's method was so immediately popular that he complained other theorists were stealing his methods, and it has remained common to the present day. It is useful to contrast Roman numerals with alternatives, notably:

- Absolute labels for chords such as 'C', as often found in lead sheets;
- Function-theoretic labels such as 'T[onic]';
- Inversional symbols accompanying bass notes as used in figured-bass notation.

Absolute chord labels are performer-oriented in the sense that they specify which notes should be played, but do not provide any information about their function or meaning: thus one and the same symbol can represent a tonic, dominant, or subdominant chord. Accordingly, these labels obscure a good deal of musical structure: a dominant-functioned G major chord (i.e. G major in the local key of C) is considerably more likely to descend by perfect fifth than a subdominant-functioned G major (i.e. G major in the key of D major). This sort of contextual information is readily available to both trained and untrained listeners, who are typically more sensitive to relative scale degrees than absolute pitches.

Roman numerals include contextual information at the cost of increased subjectivity: by labelling chords relative to a local tonic, they require the analyst to make a potentially difficult decision about what the tonic is. However, these decisions can be undone: given a Roman numeral and a key, one can algorithmically derive an absolute chord label. Thus Roman numerals can be used even in contexts where absolute labels are needed (for instance, exploring the frequency of "open" chords in guitar-based music). This makes it a good choice for the construction of analytical corpora, as one can translate from Roman numerals to absolute labels but not vice versa.

Function theoretical labels go one step further toward abstraction: here chords are identified not by their note content but by their perceived harmonic role. (Usually, the three Riemannian functions T, S, D are employed, though other writers have suggested additional harmonic categories.) Thus in C major, a term like "D[ominant]" could mean B-D-F, G-B-D-[F], or some other chord entirely; just as "S[ubdominant]" could mean either D-F-A-[C] or F-A-C. Because many distinct harmonies map to a single functional term, these labels cannot be translated into Roman numerals or absolute chord labels. However, function labels can often be recovered from Roman numerals: in ordinary musical contexts, vii and V are Dominants, while ii and IV are Subdominants [17]. Once again the asymmetry gives us reason to prefer Roman numerals for analytical corpora.

The inversional symbols of figured-bass notation are in many ways analogous to absolute chord labels, but with an important difference: they are often used to label "non-harmonic" sonorities containing dissonances that do not belong to the underlying harmony. For this reason, many recent theorists have favored them, [5, 10] be-

lieving that Roman numerals impose too strong a distinction between the harmonic and non-harmonic realm. (Ironically, Riemann himself made the converse complaint against figured-bass notation, believing that it did not draw a sharp enough distinction between harmonic and non-harmonic [13]). When we restrict our attention to tertian sonorities, however, figured-bass symbols are largely equivalent to absolute chord labels, telling performers what notes to play but not what they mean.

Each of these methods has its benefits and shortcomings; we have chosen Roman numerals for their widespread popularity, their translatability to the other formats, and their proven utility.

1.1 Motivation for Roman Numeral Analysis Corpora

Roman numeral analysis involves several different layers of subjective judgment:

- which notes are harmonic / non-harmonic;
- what complete harmony, if any, an incomplete or otherwise ambiguous chord might represent; and
- the underlying keys and when they change.

Thus Roman numeral analysis is not an automatic process, and humans may legitimately disagree about the best analysis of a given passage [14, 15, 18]. Instrumental textures present a particular challenge because it is not always clear where harmonies change, and even a single, monophonic passage such as an Alberti bass can imply multiple voices. Even in rhythmically simpler genres, such as the Bach chorales, there are significant challenges to automatic Roman numeral parsing.

One of us has written a rule-based Roman numeral analyzer that reproduces human analyses of the Bach chorales with roughly 82% accuracy (DT forthcoming work; code available on request). That figure represents something like the state of the art, and while 82% accuracy is sufficient for some applications, it falls far short of what is needed in serious analytical and pedagogical contexts. Furthermore, extending the method to instrumental textures poses a range of considerable challenges.

Such challenges motivate the construction of human-made corpora, which can serve as the "ground truth" for evaluating computational analyses. Human-made corpora can also allow us to study the degree of alignment among expert analysts, as well as providing important first-order information about harmonic practices in different historical eras. Finally, the combination of automatic analyses with machine-readable scores facilitates a host of analytical projects including the automatic identification of non-harmonic tones.

2. THE .RNTXT SPECIFICATION STANDARD

We now outline our proposed standard. Previous work to define standards and create corpora include [1, 3, 4, 7, 8, 11, 12, 16]. Our goal is to create a format that is not just parsable by a computer, but also easy for human beings to read and write. Previous work tends to neglect this latter consideration, and thus limits corpus creation to dedicated coders and researchers prepared to learn formats divergent from textbook models. The Clercq-Temperley and DCMLab's tabular representation are user-friendly exceptions; we provide converters as part of the `music21` code to connect with those format (as discussed below in Section 3.1) [2]. Additionally, [4] reports that TAVERN has plans for a forthcoming `music21` converter which would further extend the connections and interoperability between these corpora. We provide extensive online examples (cf. Section 3.2) below and include one extract in the text as Figure 1.

```
Composer: J. S. Bach
Piece: Chorale BWV269
Analyst: Dmitri Tymoczko
Proofreader: David Castro

Time Signature: 3/4

Form: chorale
m0 b3 G: I
m1 b2 IV6 b3 V6
m2 I b2 V b3 vi
m3 IV b2.5 viio6 b3 I
m4 V || b3 I
m5 V6 b2 vi6/5 b3 viio6
m6 I6 b2 ii6/5 b3 V b3.5 V7
m7 I || b3 I
m8 I b2 ii b2.5 viio6 b3 I6
m9 I6 b2.5 V4/3 b3 I
m10 V || b3 vi
Note: consecutive first inversion triads
m11 vi b2 iii6 b3 ii6
m11var1 vi b2 I6/4 b3 ii6
m12 I6 b3 V7
m13 I b2 I6 b3 V7/IV
m14 IV || b3 I
m15 V6 b1.5 V6/5 b2 I b3 viio6
m16 I6 b2 I b3 V b3.5 V7
m17 vi b2 IV b3 I
m17var1 vi b2 IV b2.5 viio6/4 b3.5 I
m18 V || b3 I
m19 V6 b3 IV6
m20 vi b2 ii6/5 b3 V b3.5 V7
m21 I
```

Figure 1. An example of the new standard as used to represent an analysis of the Bach chorale BWV269.

2.1 Metadata

Documents may begin with the following optional lines:

- Composer: e.g. 'Mozart'
- Piece: Name and/or catalogue number, e.g. 'K550'

- Analyst: The name of the original analyst / originator of this document, e.g. 'Jo Blogs'.
- Proofreader: The name(s) of the anyone involved in checking and correcting this work, e.g. 'Jo Vlogs'.

After the metadata, may include movement and time signature information on separate lines.

Movement. Use Arabic numbering to label the movement of a multi-movement piece.

```
Movement: 1
```

Each movement is typically contained in its own file.

Time Signature. Specify time signatures on a separate line in the form of a simple string, such as:

```
Time Signature: 6/8
```

Changes of time signature are notated with a new specification immediately preceding the change. If no time signature is specified 4/4 is assumed.

Key Signature. The format supports the (optional) specification of the notated key signature as a number of sharps (or negative number for flats) in the key signature. So the specification:

```
Key Signature: -1
```

stands for a key signature of one flat (F major or d minor).

Minor Sixth / Minor Seventh. Specifies how scale degrees 6 and 7 are to be interpreted in minor mode. Supported values are 'quality' (major chords are on flattened degrees, minor or diminished on raised degrees), 'cautionary' (default: same as quality, but # and b can used to leave off ambiguity), 'sharp' (raised degrees are standard; lowered degrees require b), and 'flat' (lowered degrees are standard). See github.com/MarkGotham/When-in-Rome for a tabular comparison of these options.

2.2 General syntax of Chords

After the metadata, the document proceeds to itemize each chord with one line per measure. Each line of analysis starts with the symbol 'm' followed immediately by a measure number with no space; it then proceeds with pairs of beat numbers (preceded by 'b') and their corresponding Roman numerals. For instance, the line:

```
m5 b1 IV6 b2 V
```

indicates that in measure 5, a IV^6 chord falls on beat 1, followed by a V chord on beat 2.

Chord Inversions. The format supports all standard representations of triad and seventh inversions, as itemized in Section 2.3. Slashes are optional for separating subscript from superscript numbers, so I^6_4 may be `I6/4` or `I64`.

Missing beats. If no beat is specified at the start of a line, beat 1 is assumed. For any further beats that are missing, the existing chord remains in effect. So the line:

```
m14 IV6 b2 V b4 V2
```

indicates that measure 14 starts with IV^6 chord on beat 1, followed by a V chord on beat 2 which remains in effect across beat 3, and becomes V^2 on beat 4.

No chord. 'NC' indicates a passage with no chord – where one chord terminates prior to the onset of the next.

Measure numbers. Each line begins with a measure number and each movement should be numbered separately. This means that every movement will start with measure 1, except in the case of an initial anacrustic measure which is numbered 0. (The `music21` parser interprets the length of the anacrusis based on the first notated beat: thus in 4/4, `m0 b4 C: V` indicates a quarter-note pickup, while `m0 b4.5 C: V` indicates an eighth-note pickup.) Thus measure 1 is always the first full measure. For multiple alternative measure numbers, such as first / second time repeats or endings, use lower-case Latin script, so '216a', '217a', '218a' for the first, and '216b', '217b', '218b' for the second. See 'Repeats' in Section 2.5 below for details of how to handle measures that repeat the harmonic context of other measures.

Beats. Beat numbering follows the conventions for the given meter, so 4/4 has 4 beats in total, while 6/8 and 2/2 have two. Thus a succession of eighth notes will be assigned different beat positions depending on that context, for instance:

- in 4/4: 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5.
- in 2/2: 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75.
- in 6/8: 1, 1.33, 1.66, 2, 2.33, 2.66 (or 2.67).

The X.33 and X.66 format is also used for the second and third units in a triplet division of a duple meter. For further division of ternary beats, analysts can either use more precision or multiple decimal points, so that '1.833' and '1.66.5' indicate the point halfway through the second eighth of the first beat of a 6/8 measure.

Keys. Indicate key changes with the name of the key in the correct (upper or lower) case, followed by a colon. `C:` indicates the key of C major; while `c:` is c minor. Sharps are notated with '#' and flats with either 'b' or '-'. Key changes occur between beat and chord, so

```
m112 IV6 b4 C: ii
```

indicates a key change to C major on beat 4 of measure 112. A single chord can be notated as a pivot chord belonging to two keys like so:

```
m112 IV6 b4 vi C: ii
```

Here, one and the same (D minor) chord is recorded both as 'ii' in the new key of C major and 'vi' in the old key (F major). (Note the lack of a beat between the Roman numerals in the two keys.) There is no need to change keys for modally-inflected chords:

```
m112 IV6 b3 iv6
```

indicates a major triad $\hat{6}$-$\hat{1}$-$\hat{4}$ followed by the minor triad $\flat\hat{6}$-$\hat{1}$-$\hat{4}$ with the same root.

A key with a semicolon preceding the colon, such as `C;:` indicates that the new key should also be indicated with a change of key signature in any generated score.

In cases of key ambiguity, a secondary key may be preceded by a question mark and an open parentheses such as `?(Bb:` which indicates that the analyst believes that B-flat major is also a valid key but that the prevailing key (such as g-minor) is the one to which subsequent Roman numerals will refer. These optional keys can be concluded by reversing the direction of the parenthesis as in `?)Bb:`.

Chromatic Alteration. For altered chords, simply use the appropriate triadic symbol: lowercase with the suffix

'o' for diminished, lowercase for minor, uppercase for major, uppercase with the suffix '+' for augmented. Roots can be raised or lowered with a preceding '#' and 'b'(e.g. `iv`). Thus `bIII+` indicates an augmented triad on a lowered scale-degree 3, or Eb-G-B in C major.

Altered, added, and omitted tones. These may be indicated by an operation and chord step in brackets. Thus `V4/3[b5]` indicates a V_3^4 chord with whose bass is lowered. Note the potentially confusing (but standard) combination of figured-bass and root-functional symbols: in `V4/3[b5]` the '43' refers to intervals above the bass, while [b5] refers to an interval above the root. The symbol 'no' can be used to specify chord tones that are to be removed. Each alteration should appear in its own set of brackets. Thus, as a non-sensical demonstration of multiple possible features, `V65[no5][add#6][b3]` indicates Bb-E#-F-G in C major.

Minor mode. In minor, scale degrees 6 and 7 are variable depending on whether the melodic or natural minor collection is used. For the purpose of legibility the expectation is that common chords conform to one of the standard minor scales. Thus `a: VII` refers to G-B-D while `a: viio` refers to G#-B-D. Similarly, `a: VI` is F-A-C while `a: vio` is F#-A-C. Augmented chords are treated as altered major chords (i.e. based on natural-minor scale degrees), while diminished chords are treated as altered minor triads (raised degrees). These interpretations can be changed through the 'Sixth minor' and 'Seventh minor' metadata tag described above.

Applied Chords. 'Applied', or 'secondary', chords are notated using '/'. For instance, `V/V` indicates the dominant chord of the dominant key of the prevailing tonic, so D major (V of G major) in the key of C major. Applied chords are typically used for local tonicisations (including the purely diatonic V/III in minor), reserving 'full' key changes like `C:` for longer modulations. Chained applied chords such as `V7/V/V` are allowed.

Augmented and Neapolitan sixth chords. These are sufficiently canonical to be afforded their own abbreviation. Thus 'Ger*' is shorthand for `#ivo*[b3]` where * labels the appropriate inversion (7, 6/5, 4/3, or 2), 'It*' for `#ivo*[b3]` (with possible inversions being 5/3 or blank, 6/3 or 6, or 6/4), and 'Fr*' for `II*[bV]`. For the Neapolitan, 'N' and 'N6' are both accepted abbreviations for `bII6`. Use `bII` for the Neapolitan in root position.

Cadential 6/4. For computational simplicity, the parser expects any of `I64`, `I6/4` or `Cad64` for the cadential 6/4 chord (`V64` is not treated as equivalent here since its meaning depends on context).

2.3 Chord Symbol Summary

The following list summarises the possible symbols available for use in describing chords.

- Sharps and flats: b = flat, # = sharp, bb = double-flat, ## = double sharp, etc.
- Major Triads: I, II, III, IV, V, VI, VII (upper case)
- Minor Triads: i, ii, iii, iv, v, vi, vii (lower case)
- Triad suffixes: add '+' to major for an augmented triad; 'o' to minor for a diminished triad (note, this is the letter 'o' not the numeral '0'); '/o' indicates a half-diminished seventh chord.
- Abbreviations: 'It', 'Ger', 'Fr', 'N', 'Cad'.
- Triad inversions: no symbol or '5/3' for root position; '6' or '6/3' for first inversion; '6/4' for second inversion (with or without the optional '/').
- Seventh inversions: '7' for root position, '6/5' for first inversion, '4/3' for second inversion and '2' for third inversion. '9', '11', and '13' are also supported in root position.
- Altered/added/omitted notes: use square brackets and Arabic numerals, e.g. '[no5]' for no fifth.

Together, these elements are sufficient to describe any standard tonal chord. Roman numerals may involve more or less complicated combinations of these symbols. Formally, the triad type is the only required element, so 'I' and 'V' (and their minor-mode equivalents) are the simplest possible syntatical entries. A maximally-complex chord description involves all of the available elements, which will be parsed in the following order:

1. Triad type (with quality indicated by case);
2. Triad suffix for diminished / augmented;
3. Accidental for raising/lowering the root (sometimes computed, in minor, from triad type);
4. Accidental for modifying a seventh and/or inversion that follows;
5. Seventh, ninth, etc. and/or inversion;
6. Accidental for modifying an ...
7. Altered/added/omitted note;
8. Relative key (in the case of applied chords)

Steps 6 and 7 also may be repeated.

See Section 3.1 for details of how the code processes this, and github.com/MarkGotham/When-in-Rome for full reference lists of all possible Roman numerals along with the pitches they entail in C major and a minor.

2.4 Form, Phrase, and Pedals

Pedal points. With pedal points, analysts face a choice between integrating the pedal into the Roman numerals – for instance, `I IV6/4 I` and `V I6/4 V` each has the same tones in the bass throughout – or to indicate that the pedal has harmonically separated from the passage and should no longer be represented in this way [9, p.113]. This format supports both notations. To notate a pedal separately, use a line of the kind:

`Pedal: G m14 b3 m19 b1`

Here a G pedal begins in measure 14 beat 3 and lasts until (but not through) measure 19 beat 1.

Large-scale formal labels. Formal sections can be identified using the prefix `Form:`. Examples include the major sections of Sonata Form ('Exposition', 'Second Theme', 'Development', 'Recapitulation' and 'Recapitulation Second Theme'); numbered variations in variations form ('Variation N'); and the large-scale divisions of a Rondo ('Rondo A', 'Rondo B'). These should be positioned on a separate line before the measure in which the section begins.

Phrase boundary. Phrase boundaries can be identified with '||'. This provides data useful for many lines of enquiry. From the strictly harmonic perspective, this can help to contextualise unusual progressions. For instance, in

```
m33 V || b2 IV
```

the oddity of an apparent V-IV progression is contextualised by the fact that it occurs across a phrase break: the music stops on V before resuming on IV.

2.5 Repeated Progressions and Variants

Repeats. For different passages with the same harmonies, the format supports the following abbreviation:

```
m3-4 = m1-2
```

This indicates that measure 3 is the same as measure 1 and measure 4 is the same as measure 2. This shorthand works for exact repeats of chords progressions, with the same chords, in the same order, in the same part of the measure. For near variants, judicious copy-and-paste is required, taking care to make the necessary changes. One of the authors (DT) has written a simple python program to renumber measures and shift chords into a new key, available on request.

Variant readings. For multiple readings of the same passages, use the 'var' tag. For instance,

```
m1 viio6
m1var1 ii
```

indicates that the chord in measure 1 is ambiguous: it may be viio6 or ii. Multiple variants may be indicated with var1, var2.

Notes. The 'Note' tag affords an opportunity to record any other noteworthy elements, such as a pattern not evident from the Roman numerals alone. Include notes on separate line, before the moment of interest. This subsidiary feature is for the analyst's reference only and not processed as part of the harmonic analysis.

3. CODE AND CORPUS

Apart from the .rntxt standard specification, this paper also presents a code library in `music21` for handling Roman numeral analyses, as well as a set of initial corpora.

3.1 Code

The code focusses on translation routines for the .rntxt representation defined above, the related Clercq-Temperley standard defined by Trevor de Clercq and David Temperley for rock harmony ('.tdc' extension), and the DCMLab's tabular representation format ('.tsv' extension). [1]

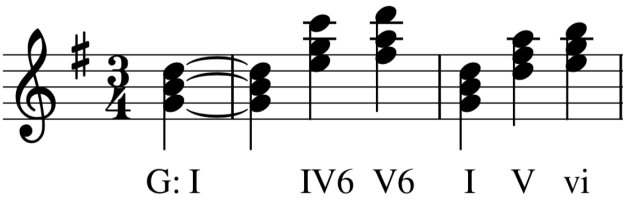


Figure 2. An example of `music21`'s default musical-notation rendering, with chords in close position and the numeral itself included as a 'lyric'. This example is from the start of BWV269, corresponding to Figure 1.

Like the .rntxt representation format, the code also provides a neutral conduit for processing Roman numeral analyses in various ways: converting between representation formats, rendering the analyses in musical notation, and engaging in computational analysis using the wider `music21` code base. Again, that neutrality means supporting any in-principle approach to Roman numeral analysis as long as the representation meets the syntactical criteria of the format.

Exceptions are raised in the case of divergence from the syntax, and the parsers are fully integrated into `music21`, so both `corpus.parse()` and `converter.parse()` read directly from the file types listed (.rntxt, .tdc, and .tsv). Files with other extensions can be parsed by passing in a format parameter: `.parse('file.txt', format='romantext')`.

All features of RomanText described in this paper are supported by the most recent release of music21 (v.5.7), except [addX] tones and settings for alternative parsing of vi/VI and vii/VII included in v.6.0alpha, and variant readings which are not currently supported.

From these RomanText analyses, the code creates a `music21` Score with a single Part object containing all the RomanNumeral objects within the relevant measures. Figure 2 shows an example of how `music21` renders Roman numerals: unless specified otherwise, the tonic is placed in octave 4 (the octave above middle C inclusive), the root (or imagined root if in inversion) is placed above that tonic, and the real bass of each chord is placed in the octave above that pitch. Where the corresponding score exists in an encoded format, this analysis part can be inserted as an additional 'part' in order to view the relationship between score and analysis in musical notation. Future work may see an option for matching the analysis up with the original piece in order to render chords in their original spacing.

The `RomanNumeral()` class extends the `Chord` class, inheriting mutually useful variables like `.root`,

[1] For more information on the deClercq-Temperley format, see the scholarly reports in [3] and [16]. For the DCML lab's standard, see [12] and the code base at `github.com/DCMLab/ABC/` For music21's provision, see the code base at: `github.com/cuthbertLab/music21/tree/master/music21/romanText` or for documentation: `web.mit.edu/music21/doc/moduleReference/moduleRoman.html` for the code and Chapter 23 of the User's Guide: `web.mit.edu/music21/doc/usersGuide/`

and introducing new read/write attributes. Thus anything one can do to or with a `Chord` is also possible with a `RomanNumeral`. A Roman numeral pairs a `.figure` (such as `I6`) with a `.key` (such as `C`), but the class supports a range of additional attributes, some settable, others read-only.

Additional methods support the creation of RomanNumeral object ab initio, such as the `.romanNumeralFromChord()` method which processes any chord (which in turn can be made from a list of pitches) into a `RomanNumeral` object when paired with a specific key.

As discussed in Section 2, the RomanText format supports additional elements that are more textual than harmonic: 'large scale formal labels', 'pedals', and 'notes'. music21 processes these as text with a dedicated subclass of TextExpression.

3.2 Corpus

Finally, we present a set of example corpora approximately representing the start, middle and end of common-practice tonality as follows (with the total analyses in brackets):

1. Monteverdi madrigals, Books 3–5 (48 works);
2. Bach chorales (a sample, 20 analyses);
3. Preludes from the first book of Bach's Well Tempered Clavier (complete, 24 preludes);
4. Beethoven string quartets, converted from the DCMLab's ABC corpus and with manual error-correction (complete, 16 quartets, 70 movements);
5. 19th-century French and German songs from the 'Scores of Scores' corpus [6], (sample, 50 songs).

See github.com/MarkGotham/When-in-Rome for directions to these corpora: the first two are included in the latest version of music21; more will be added after error checking. We offer them in order to illustrate how the format works in a range of repertoire contexts, and to provide an initial dataset for experimenting with the format. An additional corpus of approximately 1,000 scores, ranging from Dufay to Brahms, will be released within a year.

As we have been at pains to point out, the .rntxt format accommodates any kind of Roman numeral analysis, as long it is adopts the basic syntax as outlined above. That said, any particular corpus will have to make 'policy' decisions about its approach, if it is to be consistent. In these corpora, we have elected to prefer:

1. harmony changes on metrically strong positions and at regular intervals;
2. to analyse similar material in similar ways;
3. to identify as 'harmonic' notes that do not belong to any common species of non-harmonic tone (e.g. notes that are both leapt-to and leapt-from); and
4. harmonic analyses that are more consistent with standard harmonic theory.

No.1 and No.2 are intuitive enough, though No.2's reliance on 'similar' leaves ample room for ambiguity; furthermore there are cases where harmonic considerations lead to different analyses of parallel passages. Rule No. 3 requires harmonic analyses to conform to the precepts of traditional contrapuntal theory, which is mostly appropriate for our chosen repertoires (though some allowances need to be made in the case of Monteverdi).

No.4 is arguably more interesting. The preference may appear circular, but it is not: standard harmonic theory asserts that the majority of tonal chord progressions can be understood as conforming to a small number of harmonic and contrapuntal patterns; it does not assert that all passages are unambiguous, nor that composers tried to avoid analytical ambiguity when composing. Thus analysts commonly rely on their harmonic expectations when identifying chords. For instance, given a D-F dyad between I and I^6 in a common-practice C major context, preference No.4 helps lead us to identify the most promising candidates for completion as ii, V^4_3, and $viio^6$ and to choose either of the latter two, since the progressions I $viio^6$ I6 and I V^4_3 I^6 are significantly more common than I ii I^6. (The former is more common in Bach; the latter in Beethoven, so detailed stylistic information is needed in this case.) The sense of 'commonness', furthermore, can be justified by looking only at those cases where a complete triad intervenes between I and I^6. In this way, we use our sense of what happens in the non-ambiguous cases to guide our interpretation of the ambiguous ones.

A more thorough-going discussion of the philosophical options available to Roman numeral analysts will be published by author DT soon.

4. SUMMARY AND OUTLOOK

This paper presents a specification for Roman numeral analysis, code for working with those analyses, and a set of example corpora. The format thus supports a range of data-driven approaches to harmonic analysis, as well as other applications including pedagogical (e.g. visualisation and the selection of pertinent teaching examples) to compositional (setting harmonic constraints for stochastic composition). We hope that these offerings will contribute positively to ISMIR's 20th anniversary call for 'reusable' material to 'build up consistent knowledge across the community.'

5. ACKNOWLEDGEMENTS

The development of music21's RomanText processing module was generously supported by a grant from the Seaver Institute. Author MG's work was supported by a grant from Cornell University's Active Learning Initiative.

6. REFERENCES

[1] Tsung-Ping Chen and Li Su. Functional harmony recognition of symbolic music data with multi-task recurrent neural networks. In Emilia Gómez, Xiao Hu, Eric Humphrey, and Emmanouil Benetos, editors, *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, pages 90–97, 2018.

[2] Michael Scott Cuthbert and Christopher Ariza. Music21: A toolkit for computer-aided musicology and symbolic music data. In *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, Utrecht, Netherlands, August 9-13, 2010*, pages 637–642, 2010.

[3] Trevor de Clercq and David Temperley. A corpus analysis of rock harmony. *Popular Music*, 30(1):47–70, 001 2011.

[4] Johanna Devaney, Claire Arthur, Nathaniel Condit-Schultz, and Kirsten Nisula. Theme and Variation Encodings with Roman Numerals (TAVERN): A new data set for symbolic music analysis. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, pages 728–734, 2015.

[5] Robert O Gjerdingen. *Music in the Galant Style*. Oxford University Press, Oxford ; New York, 2007.

[6] Mark Gotham, Peter Jonas, Bruno Bower, William Bosworth, Daniel Rootham, and Leigh VanHandel. Scores of scores: An openscore project to encode and share sheet music. In *Proceedings of the 5th International Conference on Digital Libraries for Musicology*, DLfM '18, pages 87–95, New York, NY, USA, 2018. ACM.

[7] Christopher Harte. *Towards Automatic Extraction of Harmony Information from Music Signals*. PhD thesis, Department of Electronic Engineering, Queen Mary, University of London, London, 2010.

[8] Christopher Harte, Mark B. Sandler, Samer A. Abdallah, and Emilia Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *ISMIR 2005, 6th International Conference on Music Information Retrieval, London, UK, 11-15 September 2005, Proceedings*, pages 66–71, 2005.

[9] Paul Hindemith. *The Craft of Musical Composition*, volume 1. tr. Arthur Mendel, Schott, London, 1945.

[10] Ludwig Holtmeier. Heinichen, Rameau, and the Italian thoroughbass tradition: Concepts of tonality and chord in the rule of the octave. *Journal of Music Theory*, 51(1):5–49, 2007.

[11] Hitomi Kaneko, Daisuke Kawakami, and Shigeki Sagayama. Functional harmony annotation database for statistical music analysis. In Kaneto, editor, *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, Utrecht, Netherlands, August 9-13, 2010*, pages N/A – Late–breaking demo. International Society for Music Information Retrieval, 2010.

[12] Markus Neuwirth, Daniel Harasim, Fabian C. Moss, and Martin Rohrmeier. The Annotated Beethoven Corpus (ABC): A Dataset of Harmonic Analyses of All Beethoven String Quartets. *Frontiers in Digital Humanities*, 5(16), 2018.

[13] Hugo Riemann. *Vereinfachte Harmonielehre; oder, Die Lehre von den tonalen Funktionen der Akkorde [Harmony Simplified: Or, the Theory of the Tonal Functions of Chords]*. Augener, London, 1893.

[14] John Rothgeb. Strict counterpoint and tonal theory. *Journal of Music Theory*, 19(2):260–284, 1975.

[15] John Rothgeb. Re: Eytan Agmon on functional theory. *Music Theory Online*, 2(1), 1996.

[16] David Temperley and Trevor de Clercq. Statistical analysis of harmony and melody in rock music. *Journal of New Music Research*, 42(3):187–204, 2013.

[17] Dmitri Tymoczko. Progressions fondamentales, functions, degrés, une grammaire de l'harmonie tonale élémentaire. *Musurgia*, 10:35–64, 2003.

[18] Dmitri Tymoczko. *Geometry of music: harmony and counterpoint in the extended common practice*. Oxford University Press, New York; Oxford, 2011.

[19] G. Weber. *Versuch einer geordneten Theorie der Tonsetzkunst*. Schott, Mainz, 1832.

20 YEARS OF PLAYLISTS: A STATISTICAL ANALYSIS ON POPULARITY AND DIVERSITY

Lorenzo Porcaro[1], Emilia Gomez[1,2]
[1]Music Technology Group, Universitat Pompeu Fabra, Barcelona
[2]Joint Research Centre, European Commission, Seville
{lorenzo.porcaro,emilia.gomez}@upf.edu

ABSTRACT

Grouping songs together, according to music preferences, mood or other characteristics, is an activity which reflects personal listening behaviours and tastes. In the last two decades, due to the increasing size of music catalogue accessible and to improvements of recommendation algorithms, people have been exposed to new ways for creating playlists. In this work, through the statistical analysis of more than 400K playlists from four datasets, created in different temporal and technological contexts, we aim to understand if it is possible to extract information about the evolution of humans strategies for playlist creation. We focus our analysis on two driving concepts of the Music Information Retrieval literature: popularity and diversity.

1. INTRODUCTION

The advent of the streaming era has transformed the role of music playlists, which have become central to the listening experience. The current interest by both academia and private company is illustrated by the ACM RecSys Challenge 2018 for Automatic Music Playlist Continuation, where almost 2K people registered for participating [4]. However, the interest in algorithmic-enhanced methods for playlist generation started much before, when large catalogues of digital music became available, around the beginning of this century [1, 13, 17].

In the last 20 years, Music Information Retrieval (MIR) literature has addressed several aspects of playlists, both from a user and an algorithmic perspective [8]. In this work, we focus on the analysis at scale of large playlist datasets in order to understand how humans create a playlist in different contexts. The main hypothesis of our work is that technological innovations occurring in the last two decades have affected how people are experiencing music, and thus how music pieces are grouped together.

We center our attention on two main facets considered important in the design of playlist generation systems, according to the literature. On one hand, we address the characterization of playlists in terms of popularity, which has been a focus of several studies and related to the so-called long-tail effect [3]. On the other hand, we address the semantic diversity of a playlist, considered as the contrary to the semantic similarity concept. The trade-off between similarity and diversity has been already object of analysis [6, 7], so following a similar direction we study the diversity of semantic information in playlists.

In detail, we measure the popularity of a playlist, starting from the popularity of its component tracks. The advantage of this measure relies in the possibility to compute it without looking at the content. Furthermore, using tags retrieved from *Last.fm* [1] , we define a playlist diversity index based on the semantic distance between its tracks. The distance is computed using tag embeddings, a compact and meaningful representation of user-generated annotations.

This work has three main contributions. First, we propose and implement two distinct indexes for playlist dataset characterization: one related to the concept of popularity and another one to the concept of diversity. Second, we apply, study and discuss the statistical distribution of these measures to four datasets, containing more than 400K playlists created in the last 20 years. Third, we release the data and software used for the analysis in order to foster reproducible research and future work in the topic.

The paper is structured as follows. Section 2 provides an overview of previous works related to the analysis of the playlist creation processes. We then propose an analysis methodology in Section 3, which includes a description of the model, the considered datasets and the proposed statistical measures. Section 4 provides the obtained results, which are discussed in Section 5.

2. RELATED WORK

A playlist is usually described in a very broad way as "*an ordered sequence of songs meant to be listened to as a group*", a noteworthy definition in the literature [9]. Several approaches have been proposed to build computational models of the mechanisms behind playlist creation such as Combinatorial Pattern Generation [13], Gaussian Process Regression [17], Markov Chains [5], or Hypergraph Random Walks [12] among the others. Recent studies for analyzing playlists proposed context-aware algorithms, which

Attribution: Lorenzo Porcaro[1], Emilia Gomez[1,2]. "20 years of playlists: a statistical analysis on popularity and diversity", 20th International Society for Music Information Retrieval Conference, Delft, The Netherlands, 2019.

[1] https://www.last.fm

	AOTM	CORN	SPOT	DEEZ
Oldest playlist	1998	2010	2012	2013
Newest playlist	2011	2011	2015	2018
# playlists	100K	15K	175K	82K
Max length	60	75K	47K	400
# playlists (w/o outliers)	97K	15K	155K	74K
Max length (w/o outliers)	33	361	109	65
Avg length	19	131	27	17
# tracks	972K	75K	2,789K	277K

Table 1. Summary of the datasets.

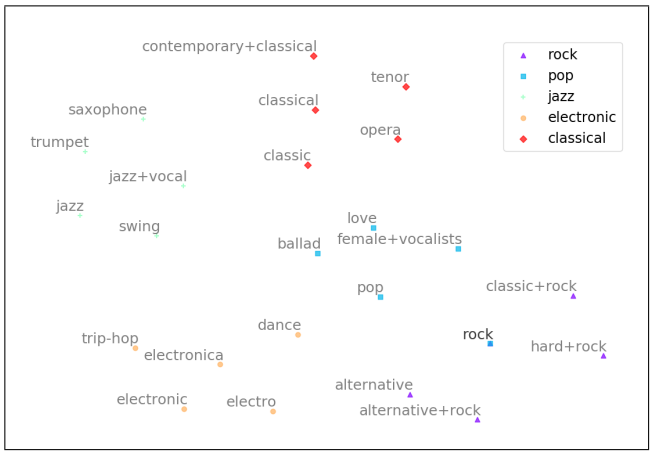


Figure 1. Top 5 similar tags found for "*rock*", "*pop*", "*jazz*", "*electronic*", "*classical*", using the tag-embeddings computed from DEEZ corpora. Distance is calculated with approximate nearest neighbor algorithm, using euclidean distance of normalized vectors, plotted using t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm.

takes into account characteristics such as the playlist title [16], and also sequence-aware algorithms, which analyze the order of songs in a playlist [20]. For an extensive review of manual, automatic and assisted techniques for playlist creation, we refer to [8].

Understanding how people create playlist is fundamental for creating computational models capable of emulating this human generation process (automatic playlist generation) [5], or predicting the most likely song to add in a given playlist (automatic playlist continuation) [4]. In addition, the intrinsic value that individuals give to a set of songs cannot be always fully explained by the analysis of the acoustic features, as tempo or tonality, emotional state or contextual information [6], and the sentence "*Making a playlist is more of an art than a science*" partly summarizes this hindrance [7]. Finally, interactive tools for supporting users during the decision-making process of playlist creation have been shown effective, but at the same time affecting human decisions. Indeed, in [11] the authors show how these kinds of tools can bias humans towards adding

AOTM	CORN	SPOT	DEEZ
Rock	Rock	Rock	Rock
Indie	Altern.	Pop	Pop
Altern.	Pop	Indie	Fem. Voc.
Pop	Jazz	Altern.	Altern.
Fem. Voc.	Fem. Voc.	Electr.	Indie
Altern. Rock	Indie	Fem. Voc.	Hip Hop
Class. Rock	Class. Rock	Hip Hop	Electr.
Indie Rock	Soul	Jazz	French

Table 2. Top tags used within each dataset (Fem. Voc.=Female Vocalist; Altern.=Alternative; Class. Rock = Classic Rock; Electr.=Electronic).

tracks more popular or more recent, in comparison to what they would independently add to the playlist. It can be considered as a consequence of the difficulty of creating models which effectively reflect human behaviors.

3. METHODOLOGY

3.1 Dataset

This study considers a total of 409K playlists (2.3M songs) from four different playlist datasets (see summary in Table 1), three of them already proposed in the literature:

1. Art of the Mix [2] (AOTM): Playlists submitted by users to the Art of the Mix website [2].
2. Yes.com [5] (CORN): Playlists from radio stations in the United States.
3. Spotify [16] (SPOT): Playlists from Twitter's users tweeting via Spotify.
4. Deezer (DEEZ): Playlist from Deezer's users, crawled in-house.

The datasets were considered because of the heterogeneity of their nature. Indeed, they have been created using playlists from different periods, with different usage and purpose. CORN [3] provides playlists from radio stations in the United States, without restrictions on musical genres. This is the only dataset where playlists are not generated by users. AOTM [4] is the dataset containing the oldest playlists, covering a 13 years period from 1998 to 2011. It is formed by playlists submitted by users to Art of the Mix, a website where a community of playlist passionates share their creations. SPOT [5] has been composed tracking Spotify's users active in Twitter between 2012 and 2015. Similarly, DEEZ has been created using the Deezer API, selecting users playlists created between 2013 and 2018.

There are two main differences between AOTM and the other two user-generated datasets. First, SPOT and DEEZ

[2] http://www.artofthemix.org
[3] http://www.cs.cornell.edu/~shuochen/lme/data_page.html
[4] https://bmcfee.github.io/data/aotm2011.html
[5] http://dbis-nowplaying.uibk.ac.at/#playlists

Artist	Track	Tags
The Blacktop Cadence	Off track	('punk', 100), ('indie rock', 100)
The Cure	Maybe someday	('post-punk', 100), ('new wave', 92)
The Blacktop Cadence	I don't do well in social situations	('punk', 100),('indie rock', 100)
Jefferson Airplane	Today	('classic rock', 100), ('Psychedelic Rock', 98)
Husker Du	Something I learned today	('punk', 100), ('hardcore', 49)
Superchunk	Punch me harder	('indie', 100), ('college rock', 50)
Willie Bobo	Fried neck bones and some home fries	('sexy', 100), ('downtempo', 100)
Wu-tang Clan	Clan in da front	('Hip-Hop', 100), ('rap', 81)

Table 3. Example of tracks and relative tags of a playlist with low diversity ($pDI = 0.12$, *top*), and a playlist with high diversity ($pDI = 0.98$, *bottom*)

playlist creation process is embedded on particular streaming platform, while this does not apply to AOTM. Second, playlists in AOTM have been created before streaming services became intensively used worldwide [6], while DEEZ and SPOT dataset are representative of a period in which streaming technologies were already consolidated.

As pre-processing step, we filtered out playlists with less than 4 unique tracks. In addition, we excluded extremely long playlists by computing, for each dataset separately, the interquartile range (IQR) of the playlist lengths and excluding all playlists that are longer than $Q_3 + 1.5\,IQR$, where Q_3 is the 3rd quartile. Table 1 shows a summary of the dataset characteristics.

3.2 Playlist Popularity Analysis

We address the characterization of popularity by defining a popularity index for both tracks and playlists based on a set of metrics proposed in the literature.

We estimate the *Track Popularity Index* (tPI) as the track frequency within a dataset, i.e. the number of playlists in a dataset in which it occurs, as proposed in [20]. This index is normalized between 0 and 1, using min-max normalization. In order to understand how track popularity is distributed, we uniformly split each dataset into 10 different groups, according to tPI. The first one contains tracks with $tPI \in [0, 0.1)$, the second one with $tPI \in [0.1, 0.2)$, etc. We then analyze the statistical distribution of track popularity per group, using two qualitative measures proposed in the literature: *Shannon* index [18] and *Simpson* index [19]. For both indexes, 0 indicates that there is no variation in terms of popularity, while 1 indicate that popularity varies significantly within the dataset.

In addition, we define the *Playlist Popularity Index* (pPI) for a playlist p as the average of tPI for the playlist tracks t_i

$$pPI(p) = \frac{1}{len(p)} \sum_{i=1}^{len(p)} tPI(t_i) \qquad (1)$$

We compute pPI for each playlist and we then compute the $Gini$ coefficient [10] to estimate the degree of imbalance of the playlist popularity distribution for each dataset, i.e. we obtain a measure of the statistical dispersion of playlist popularity. $Gini$ coefficient is comprised between 0 and 1, where 0 express the maximum balance, which means that pPI is almost equally distributed between playlists in the dataset, while 1 represents an unbalanced situation, which means that few playlists have high pPI, while several playlists have low pPI.

In the case of playlist popularity, the choice of using *Gini* coefficient is motivated by the idea of having a value representing the influence of every playlist on the overall dataset distribution. Differently, when tracks are grouped together according to their popularity, thanks to *Shannon* and *Simpson* indexes we have an estimation of how tracks are distributed within groups.

3.3 Playlist Semantic Diversity Analysis

In order to characterize the diversity of tracks on a given playlist, we consider a semantic distance measure based on user-generated tags. For every track, we queried *Last.fm* website to retrieve its top 5 related tags. We then proceed as follows. First, we create a tag-vector representation to estimate tag distances. Second, we use a linear combination of tags-embedding distances, weighted with a tag popularity count, to obtain the distance between every two tracks of a playlist. Third, we average the distances between tracks to yield a final diversity estimation for a playlist, according to the retrieved tags.

Finally, we obtain two metrics: 1) a distance between tracks, based on tag-similarity; 2) a playlist diversity index, based on the distance variations between tracks. We provide more details on the process in the next sections.

3.3.1 Tag-embeddings

Recent developments of NLP techniques have shown how particular types of words vector representation can bring with them valuable semantic information. In our study, we select the *GloVe* [15] learning algorithm [7], an architecture that exploits the ratio of word-word co-occurrences probabilities within a corpora for generating tag embeddings.

[6] `https://www.ifpi.org/downloads/GMR2016.pdf`

[7] `https://nlp.stanford.edu/projects/glove`

We choose this representation because of its compact form, the low computation cost needed for training the model with new corpora, and the facility to compute a distance metric between embeddings. For training the model, we combine the retrieved tags from *Last.fm* to create a corpus of track tags, and we use it to generate a vector representation for each tag.

In Table 2, we report the most frequent tags for the four datasets. Most of them are shared between datasets, so we study the few not shared tags to better understand their peculiarities. As an example, *"French"* only appear in DEEZ, having Deezer been founded in France. Furthermore, we observe how *"Electr."* and *"Hip Hop"* tags only appear in SPOT and DEEZ. The rise of commercial music in these these two genres in the last decade may be reflected in their extended presence.

Figure 1 shows an example of similar tags, where the distance has been computed using the trained tag-embeddings. *"rock"* and *"pop"* clusters are quite near, as *"jazz"* and *"classical"*. Within *"electronic"* similar tags, *"dance"* is the nearest one to *"pop"*. Within the *"jazz"* cluster, there are *"trumpet"* and *"saxophone"*, two instruments often related to this genre. These are some examples of observations that can be derived, and which reflect semantic information contained in the computed vector representation.

3.3.2 Playlist Track-Tag diversity index

We define a track *tr* as linear combination of its T weighted tags:

$$tr = \sum_{i=1}^{T} w_i \, tag_i \tag{2}$$

where in our settings $T = 5$. Basing on (2), we define a distance measure between tracks, named *Track-Tag distance* (d_{TT}), as follow

$$d_{TT}(tr^{(1)}, tr^{(2)}) = \frac{1}{T} \sum_{i=1}^{T} W_i^{(1,2)} \, d_{tag}(tag_i^{(1)}, tag_i^{(2)}) \tag{3}$$

where the weight term is

$$W_i^{(1,2)} = \frac{w_i^{(1)} + w_i^{(2)}}{2\, max(w_i^{(1)}, w_i^{(2)})} \tag{4}$$

and the distance between two tags is defined as

$$d_{tag}(tag_1, tag_2) = \sqrt{2(1 - cos(tag_1, tag_2))} \tag{5}$$

$cos(tag_1, tag_2)$ represents the cosine similarity between tag-embeddings. The computation of the tag distances has been carried out with the Annoy Python library [8], which makes use of the approximate nearest neighbor technique for an efficient computation of the euclidean distance of normalized vectors. In the d_{TT} formula, $W_i = 1$ if $w_i^{(1)} = w_i^{(2)}$, so weights do not impact the distance. Otherwise, $W_i \in (0.5, 1)$, hence final distance decreases when multiplying the weight term with the tag distance. In detail, d_{TT} is near to 0 when two tracks have a high similarity, while it is around 1 when they are very different, according to their combination of user-generated tags.

For understanding a playlist diversity in terms of semantic annotations, we first compute the d_{TT} distance for every combination of two tracks in the playlist. Summing the distances and dividing by the number of total possible combinations, we obtain the *Playlist Track-Tag diversity index* (pDI):

$$pDI(playlist) = \frac{2}{M(M-1)} \sum_{i,j} d_{TT}(tr_i, tr_j),$$
$$\forall tr_i, tr_j \in playlist, j > i$$

where M is the length of the playlist. pDI is near to 0 when there is a low diversity between tracks within the playlist, and almost 1 when tracks are extremely diverse, according to the d_{TT} distance. Table 3 provides two examples of playlists with different pDI values. The playlist with low pDI is formed by tracks mainly tagged as *"punk"*, *"rock"* or *"indie"*, while the playlist with high pDI has tracks with tags more diverse, passing from *"punk"* to *"downtempo"*, to *"hip hop"*.

4. RESULTS

4.1 Playlist Popularity Analysis

Results of the popularity analysis are shown in Table 4. We first observe the great differences between radio playlists from the CORN dataset and user-generated playlists from the other datasets. Tracks' popularity tPI in CORN varies significantly more than in the other cases, according to the *Shannon* and *Simpson* indexes, and the percentage of tracks with $tPI \in [0.0, 0.1)$ is smaller, indicating a large presence of popular tracks within the dataset. The mean of playlist popularity pPI is not extremely high, but it is more balanced than for DEEZ, SPOT and AOTM according to the *Gini* coefficient. Results can be interpreted as a consequence of the nature of radio playlists. Indeed, tracks rotation in radios is often constrained by commercial policies, because artists, or someone in their behalf, have to pay for broadcasting their tracks. This clearly makes difficult for artists with few resources to be on air in a radio. This phenomena is reflected in having few tracks from the long-tail, i.e. less popular, inserted in radio playlists. The balanced playlist popularity level also derives from the policy of alternating popular tracks with less know ones [9] .

Regarding user-generated playlists, we observe how track popularity in DEEZ and SPOT are quite similarly distributed. On the contrary, in AOTM the presence of 99.99% of track with $tPI \in [0.0, 0.1)$ influences both the *Shannon* index, the *Simpson* index and the *Gini* coefficient, creating an unbalanced situation with no diversity in terms of track popularity. However, results of playlist popularity analysis give similar values among the three user-

[8] https://github.com/spotify/annoy

[9] https://www.digitalmusicnews.com/2015/02/19/five-things-internet-radio-steal-broadcast-radio

	AOTM	CORN	SPOT	DEEZ
Top tPI	4368	1746	1270	2523
Top tPI (normalized by dataset size)	0.045	0.112	0.008	0.034
Track with $tPI \in [0.0, 0.1)$ (%)	99.99	96.09	99.85	99.74
tPI Shannon index	$3.1 \cdot E^{-5}$	0.212	0.013	0.020
tPI Simpson index	$4.3 \cdot E^{-6}$	0.076	0.003	0.005
Avg pPI (normalized by Avg playlist length)	1.19	2.01	1.76	9.64
pPI Gini coefficient	0.66	0.39	0.67	0.55

Table 4. Summary of playlist popularity analysis results.

	AOTM		CORN		SPOT		DEEZ	
	Original	Random	Original	Random	Original	Random	Original	Random
Mean	0.68	0.72	0.84	0.85	0.58	0.68	0.63	0.82
Std	0.12	0.10	0.09	0.03	0.19	0.11	0.19	0.11
Max	0.99	0.98	1.05	0.97	1.13	0.99	1.14	1.09
Min	$1.3 \cdot E^{-5}$	0.21	0.33	0.69	$7.3 \cdot E^{-7}$	0.22	$1.9 \cdot E^{-5}$	0.27
Gini	0.10	0.08	0.06	0.02	0.19	0.09	0.17	0.08
QCD	0.11	0.10	0.07	0.03	0.21	0.12	0.22	0.09

Table 5. Playlist Track-Tag diversity index (pDI) descriptive statistics. QCD indicates the quartile coefficient of dispersion, computed as $QCD = (Q3 - Q1)/(Q3 + Q1)$, where Q_1 and Q_3 are the first and third quartiles.

generated datasets, where only the DEEZ stands out for having a high average value of pPI.

In general, the popularity of a playlist, intended as average of the frequency of its tracks within a dataset, can be influenced by several factors. As example, AOTM dataset has been created with playlists from 1998 to 2011, when music was consumed by means of different services than the ones which lead the market today. We suppose that playlists in AOTM do not often come from the interaction with a large music catalogues, or with algorithms for facilitating music search and discovery for playlist generation, and this can be related to a major presence of less popular tracks. Furthermore, current streaming services employ several tools to facilitate playlist sharing, to make this a collaborative process, and to incorporate tracks of a playlist into new playlists [14]. In terms of popularity, the possibility to share a playlist can have a positive impact, increasing the accessibility to much more content and then reducing the number of less popular tracks.

4.2 Playlist Semantic Diversity Analysis

We faced two limitations when retrieving tags from *Last.fm*: 1) we did not find tags for all queried tracks; 2) for part of the tracks, the associated tags were small, less than five. As a consequence, we follow a conservative approach when computing the Playlist Track-Tag diversity index: 1) we only consider playlists for which all tracks have associated tags (*complete information*); 2) tracks are only compared with other tracks with the same number of tags (*balanced information*).

After obtaining the semantic index for each playlist, we compute descriptive statistics for understanding how the computed descriptor characterizes these datasets. In addition, for every case we also extract the same statistics on playlists of average size, created with random tracks from the original playlists. In Table 5, we observe part of the differences between datasets.

As in the previous sections, the analysis on CORN radio playlists provides values of different order of magnitude than user-generated playlists. Indeed, the mean and the minimum value of the diversity index pDI are higher for this dataset. This can be related to the fact that radio playlists are rarely composed by tracks from the same artist, as it is the case for low diversity playlists according to our analysis. Moreover, we observe that CORN playlists are more balanced in terms of tag-similarity, as they are in terms of track popularity, as represented by a low *Gini* coefficient and low quartile coefficient of dispersion.

In order to better understand how the diversity index represents playlists' characteristics, we have carried out a qualitative analysis of the 10% of playlists with higher, and 10% with lower diversity. For every groups of playlists, we compute the following values, reported in Table 6: 1) average of Playlist Track-Tag index (Avg pDI); 2) average of unique tags for playlist (Avg tag count); 3) number of playlists with at least one tag in common between all the tracks (Common tags); 4) average of unique tags over tracks (Avg Tag/Tracks); 5) average of unique artists for playlist (Avg artist count); 6) playlist with tracks from the same artist (Single-Artist); 7) average of unique tracks for playlist (Avg track count) ; 8) average of unique tracks over artist (Avg Tracks/Artist).

From the analysis of these values, we confirm previous observations on the pronounced difference between CORN and the other datasets. Looking at the percentage difference of each parameter, we see how CORN playlists span

	AOTM			CORN			SPOT			DEEZ		
	Low	High	PD	Low	High	PD	Low	High	PD	Low	High	PD
Avg pDI	0.43	0.85	64	0.65	0.98	40	0.19	0.87	128	0.28	0.93	108
Avg Tag Count	20	44	75	25	25	0	15	30	67	11	22	67
Common Tags (%)	34	0	34	0	0	0	23	1	22	44	1	43
Avg Tag/Tracks	1	3	100	3	4	29	1	3	100	2	3	40
Avg Artist Count	6	15	86	9	7	25	2	7	111	2	5	86
Single-Artist (%)	55	2	53	0	0	0	83	22	61	69	7	62
Avg Track Count	17	16	6	10	7	35	12	11	9	8	7	13
Avg Tracks/Artist	11	1	167	1	1	0	11	4	93	6	2	100

Table 6. Qualitative analysis results of low/high 10% playlists, ranked by their pDI. Column "PD" reports the percentage difference (in %) between low and high cases values, calculated as $PD(a,b) = 100\ \frac{|a-b|}{(a+b)/2}$

a small range of diversity, in comparison to user-generated playlist datasets. SPOT and DEEZ present more variation in terms of diversity, reflecting the values obtained for the quartile coefficient of dispersion, presented in Table 5.

In general, these parameters are coherent with the analysis carried out before: playlists with a low pDI, hence with less diversity, present in average a smaller number of unique tags, more tags in common between tracks, few artists for playlist and a higher percentage of single artist playlists. Playlists with a high index present the inverse characteristics.

5. CONCLUSIONS

We have presented a statistical analysis of more than 400K playlists (2.3M songs) from four different datasets, three composed by user-generated playlists, while one, CORN, composed by radio playlists. Two of the user-generated datasets, SPOT and DEEZ, have playlists created between 2012 and 2018, while AOTM playlists between 1998 and 2011. We develop our analysis using descriptive statistics, and in addition we make use of indexes from the information retrieval literature for evaluating the distribution of the analyzed features within the sets. In particular, we focused on two aspects: popularity and diversity.

From the proposed metrics, we observe how differences between datasets emerge, reflecting the distinct context in which playlists have been created. On one side, radio playlists analysis shows clear different results from the ones obtained from user-generated playlist. On the user-generated side, we observe how the study of AOTM playlists reveals different characteristics than for SPOT and DEEZ datasets. Behind this fact, we identify as possible cause the change of music listening behaviours, shifting from the idea of personal music repositories in the beginning of the digital era, to the dominance of streaming services of today. We hypothesize that this paradigm change has also impacted the way users create playlists, and our results partially reflect this shift.

In our analysis, we have found a more balanced situation in SPOT and DEEZ datasets in terms of popularity, although they contain playlists with a high level of diversity in terms of semantic tags. Even if different explanations can be at the root of the different values, e.g. the larger song search space of streaming services, the lower cost to create and share a new playlist online, or the recommendation algorithms that support playlist creation, we cannot identify a specific cause with our analysis.

The proposed methodology can be applied to characterize playlists in terms of popularity and semantic diversity, allowing the comparative analysis of human-generated and algorithm-generated playlists in different contexts such as historical periods, platforms and musical genres. We find extremely valuable to compare different playlist datasets, as it allows to understand how changes in the listening experience are affecting playlist creation strategies.

We hypothesize that if we extend this analysis to a larger number of datasets, we would achieve a better understanding of these changes. For instance, one of the limitations of the considered datasets is that they provide a Western-centric view. Adding playlist creators country information could enrich our study. Similarly, a yearly-based temporal analysis would help to better understand temporal variations. Moreover, adding other content- and context-based features from playlists can help to explore factors that are hidden in the presented analysis. Finally, it has already been shown that considering the tracks ordering is helpful for extracting playlist characteristics [20], so we plan to include this information in future research.

To facilitate the reproducibility and transparency of our study, the data and the software used are made publicly available [10] .

6. ACKNOWLEDGMENTS

This work is partially supported by the European Commission under the TROMPA project (H2020 770376).

7. REFERENCES

[1] J.-J. Aucouturier, and F. Pachet. "Scaling up music playlist generation", *IEEE International Conference on Multimedia and Expo* , pp. 105–108, 2002.

[10] https://github.com/MTG/playlists-stat-analysis

[2] A. Berenzweig, B. Logan, D.P.W. Ellis, and B. Whitman. "A Large-Scale Evaluation of Acoustic and Subjective Music-Similarity Measures", *Computer Music Journal*, Vol. 28, No. 2, pp. 63–76, 2004.

[3] O. Celma. *Music Recommendation and Discovery - The Long Tail, Long Fail, and Long Play in the Digital Music Space*, Springer, 2010.

[4] C.-W. Chen, P. Lamere, M. Schedl, and H. Zamani. "Recsys challenge 2018: automatic music playlist continuation", *Proc. of the 12th ACM Conference on Recommender Systems*, pp. 527–528, 2018.

[5] S. Chen, J.L. Moore, D. Turnbull, and T. Joachims. "Playlist prediction via metric embedding", *Proc. of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '12*, 2012.

[6] K. Choi, G. Fazekas, and M. Sandler . "Understanding Music Playlists", *Machine Learning for Music Discovery Workshop at the 32nd International Conference on Machine Learning*, 2015.

[7] S. Cunningham, D. Bainbridge and, A. Falconer. "'More of an art than a science': Supporting the creation of playlists and mixes", *Proc. of the 7th International Conference on Music Information Retrieval*, pp. 240–245, 2006.

[8] R. Dias, D. Gonçalves, and M.J.Fonseca. "From manual to assisted playlist creation: a survey", *Multimedia Tools and Applications*, Vol. 76, No. 12, pp. 14375–14403, 2017.

[9] B. Fields, P. Lamere, and N. Hornby. "Finding a path through the jukebox: the playlist tutorial", *Proc. of the 11th International Society for Music Information Retrieval Conference*, 2010.

[10] C. Gini. " Concentration and dependency ratios (in Italian)", *English translation in Rivista di Politica Economica*, Vol. 87, No. 1997, pp. 769–789, 1909.

[11] I. Kamehkhosh, Di. Jannach, and G. Bonnin. "How automated recommendations affect the playlist creation behavior of users", *CEUR Workshop Proc.*, 2018.

[12] B. McFee, and G. Lanckriet. "Hypergraph models of playlist dialects", *Proc. of 13th International Society for Music Information Retrieval Conference*, pp. 343–348, 2012.

[13] F. Pachet, P. Roy, and D. Cazaly. "Combinatorial approach to content-based music selection", *IEEE Multimedia*, Vol. 7, No. 1, pp. 44–51, 2000.

[14] Y.S. Park, and B. Kaneshiro. " An Analysis of User Behavior in Co-Curation of Music Through Collaborative Playlists", *Extended Abstracts for the Late-Breaking Demo Session of the 18th International Society for Music Information Retrieval Conference*, 2017.

[15] J. Pennington, R. Socher, and C. Manning. "Glove: Global Vectors for Word Representation", *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1532–1543, 2014.

[16] M. Pichl, E. Zangerle, and G.Specht. "Towards a Context-Aware Music Recommendation Approach: What is Hidden in the Playlist Name?", *Proc. of the 15th IEEE International Conference on Data Mining Workshop*, pp. 1360–1365, 2016.

[17] J. C. Platt, C. J. C. Burges, S. Swenson, C. Weare, and A. Zheng. "Learning a Gaussian process prior for automatically generating music playlists", *Advances in Neural Information Processing Systems*, Vol. 2, No. 14, pp. 1425–1432, 2002.

[18] C. Shannon. "A Mathematical Theory of Communication", *The Bell System Technical Journal*, Vol. 27, No. 1948, pp. 379–423, 1948.

[19] E. H. Simpson. "Measurement of diversity", *Nature*, Vol. 163, No. 688, 1949.

[20] A. Vall, M. Schedl, G. Widmer, M. Quadrana, and P. Cremonesi. "The importance of song context in music playlists: Enabling recommendations in the long tail", *CEUR Workshop Proc.*, 2017.

IDENTIFICATION AND CROSS-DOCUMENT ALIGNMENT OF MEASURES IN MUSIC SCORE IMAGES

Simon Waloschek, Aristotelis Hadjakos
Center of Music and Film Informatics
Detmold University of Music, Germany
{s.waloschek, a.hadjakos}@cemfi.de

Alexander Pacha
Institute of Information Systems Engineering
TU Wien, Austria
alexander.pacha@tuwien.ac.at

ABSTRACT

In the course of editing musical works, musicologists regularly compare multiple sources of the same musical piece, such as composers' autographs, handwritten copies, and various prints. For efficient comparison, cross-source navigation is essential, enabling to quickly jump back and forth between multiple sources without losing the current musical position. In practice, measures are first annotated by hand in the individual source images and then related to each other. Our approach automates this time-consuming and error-prone process with the help of deep learning. For this purpose, we train a neural network that automatically finds bounding boxes of all measures in images. A second network is trained to compute the similarity between two measures to determine if they have the same musical content and should, therefore, be linked for navigation. Sequences of outputs from the second network are matched using Dynamic Time Warping to provide the final proposal of measure relationships, so-called *concordances*. In addition to cross-source navigation, the results can be used to spot structural differences across the sources which are essential for editorial work, so that musicologists can focus more on analytical tasks.

1. INTRODUCTION

Modern musical editions are the result of a long musicological process. From the composer's manuscript to the printed music book, a musical work usually undergoes a large number of iterations and minor corrections, occasionally even substantial changes, such as striking or reworking complete parts [1]. Many of these changes are either unintentional—e.g., errors in handwritten copies, typographical errors by publishers—or generally not documented in a transparent manner. Musicologists, therefore, work on this genesis when editing a work and try to record the chronological order and causalities in their edition creation process.

The first step in this process is, therefore, the screening of the source material to identify differences between the various sources of a work. To facilitate this process, links are created between the sources so that editors can quickly switch back and forth between them. Adequate granularity of these links are usually musical measures, a feasible compromise between annotation effort and accuracy [29]. Currently, the measures of all sources are manually annotated with bounding boxes and related to each other in a very time-consuming and error-prone way.

We have automated this multi-stage process by first recognizing and sorting measures in score images (both handwritten and typeset) and then linking them according to their musical content. For this purpose, deep learning was used to develop a distance metric in an end-to-end fashion without an intermediate representation. The results can be further processed using classic alignment algorithms from the MIR community such as *Dynamic Time Warping* (DTW). While DTW-based approaches have achieved sufficient quality for practical use, audio-to-score alignment is still an active field of research [31]. Promising approaches for the synchronization of scans and sound recordings [5,6] are currently limited to monophonic and piano music and have not yet achieved sufficient accuracy for most real-world scenarios. With the contribution of this paper, we decrease a potential gap in the "audio – symbolic score – image" triangle and offer a new way for measure-accurate alignment across modal boundaries.

2. RELATED WORK

Detecting measures can be seen as a preprocessing step in Optical Music Recognition (OMR). Therefore, it was rarely singled out as a dedicated task. While Pedersoli and Tzanetakis perform document segmentation, they only distinguish between music scores and text blocks [22]. The only research we know of, that specifically addresses the automatic extraction of measures is by Vigliensoni et al. [30]. In their work, they attempt to extract measures with a traditional computer vision approach by heuristically finding all bar lines and then joining them into measures. Their approach requires human intervention for each page and straight bar lines to work well.

For retrieval of sixteenth-century musical texts, Crawford et al. [4] have recently proposed a two-step procedure. They run an OMR algorithm to obtain an intermedi-

ate format, followed by a second step that uses n-grams and minimal absent words (MAWs) to find duplicates, related texts, or parts that have the same musical material. Neural networks make such intermediate formats partly obsolete and allow for learning bimodal embeddings end-to-end as shown by Dorfer et al. [5, 6], who correlate the scanned music score with a sound recording. For this purpose, synchronization was considered either a reinforcement learning problem [6] or a metric learning problem [5]. In the metric learning approach, Dorfer et al. use the *pairwise ranking loss*—also known as *triplet loss* [26]—that draws triplets from a dataset consisting of an anchor, a positive example (picture fits the audio) and a negative example (picture does not fit the audio). This loss function creates an embedding, where images and audio with the same content are appear close together, while non-matching images and audio are placed relatively far apart. Their approach has successfully been used before in other application domains, such as facial recognition [26]. We resort to a similar cost function for metric learning (see section 4.2).

As the basis for our detection, we use a convolutional neural network (CNN). While CNNs are currently an active field of research for OMR, the most influential approaches come from the research area of computer vision. They are used for many tasks, including image recognition, semantic segmentation, object detection, and instance segmentation. R-CNN [9] performs object detection by analyzing a large number of heuristically generated region proposals that are classified into background or one of the classes of interest. Additionally, the bounding box is refined with regression. R-CNN uses a CNN that extracts features for object detection. These features are used in a downstream SVM for classification and regression. Faster R-CNN [23] improves the process by incorporating both the region proposal step as well as the classification and regression into the architecture of the neural network.

CNN-based computer vision approaches are largely transferable to OMR and actively used for Music Information Retrieval: Gallego and Calvo-Zaragoza are using auto-encoders to remove staff lines [8]. Pacha et al. compare various CNN-based approaches for detecting music symbols in scores [21]. CNNs can also be used for semantic segmentation for staff-line removal, music and text separation as well as for layout analysis as shown by Calvo-Zaragoza et al. [3]. Using U-Nets [25], Hajic et al. do semantical segmentation of handwritten music [10]. Pacha and Calvo-Zaragoza recognize musical objects in mensural notation using region-based CNNs [20]. By learning energy levels that are used as inputs to a watershed algorithm, Tuggener et al. recognize music symbols [28]. In addition to the energy levels, the network also predicts class labels and bounding boxes. And finally, Calvo-Zaragoza and Rizo use convolutional recurrent neural networks trained with a Connectionist Temporal Classification (CTC) loss to recognize musical symbols in monophonic music scores [2]. To simulate non-ideal image conditions, they artificially distort the images.

3. DATA & ANNOTATIONS

The success of Deep Learning approaches largely depends on the amount and diversity of data used during training. Since no dataset of sufficient size was available for measure recognition or the concordance task, we created a large dataset ourselves in cooperation with musicologists and professional musicians.

Our dataset contains measure annotations that were created manually by musicologists for digital music editions. In most cases, the image sources are high-resolution scans of facsimiles, occasionally supplemented by early music prints and PDFs exported directly from music engraving software. Due to an imbalance between handwritten and typeset scores, we additionally obtained scores from the *IMSLP/Petrucci Music Library* while paying attention to varying image quality, the used engraving mechanism as well as diverse musical content. We complemented our collection with 140 pages from the MUSCIMA++ dataset [1] [7, 11].

Our data collection has a total of 8 251 pages with 81 124 annotated measures. The distribution according to engraving type and the number of systems per page is given in Table 1. One category is particularly overrepresented: handwritten music scores with just one system per page because of a large quantity of full orchestral scores from operas by Carl Maria von Weber. Book covers, text pages, and empty pages have zero systems.

Systems per page	**Pages per engraving type**	
	Handwritten	Typeset
0	413	113
1	5627	932
2	175	553
3	122	175
4 or more	102	39
Total pages	6439	1812

Table 1. Overall distribution of the dataset used.

The accuracy of the measure annotations varies. Since the exact boundaries are not relevant for musicologists, they were recorded only roughly. That is why many bounding boxes contain small overlaps with adjacent measures as shown in Figure 1.

To annotate the measures in the individual pictures, the Android app *Vertaktoid* [2] [18] was used. It allows to conveniently draw bounding boxes for all measures with a pen directly on the tablet screen. The results can then be exported to the MEI format [24] and used as ground truth training data.

Data coming from digital music editions are partly provided with concordance annotations between the measures.

[1] The measure annotations are published as separate dataset at `https://apacha.github.io/OMR-Datasets/#muscima`

[2] `https://github.com/cemfi/vertaktoid`

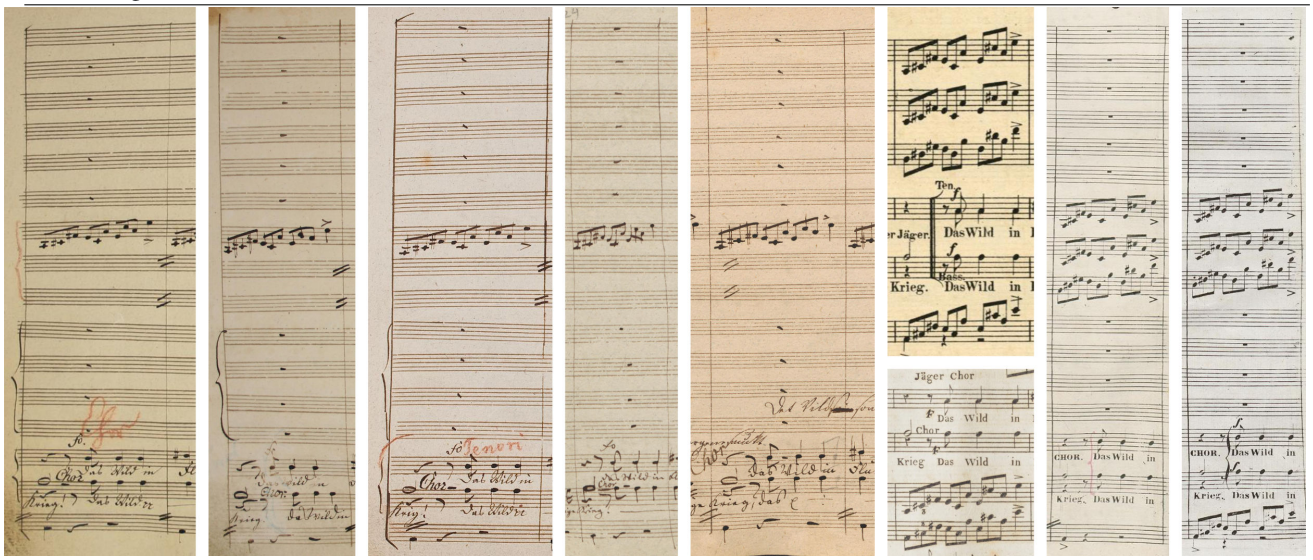


Figure 1. Examples of cropped measures originating from different sources of the same work. All measures represent the same musical position, i.e. the same measure, within the work, but are in part extremely diverse in terms of instrumentation, graphic representation and also image resolution.

4. ALIGNING MEASURE SEQUENCES

Our proposed solution for the given task can be split into three individual parts. First, we have to find the bounding boxes of all measure in the score images. Then we need a metric in order to compute the similarity between two given measure in terms of musical content. And finally, we have to compute actual concordances for multiple sources of the same music.

4.1 Optical Measure Recognition

For automatically detecting measures in complete music scores, we propose a machine-learning approach with deep convolutional neural networks and a Faster R-CNN detector [23]. Faster R-CNN has been shown to work well in a range of situations, including detecting music objects [21]. In this case, there is just one class of objects that needs to be detected, and the objects typically cover large portions of the entire image with little overlap. Our implementation is based on the TensorFlow Object Detection API framework [14] and freely available online [3] .

We split the dataset randomly into 80% for training, 10% for validation, and 10% for testing. To avoid a bias toward scores with just one system, we sample the images equally from the ten categories depicted in table 1. The only exception are images without systems which are sampled only half as often as the other categories.

We tested the three different backbones, ResNet50, ResNet101 [13], and Inception-ResNet-V2 [27] and restricted ourselves to these to enable transfer-learning by initializing the networks with weights trained on ImageNet which generally improves the learning process, especially at the beginning. Input images are resized to be no longer than 1024 pixel on the longest edge. The Intersection over

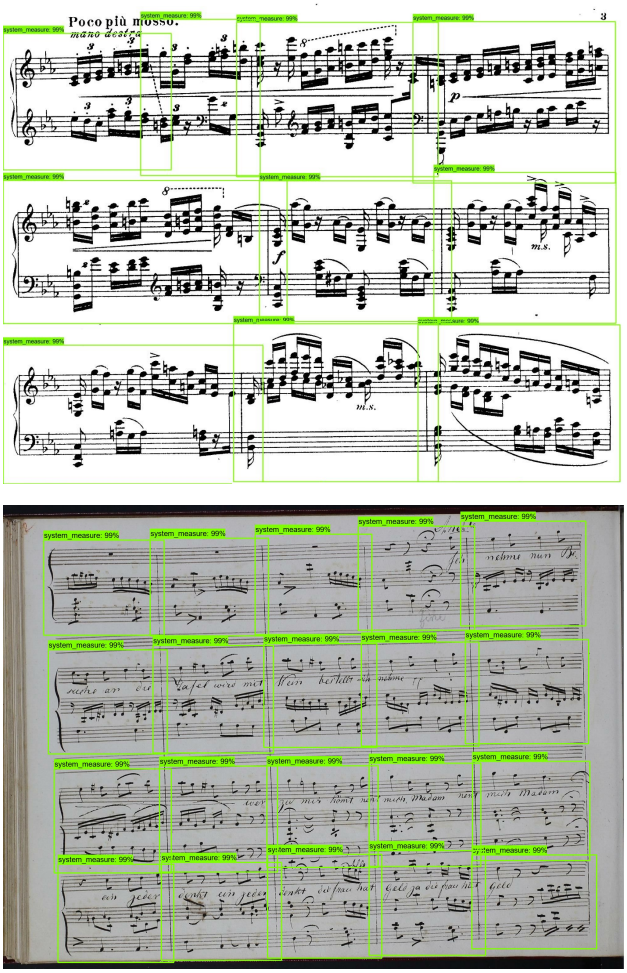


Figure 2. Two samples of the detection results. Measures are detected robustly in typeset and handwritten scores without the need for preprocessing the images.

[3] https://github.com/OMR-Research/MeasureDetector

Union (IoU) measures how well two bounding boxes overlap. If two predictions are very close, non-maximum suppression filters the box with the lower score. The IoU threshold is set to 0.6 and a maximum of 600 objects are detected per image. These parameters are derived from statistical analysis of the entire data set and cover $> 99.99\%$ of the dataset.

We evaluated the optical measure detection with the commonly used average precision (AP) metric, as defined for the COCO detection challenge [15]. It produces a single number that measures how well objects were detected. A detection is considered a match with the underlying ground truth if the IoU is above a certain threshold. The trained models achieve very good results with 78.7% AP (IoU=0.5:0.95) on the test set for the top-performing model with Inception-ResNet-V2 [27] backbone. A few samples of the detection output are depicted in Figure 2.

Given that the measure recognition step does not necessarily return the measures of a page in the musically correct order, we sort them according to the measure numbering rules outlined by Mexin et al. in [18].

4.2 Metric Learning

Now that the scans of all scores are divided into individual measures, they have to be compared with each other to identify equivalent measures. Again, a deep learning approach is used to learn such a musical similarity metric between two measures directly from the images. The neural network is trained to compute an embedding for measure images so that similar measures are placed in the proximity of one another in the embedding space. This allows for convenient comparison of two measures by computing their distance, e.g., using the L^2 norm.

The idea is based on *triplet loss* [26]: A pair of equivalent measure images from two different sources is drawn from the list of concordances. We will call them the *anchor* image and the *positive* image. Additionally, a *negative* measure image is drawn from the same source as the positive image, serving as a counterexample, i.e. having no musical relation to the anchor or the positive measure image. Each of these three images is fed separately into the same neural network, resulting in three k-dimensional vectors. The loss function is defined as

$$\mathcal{L} = max(d(f^a, f^p) - d(f^a, f^n) + \alpha, 0) \qquad (1)$$

with f^a, f^p, and f^n being the resulting vectors from the network f for the three images and a distance measure d. Training with this loss function minimizes the distance from the anchor to the positive image while maximizing the distance between the anchor and the negative image. The additional margin α defines how far away the least dissimilarity should be. Finally, the surrounding $max(...)$ function ensures that the loss never gets negative.

We chose ResNet50 as the base network and replaced the usual final average pooling and classification layers by a fully connected layer with k-dimensional output. (Other CNN-based networks used for computer vision would most likely work comparably well.) All measure images are resized to 512×512 pixels but the original width and height information is also passed to the network as additional input.

The success of the used loss function depends heavily on the sampling strategy for the image triplets as discussed by Wojke and Bewley in [32]. In our context, there are three specific problems in the dataset:

1. A randomly sampled negative image might accidentally have the same musical content as the two other images. Those cases are not covered in the concordance dataset since not all measures with equal content have to be linked together.
2. Intuitively, it seems beneficial to take the previous or subsequent measure of the positive sample as the negative measure with the goal of enhancing the contrast between them in terms of increased distance in the embedding space. This would make adjacent measures more distinguishable. But again, the chance of these measures having the same content is higher compared to random sampling.
3. Especially handwritten sources sometimes exhibit excessive use of measure repeats and other abbreviations as can be seen in the left part of Figure 1. Such symbols are meaningless if their immediate context is not given.

The first two problems could be solved by manually adding all measures with the same content to the list of concordances. Given the amount of images, we decided against doing so and rely on rare collisions thanks to the large number of data. We also discarded the (perfectly valid) idea of looking at adjacent measures to form the triplets.

The third problem—presence of measure repeats and abbreviations—has a direct impact on the appropriate choice of the distance metric d in our loss function; When using triplet loss, it is common practice to normalize the embedding vectors. This constraint puts all embeddings on a k-dimensional hypersphere, leading to some advantages for further processing (see [26]). Furthermore, *cosine distance* is often used to calculate the distances. Both decisions make it impossible to get an embedding vector that is equally distant to all other possible vectors. This very property, however, characterizes the meaning of measure repeats if no context is given. We, therefore, opted for no vector normalization and chose the L^2 norm as our distance metric, resulting in

$$\mathcal{L} = \sum_{i=1}^{N} \left[\|f_i^a - f_i^p\|_2 - \|f_i^a - f_i^n\|_2 + \alpha \right]_+ \qquad (2)$$

for a training batch with size N. To speed up training and ensure fast convergence we select triplets that violate the following constraint:

$$\|f_i^a - f_i^p\|_2 + \alpha < \|f_i^a - f_i^n\|_2 . \qquad (3)$$

This filter step is performed for each batch during training and makes sure that only those triplets are used that significantly contribute to the learning process. It also prevents the network from overfitting.

4.3 Concordance Computation & Manual Adjustments

Given the embedding vectors for all measures of each source of a musical work, we can compare two sources by computing the distances between all measures from one source to the other. The resulting similarity matrices can then be used for *dynamic time warping* (DTW) as described by Müller in [19] to get an alignment path between the sources as shown in Figure 3.

We implemented the canonical DTW algorithm without any noteworthy modifications to the core. Allowed step sizes inside the similarity matrix during path computation are $(0, 1)$, $(1, 0)$, and $(1, 1)$. It rarely happens that a measure gets divided into two parts at system or page breaks, so we penalized steps along a single axis by a factor of 2 to slightly enforce one-to-one mappings of the measures.

The quality of the alignment was evaluated using a dataset with two sources and given ground truth concordances as outlined in Table 2. We have decided in favor of this particular dataset because it offers several challenges that occur only rarely in other works:

Split measures: Some measures are split into two parts at page breaks. Therefore, one measure of source A maps to two other measures of source B.

Completely different sections: An entire part of the piece was replaced in source B. Finding the "correct" concordance is impossible.

Additional parts: Source B contains a 16-measure Aria that is not present in the other source.

Missing measure annotations: We also intentionally removed measures from source A to simulate annotation errors.

	Pages	Measures
Source A (typeset)	250	3098
Source B (handwritten)	532	3176
Total	782	6274

Table 2. Structure of the evaluation dataset.

In the MIR community, DTW is often used to synchronize audio and/or symbolic score sources with each other [12]. The time resolution of the features in such scenarios is usually in the range of several dozen milliseconds. Deviations in the alignment path are therefore undesirable, but can often be neglected as long as they do not exceed certain limits. In our context, however, any deviation from the ground truth marks a significant error. We took this into account and defined a very simple score for the overall performance:

$$score = 1 - \frac{\text{Number of } (x, y) \text{ pairs from alignment not in ground truth}}{\text{Total number of concordances in ground truth}} \quad (4)$$

Our evaluation showed 14 errors in relation to 3079 concordance pairs, resulting in a score of **99.545%**.

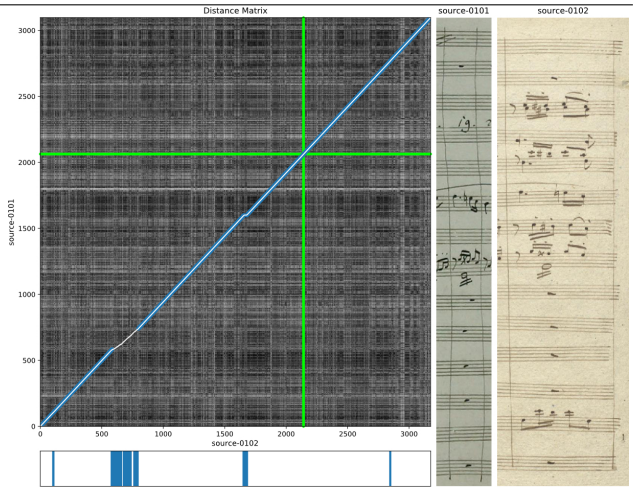


Figure 3. Interface for inspecting the computed measure concordances. The alignment (white) and ground truth (blue, only available in evaluation dataset) are plotted over the currently visible part of the similarity matrix. Measures of both sources (right) can be compared by moving a cursor within the matrix (green crosshair). A plot at the bottom indicates potentially interesting positions.

As pointed out, the remaining 0.455% error rate still present a non-negligible problem. Therefore, we developed an interface for manual adjustments to the alignment. Apart from being able to quickly compare the measures from two sources as shown in Figure 3, users can define points in the similarity matrix that have to be part of the alignment path. Each of these points splits the matrix into two parts and computes the warping path for each part individually, ensuring that either the beginning or end of the path matches the desired point. An event plot at the bottom of the matrix helps to identify regions with potential errors by showing where the alignment path is not diagonal, i.e. taking a step in $(0, 1)$ or $(1, 0)$ direction.

The mentioned obstacles for correct alignment have been handled successfully by either resulting in a correct alignment or—in case of substantial structural differences—indicating a problem that cannot be solved without human intervention by marking these parts in the plot below the similarity matrix.

This alignment and adjustment step has to be repeated for each source in regard to a *master source* of choice. The corrected alignment data can then finally be imported into the tools used by musicologists for their editorial work.

5. CONCLUSIONS AND FUTURE WORK

In this paper we proposed an approach to automate the tedious task of annotating and linking measures in heterogeneous score images, thereby allowing for cross-source navigation between measures without losing the current musical position. We used deep learning to find bounding boxes of measures in score images, learned a distance metric for measures, and used that to align measures from various sources, effectively linking equivalent musical po-

sitions across sources. The evaluation showed that our approach is feasible and solves a real-world problem while still retaining complete flexibility in case editors need to make manual adjustments, thanks to an interactive correction tool.

The presented solution still does not cover all possible situations that might occur in the editorial process. If the measure sequences to be compared have a different order, the alignment fails for these parts if not completely. We will address this specific problem in the future by identifying such passages and proposing reasonable re-ordering.

Having a musically meaningful distance metric for measures also allows closing the gap between score images and symbolic scores. The latter can be rendered with suitable engraving software and divided into individual measures, followed by the steps of our alignment pipeline. Since audio can also be rendered from symbolic scores, alignments between all three modalities are possible.

Another interesting application of our distance metric is the ability to visualize datasets in image fields as shown in Figure 4. Using dimensionality reduction algorithms such as T-SNE [16] or UMAP [17], the measures are positioned such that musically similar measures appear proximate to one another, giving new insight into a musical piece but also into the inner workings of the distance metric. For example, the visualization shows that measure repeats are placed almost in the center, indicating that their learned embedding retains the musical property of being close to basically every other measure in the embedding space.

Figure 4. 46 344 measure images from 15 different sources of the same piece are projected into a two-dimensional manifold with the UMAP algorithm. The map is interactively zoomable.

6. REFERENCES

[1] Benjamin W. Bohl, Axel Berndt, Simon Waloschek, and Aristotelis Hadjakos. Dem Igel Sitte lehren... Musikedition: von der digitalen Verfügbarkeit zur aktiven Nutzung. In Kristina Richts and Peter Stadler, editors, *„Ei, dem alten Herrn zoll' ich Achtung gern'" – Festschrift für Joachim Veit zum 60. Geburtstag*, chapter 12, pages 141–163. Allitera Verlag, Munich, Germany, 2016.

[2] Jorge Calvo-Zaragoza and David Rizo. Camera-primus: Neural end-to-end optical music recognition on realistic monophonic scores. In *19th International Society for Music Information Retrieval Conference*, pages 248–255, Paris, France, 2018.

[3] Jorge Calvo-Zaragoza, Gabriel Vigliensoni, and Ichiro Fujinaga. A machine learning framework for the categorization of elements in images of musical documents. In *3rd International Conference on Technologies for Music Notation and Representation*, A Coruña, Spain, 2017. University of A Coruña.

[4] Tim Crawford, Golnaz Badkobeh, and David Lewis. Searching page-images of early music scanned with OMR: A scalable solution using minimal absent words. In *19th International Society for Music Information Retrieval Conference*, pages 233–239, Paris, France, 2018.

[5] Matthias Dorfer, Jan Hajič jr., Andreas Arzt, Harald Frostel, and Gerhard Widmer. Learning audio–sheet music correspondences for cross-modal retrieval and piece identification. *Transactions of the International Society for Music Information Retrieval*, 1(1):22–33, 2018.

[6] Matthias Dorfer, Florian Henkel, and Gerhard Widmer. Learning to listen, read and follow: Score following as a reinforcement learning game. In *19th International Society for Music Information Retrieval Conference*, pages 784–791, Paris, France, 2018.

[7] Alicia Fornés, Anjan Dutta, Albert Gordo, and Josep Lladós. CVC-MUSCIMA: A ground-truth of handwritten music score images for writer identification and staff removal. *International Journal on Document Analysis and Recognition*, 15(3):243–251, 2012.

[8] Antonio-Javier Gallego and Jorge Calvo-Zaragoza. Staff-line removal with selectional auto-encoders. *Expert Systems with Applications*, 89:138–148, 2017.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158, 2016.

[10] Jan Hajič jr., Matthias Dorfer, Gerhard Widmer, and Pavel Pecina. Towards full-pipeline handwritten OMR

with musical symbol detection by u-nets. In *19th International Society for Music Information Retrieval Conference*, pages 225–232, Paris, France, 2018.

[11] Jan Hajič jr. and Pavel Pecina. The MUSCIMA++ dataset for handwritten optical music recognition. In *14th International Conference on Document Analysis and Recognition*, pages 39–46, Kyoto, Japan, 2017.

[12] Yun Hao. Real-time audio to score alignment (a.k.a score following). `https://www.music-ir.org/mirex/wiki/2019:Real-time_Audio_to_Score_Alignment_(a.k.a_Score_Following)`, 2019.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recogntiion (CVPR)*, pages 770–778, 2016.

[14] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014.

[16] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[17] Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[18] Yevgen Mexin, Aristotelis Hadjakos, Axel Berndt, Simon Waloschek, Anastasia. Wawilow, and Gerd Szwillus. Tools for annotating musical measures in digital music editions. In *14th Sound and Music Computing Conf. (SMC-17)*, Espoo, Finland, 2017. Aalto University.

[19] Meinard Müller. *Information Retrieval for Music and Motion*. Springer-Verlag, Berlin, Heidelberg, 2007.

[20] Alexander Pacha and Jorge Calvo-Zaragoza. Optical music recognition in mensural notation with region-based convolutional neural networks. In *19th International Society for Music Information Retrieval Conference*, pages 240–247, Paris, France, 2018.

[21] Alexander Pacha, Jan Hajič jr., and Jorge Calvo-Zaragoza. A baseline for general music object detection with deep learning. *Applied Sciences*, 8(9):1488–1508, 2018.

[22] Fabrizio Pedersoli and George Tzanetakis. Document segmentation and classification into musical scores and text. *International Journal on Document Analysis and Recognition*, 19(4):289–304, 2016.

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28*, pages 91–99. 2015.

[24] Perry Roland. The music encoding initiative (MEI). In *1st International Conference on Musical Applications Using XML*, pages 55–59, 2002.

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 234–241. Springer, 2015.

[26] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[27] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. Inception-v4, Inception-ResNet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, 2017.

[28] Lukas Tuggener, Ismail Elezi, Jürgen Schmidhuber, and Thilo Stadelmann. Deep watershed detector for music object recognition. In *19th International Society for Music Information Retrieval Conference*, pages 271–278, Paris, France, 2018.

[29] Joachim Veit and Kristina Richts. Current status and perspectives of MEI usage in musicology and in libraries. *Bibliothek Forschung und Praxis*, 42(2):292–301, 2018.

[30] Gabriel Vigliensoni, Gregory Burlet, and Ichiro Fujinaga. Optical measure recognition in common music notation. In *14th International Society for Music Information Retrieval Conference*, Curitiba, Brazil, 2013.

[31] S. Waloschek and A. Hadjakos. Driftin' down the scale: Dynamic time warping in the presence of pitch drift and transpositions. In *19th International Society for Music Information Retrieval Conference*, Paris, France, 2018.

[32] Nicolai Wojke and Alex Bewley. Deep cosine metric learning for person re-identification. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 748–756. IEEE, 2018.

QUERY-BY-BLENDING: A MUSIC EXPLORATION SYSTEM BLENDING LATENT VECTOR REPRESENTATIONS OF LYRIC WORD, SONG AUDIO, AND ARTIST

Kento Watanabe **Masataka Goto**
National Institute of Advanced Industrial Science and Technology (AIST), Japan
{kento.watanabe, m.goto}@aist.go.jp

ABSTRACT

This paper presents *Query-by-Blending*, a novel music exploration system that enables users to find unfamiliar music content by flexibly combining three musical aspects: lyric word, song audio, and artist. Although there are various systems for music retrieval based on the similarity between songs or artists and for music browsing based on visualized songs, it is still difficult to explore unfamiliar content by flexibly combining multiple musical aspects. Query-by-Blending overcomes this difficulty by representing each of the aspects as a latent vector representation (called a "flavor" in this paper) that is a distinctive quality felt to be characteristic of a given word/song/artist. By giving a lyric word as a query, for example, a user can find songs and artists whose flavors are similar to the flavor of the query word. Moreover, by giving a query combining (blending) lyric-word and song-audio flavors, the user can interactively explore unfamiliar content containing the blended flavor. This multi-aspect blending was achieved by constructing a novel vector space model into which all of the lyric words, song audio tracks, and artist IDs of a collection can be embedded. In our experiments, we embedded 14,505 lyric words, 433,936 songs, and 44,696 artists into the same shared vector space and found that the system can appropriately calculate similarities between different aspects and blend flavors to find related lyric words, songs, and artists.

1. INTRODUCTION

Given a huge collection of musical pieces such as those provided by online music services, conventional music access based on bibliographic metadata like song titles and artist names has not been sufficient. The Music Information Retrieval (MIR) community, therefore, has covered various types of music retrieval and exploration. When a listener wants to listen to familiar musical pieces, a popular approach is content-based music retrieval [4, 15, 26, 31] based on music similarity. When a query is entered by humming [5, 8, 12, 24, 29, 33], for example, similarities between the query and melody lines in a database are computed to show its title. When a query is given by a musical piece [7, 17, 19, 25, 28], a ranked list of similar musical pieces is shown. Music retrieval based on various kinds of metadata [3, 6, 14, 27] is also proposed. Although music would have multiple aspects, previous approaches typically assume a single aspect as a query. Moreover, it is necessary for users to conceive appropriate queries, which is sometimes not easy.

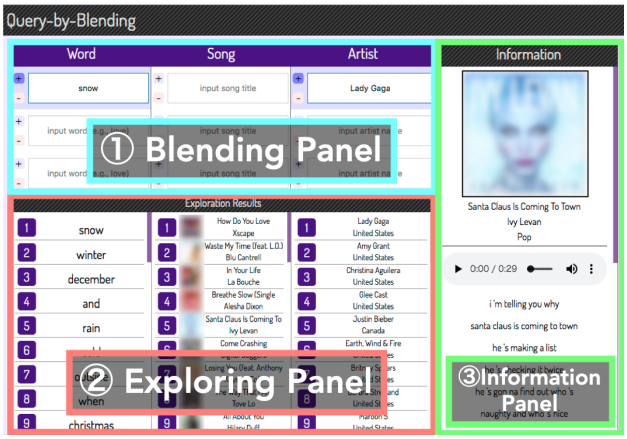


Figure 1. Query-by-Blending interface that consists of blending, exploring, and information panels.

When a user wants to discover unfamiliar musical pieces, an approach of music exploration is important. Music exploration systems typically provide interfaces that visualize a music collection by embedding musical pieces or artists into a 2D or 3D space and let users explore the collection to find favorite pieces [11, 21, 25, 30] or artists [1, 25, 27, 32]. However, it is difficult for previous music exploration systems to take different aspects of music into account. Another approach is to help users flexibly conceive a variety of queries for music discovery [9]. Such assistance, however, has not been investigated much in the MIR community.

We therefore propose a novel music exploration system, *Query-by-Blending*, that can embed three musical aspects – lyric word (word in lyrics), song audio (audio signal of a song), and artist (represented as artist ID) – into a unified high-dimensional latent vector space and enable users to find unfamiliar but interesting music content by flexibly combining those aspects (Figure 1). Query-by-Blending

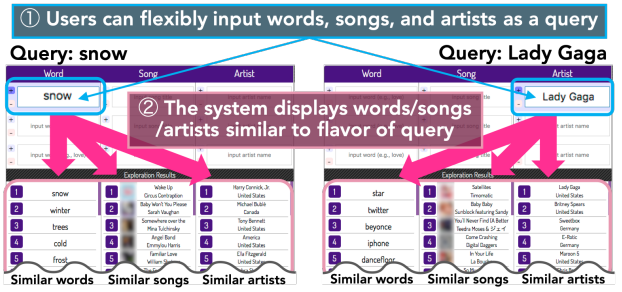


Figure 2. Query-by-Blending enables the exploration of three musical aspects: lyric word, song audio, and artist.

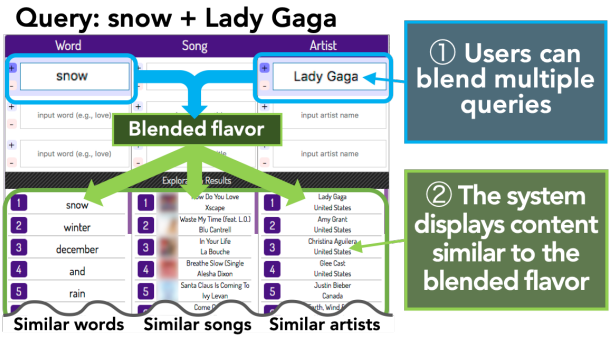


Figure 3. By blending multiple flavors, Query-by-Blending can display music content similar to the blended flavor. The blended flavor "snow + Lady Gaga" is similar to Christmas, Pop, and Soul songs/artists.

uses a "flavor" and "blending flavors" metaphor to help users combine different aspects to flexibly conceive a variety of queries. The terms "lyric-word flavor", "song-audio flavor", and "artist flavor" denote latent vector representations that are distinctive qualities felt to be characteristic of a given word, song, and artist. Each of the three kinds of flavors can be used as a query to retrieve and display three kinds of ranked lists: related lyric words, titles of songs having related song audio, and related artist names.

By giving a favorite artist as a query, for example, a user can not only listen to various songs containing its artist flavor but also see lists of lyric words and artists containing its artist flavor. Since the retrieved songs are not necessarily songs by the query artist, a user can explore a variety of unfamiliar music content. Since all three of the musical aspects are embedded into the same latent vector space as flavors, a user can add another flavor to "blend flavors" (i.e., give a query combining multiple musical aspects). Adding a lyric-word flavor, for example, causes the displayed lists to be interactively updated to give every musical content containing that flavor as well as the previous flavor a higher rank. Query-by-Blending can thus provide novel interactive, incremental, and iterative exploration experiences based on multiple musical aspects.

In implementing Query-by-Blending, it is difficult to calculate similarities among the three musical aspects because there are no large-scale annotations for supervised learning of such similarities. To overcome this difficulty, we propose a method of constructing a latent vector space model that can be trained with unsupervised learning under the assumption that a lyric word, song audio, and artist sampled from the same song tend to have similar meanings and are mapped to positions close to each other in the unified vector space. This method is based on multi-task learning, which has been used successfully across various applications of neural networks, and uses a vector model that can learn shared representations of music content by separately training each aspect of a large music collection (one including 14,505 words, 433,936 songs, and 44,696 artists).

2. QUERY-BY-BLENDING

Query-by-Blending enables a user to iteratively issue a query of any combination of lyric words, song titles, and artist names and obtain ranked lists of lyric words, song titles, and artist names that are similar to the query. The interface of Query-by-Blending is shown in Figure 1 and consists of (1) a blending panel for issuing the query, (2) an exploring panel displaying the retrieved ranked lists and allowing the user to select a song, and (3) an information panel displaying the title, artist name, and lyrics of the selected song and allowing the user to play back a short excerpt of the song for trial listening.

2.1 Exploring Three Musical Aspects

When a user enters a lyric word, song title, or artist name as a query on the blending panel, the exploring panel displays the retrieved lists of lyric words, song audio tracks, and artists whose flavors (latent vector representations) are similar to the flavor of the query. The lists are sorted in the order of the similarity. Figure 2 shows two screenshots of the interface. As shown in the left side of Figure 2, for example, when a user types "snow" into the lyric word text field on the blending panel, the exploring panel displays the lyric word "winter", "Angel Band" (the title of a song containing the word "snow"), and "Harry Connick, Jr." (who released the Christmas song album "When My Heart Finds Christmas", which can be considered to be related to "snow"). On the other hand, as shown in the right side of Figure 2, when the user types "Lady Gaga" into the artist text field on the blending panel, the exploring panel displays music content with "Lady Gaga" flavor – lyric words "star" and "dance floor", songs of Pop and Dance music sung by female singers, and an American female singer "Britney Spears". The user can then click one of the displayed songs to listen to its excerpt. These results and their music excerpts are available at a web page (`https://kentow.github.io/qbb/`). Query-by-Blending thus enables the user to flexibly issue a query to explore music content that is similar to the query.

2.2 Blending and Subtracting Flavors

Although issuing a single-flavor query with one of the three aspects (flavors) has already been useful, the blending panel further enables a user to issue a query blending multiple flavors. As shown in Figure 3, for example, when the user

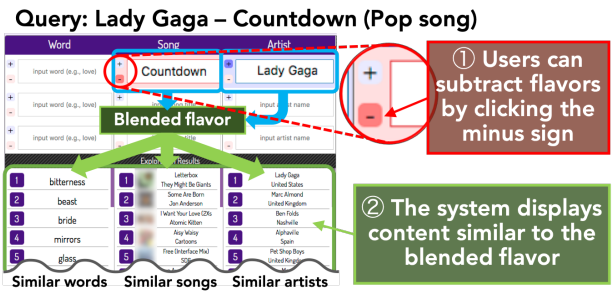


Figure 4. Query-by-Blending allows the user to subtract flavors from others. The flavor "Lady Gaga − Countdown (Pop song)" is similar to Alternative and Rock songs/artists.

types both "snow" and "Lady Gaga" into the two text fields on the blending panel, the retrieved ranked lists are updated to be more similar to the blended flavor of both of them. The exploring panel displays a Christmas song sung by the female Soul singer "Ivy Levan". Then the user can feel free to iteratively add more flavors.

Moreover, when a user issues a query and finds some of the listed songs or artists unappealing, the blending panel enables the user to subtract a flavor related to them from the current query. As shown in Figure 4, when the user subtracts a Pop song "Countdown" having a "Pop song" flavor from the original query of "Lady Gaga" (by clicking the minus sign (−) located beside the text field of the song "Countdown"), the exploring panel updates the lists to include Alternative and Rock music songs (e.g., "Letterbox") and artists (e.g., "Ben Folds") that are similar to the flavor "Lady Gaga − Countdown". The user can also subtract some flavors after blending multiple flavors.

When a user issues a query and finds unfamiliar songs or artists interesting after trial listening on the information panel, the user can add (blend) some of them to the current query to update the retrieved lists. Query-by-Blending thus provides novel exploration experiences, such as being able to incrementally conceive a variety of queries including both familiar and unfamiliar songs and artists. It enables users to flexibly update their queries by blending and subtracting various flavors to explore unfamiliar but interesting music content interactively in a trial-and-error manner.

3. IMPLEMENTATION

We implemented Query-by-Blending by developing a novel unsupervised method of constructing the unified latent vector space in which similar aspects are located nearby. Since similarities related to the three musical aspects are not annotated in a typical music collection, we leveraged the *Distributional Hypothesis* [10] that is well-known in the field of Natural Language Processing and has successfully been used in *word2vec* [23].

To explain the mechanism of capturing similarities, we first focus on the similarity between two lyric words without using audio and artist aspects. According to the Distributional Hypothesis, we assume that *words that occur in similar contexts tend to have similar topics*. As an example,

		Context word				Context audio		
		cold	white	devil	…	Relaxed audio	Tense audio	Sad audio
Target word	snow	41911	32910	33	…	50192	288	101
	winter	52202	22291	52	…	48381	210	134
	dark	18	213	43982	…	104	32017	89
	shadow	31	114	50928	…	251	22015	72
Target song audio	Jazz song A	12	9	0	…	33	10	3
	Jazz song B	10	11	0	…	43	9	7
	Metal song	0	0	20	…	4	20	1
Target artist	Jazz artist	330	221	19	…	513	45	25
	Metal artist	80	20	531	…	32	384	16

Figure 5. The Distributional Hypothesis for multiple aspects.

consider the following two lyrics:

*"***Snow** *white side street of cold NewYork City."*

"But here in the white of a cold **winter** *night, ..."*

Since "snow" and "winter" each co-occur with "cold" and "white" (i.e., the same context), we can assume that "snow" and "winter" have a similar topic. In this paper, we define a *context word* (e.g., "cold" or "white") to be a word co-occurring with a *target word* (e.g., "snow" or "winter") in the same song. This mechanism is expressed by a *co-occurrence matrix* where each row corresponds to a target word and columns give the context words (the red frame in Figure 5). In this matrix, each cell contains the frequency of co-occurrence of the target word and the context word in all the songs. In Figure 5, the target words "snow" and "winter" have high co-occurrence with the context words "cold" and "white" but low co-occurrence with "devil". Other target words "dark" and "shadow" frequently co-occur with the context word "devil". By calculating the co-occurrence matrix, we can estimate that "snow" and "winter" have similar topics and that "dark" and "shadow" have similar topics. We call each row of this matrix a *target word vector* and calculate the similarity between target words as the distance between their target word vectors.

We then extend this co-occurrence matrix to *context audio tracks* so that we can utilize audio signals for capturing the similarity between target words. Each context audio is a short fragment of song audio and is represented as an *audio-word* defined in Section 3.2. In the green frame of Figure 5, the target words "snow" and "winter" tend to co-occur with some short fragments of song audio and are considered similar, but other target words "dark" and "shadow" do not co-occur with those short fragments. Both the context words and context audio tracks can thus be used to capture the similarity and are included in each *target word vector*. As with the similarity between target words, we assume that *songs that occur in similar contexts tend to have similar topics*. In the blue frame of Figure 5, we call each row of a target song audio in the co-occurrence matrix a *target audio vector*. Each cell of a target audio vector contains the number of occurrences of a context word in lyrics of the song corresponding to the target song audio, or contains the number of occurrences of a context audio in the target song audio. In Figure 5, the target vectors of "snow", "Jazz song audio", and "Jazz artist" are close to context vectors

of "cold" and "white"; thus these multiple aspects can be located near each other. Moreover, we can also naturally calculate the similarity between any target word vector and any target audio vector as the distance between them since those vectors are represented in the same matrix.

Finally, in the same way, we can extend the co-occurrence matrix to *target artists*. In the bottom part of Figure 5, we call each row of a target artist in the co-occurrence matrix a *target artist vector*. Each cell of a target artist vector contains the number of occurrences of a context word in lyrics of all songs by its artist, or contains the number of occurrences of a context audio in all song audio tracks by its artist. By extending the Distributional Hypothesis, we can calculate similarities related to the three aspects.

3.1 Dataset

To calculate the above similarities, we made a dataset containing 433,936 songs by 44,696 artists. The dataset item for each song consists of a text file of English lyrics provided by a lyrics distribution company, an audio file of a short music excerpt (30 sec, 44.1kHz) available for trial listening on a music service, and an artist ID. Here, each text file contains all sentences of the lyrics of a song. We extracted 14,505 frequent lyric words from all the text files and did not use words that appeared less than 100 times.

3.2 Creating Audio-word Representation

To represent a short fragment of song audio for a context audio, we use a discrete symbol called an *audio-word*. The audio-word can be obtained by a *bag of audio-words* (BoAW) model [18] as follows. (1) Each music excerpt is downsampled to 22,050 Hz. (2) We use *LibROSA*, a python package for music and audio analysis, to extract 20-dimensional mel-frequency cepstral coefficients (MFCCs) with the FFT size of 2048 samples and the hop size of 512 samples. This result is represented as an MFCC matrix (20×1280). (3) The MFCC matrix is divided into 128 submatrices (20×10) without overlap. (4) Each submatrix including 10 frames of MFCCs is flattened into a 200-dimensional vector that represents local temporal dynamics of MFCCs. (5) We use the k-$means$++ algorithm [2] to group all 200-dimensional vectors of all 433,936 songs into 3,000 clusters. (6) Each cluster is regarded as a discrete audio-word. We thus obtained 3,000 audio-words.

3.3 Vector Space Model for Multiple Musical Aspects

Since the co-occurrence matrix is huge and extremely sparse, it is too computationally expensive to deal with. To overcome this problem, our latent vector space model uses a neural network to reduce the huge matrix to a dense matrix as *word2vec* [23] also does.

The structure of the model is illustrated in Figure 6. Let w_t denote the target word, let aw_m denote the audio-word, let a denote the artist ID, and let c denote the context consisting of the context word and context audio-word. To obtain a D-dimensional latent vector space/representation

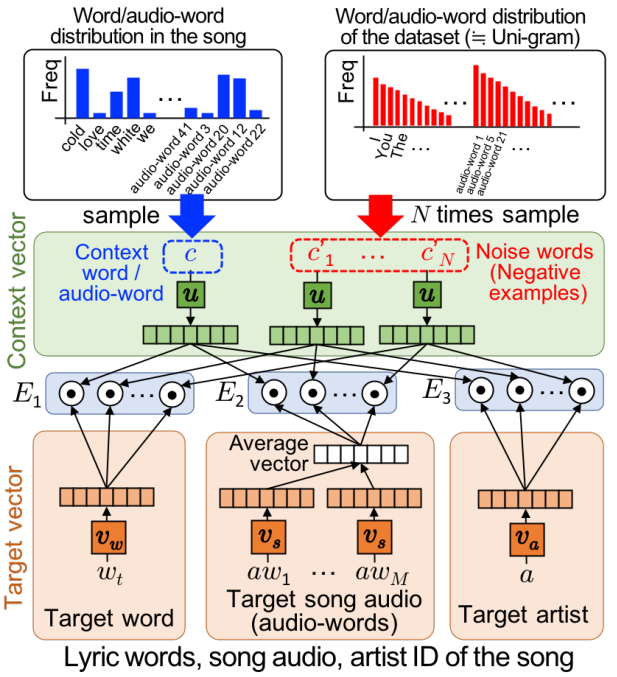


Figure 6. Multiple musical aspect vector space model.

after dimension reduction, we define an embedding function $\boldsymbol{v_w}(\cdot)$ that maps the target word to a D-dimensional vector and define an embedding function $\boldsymbol{u}(\cdot)$ that maps the context word/audio-word to a D-dimensional vector.

We formulate this as an optimization problem that minimizes the distance between $\boldsymbol{v_w}(w_t)$, the target vector of a target word w_t in a song, and $\boldsymbol{u}(c)$, the context vector of another context word or a context audio-word c randomly sampled from the same song. By iterating this sampling and minimization for every target word, the model captures the similarity between target words. In Figure 5, for example, "snow" and "winter" are frequently sampled for the target words, and "cold" and "white" are frequently sampled for the context words. In other words, the vectors of the target words "snow" and "winter" are close to both of the vectors of context words "cold" and "white" in the embedded vector space. Thus, these target word vectors can be located near each other. In the training phase to minimize the above distance, we maximize $\boldsymbol{u}(c)^{\mathsf{T}} \cdot \boldsymbol{v_w}(w_t)$, the dot product of the context word/audio-word vector $\boldsymbol{u}(c)$ and the target word vector $\boldsymbol{v}(w_t)$ in the lyrics of a song. This maximization can be done by optimizing parameters of $\boldsymbol{u}(\cdot)$ and $\boldsymbol{v_w}(\cdot)$.

To accelerate training, we not only maximize the dot product of co-occurring w and c but also minimize the dot product of w and c', where c' is a noise word. This technique is called *Negative Sampling* and is known to be useful in training word2vec [23]. We define and minimize the objective function E_1 to maximize $\boldsymbol{u}(c)^{\mathsf{T}} \cdot \boldsymbol{v_w}(w_t)$ and minimize $\boldsymbol{u}(c')^{\mathsf{T}} \cdot \boldsymbol{v_w}(w_t)$:

$$E_1 = -\log\sigma\Big(\boldsymbol{u}(c)^{\mathsf{T}} \cdot \boldsymbol{v_w}(w_t)\Big) - \sum_{n=1}^{N} \log\sigma\Big(-\boldsymbol{u}(c'_n)^{\mathsf{T}} \cdot \boldsymbol{v_w}(w_t)\Big), \quad (1)$$

where $\sigma(\cdot)$ is a sigmoid function. N is the number of negative words/audio-words $c'_n (1 \le n \le N)$ sampled from

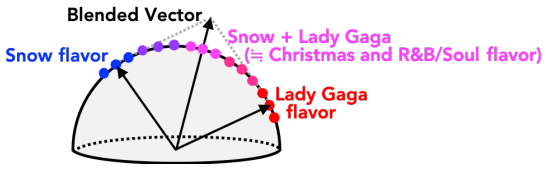


Figure 7. Normalized multiple aspect vector on hypersphere. Each circle denotes an embedded aspect. The cosine similarity between similar aspects is large.

the following noise distribution:

$$P(c'_n) = \#(c'_n)^{0.75} / \textstyle\sum_{c' \in V} (\#(c')^{0.75}), \quad (2)$$

where V is the word and audio-word vocabulary and $\#(c')$ is the global frequency of a word and audio-word c' in the whole dataset. This is for maximizing the distance between the target word and common words, such as "I" and "You", that occur frequently in the dataset.

In our current implementation, we set the vector dimension D to 400, the number of samplings for context words and context audio-words in each song to 400, and the number of negative samplings to 10. The objective function E_1 is optimized using stochastic gradient descent with a learning rate of 0.025, and training was run for 5 epochs.

We can also use the same idea to handle the similarities related to song audio tracks and artists. For song audio tracks, we define and minimize the loss function E_2:

$$E_2 = -\log\sigma\Big(\boldsymbol{u}(c)^\top \cdot \tfrac{1}{M}\textstyle\sum_{m=1}^{M} \boldsymbol{v_s}(aw_m)\Big) - \textstyle\sum_{n=1}^{N} \log\sigma\Big(-\boldsymbol{u}(c'_n)^\top \cdot \tfrac{1}{M}\textstyle\sum_{m=1}^{M} \boldsymbol{v_s}(aw_m)\Big), \quad (3)$$

where M is the number of audio-words in the song, $\boldsymbol{v_s}(\cdot)$ is an embedding function that maps the one-hot representation of every audio-word in the song to a D-dimensional vector, and the vector of the target song audio is represented by averaging all the audio-word vectors $\boldsymbol{v_s}(aw_m)$ in the song. For artists, we define and minimize the loss function E_3:

$$E_3 = -\log\sigma\Big(\boldsymbol{u}(c)^\top \cdot \boldsymbol{v_a}(a)\Big) - \sum_{n=1}^{N} \log\sigma\Big(-\boldsymbol{u}(c'_n)^\top \cdot \boldsymbol{v_a}(a)\Big). \quad (4)$$

where $\boldsymbol{v_a}(\cdot)$ is an embedding function that maps the one-hot representation of the artist ID to a D-dimensional vector. In the training phase, these objective functions E_2 and E_3 are optimized with the same settings as E_1.

The proposed model can capture similarities between the *same aspects* by minimizing E_1, E_2, and E_3. In addition, iterative optimization of the three objective functions enables training of the similarity between *multiple aspects* because these three objective functions share the embedding function for the context vector $\boldsymbol{u}(\cdot)$. In Figure 5, the target vectors of "snow", "Jazz song audio", and "Jazz artist" get closer to the context vectors such as $\boldsymbol{u}$(white); thus these multiple aspects can be located near each other.

3.4 Similarity Calculation

When calculating similarity, we use vectors obtained using $\boldsymbol{v_w}(\cdot)$, $\boldsymbol{v_s}(\cdot)$, and $\boldsymbol{v_a}(\cdot)$ without $\boldsymbol{u}(\cdot)$. We can use the cosine similarity as a measure of the similarity of two vectors. According to Levy et al. [16], multiple aspect vectors are normalized to unit length before they are used for similarity calculation, making cosine similarity and dot product equivalent. By this constraint, all the words, songs, and artists are located on a hypersphere and the system finds music content considering two flavors by calculating the content close to the blended vector on the hypersphere (Figure 7).

4. QUALITATIVE ANALYSIS

We investigated whether the trained vector space model appropriately captures the similarity between multiple musical aspects. Table 1 shows the five most similar lyric words, song audio tracks [1] , and artists – as well as their cosine similarities – that were obtained when we issued three different queries written at the top.

We can see in Table 1 that the query word "death" is similar to the words "blood" and "dead", which is reasonable since those words are often used in metal songs. Moreover, "death" is similar to song audio tracks having aggressive sounds using electric guitars and to Heavy Metal artists such as "Savatage". Metal songs like "Neuro Osmosis" were found even though their lyrics do not include the word "death".

Table 1 also shows that the query song "Amazing Grace" is similar to clean words such as "meadow" and "lullaby" and to relaxation songs such as New Age and Holiday music. Interestingly, the query artist "Michael Jackson" is similar to the artist "Janet Jackson" who is his sister even though the model does not know that they are siblings. This query is also similar to rhythmic songs by other artists.

These results indicate that the multi-aspect vector space model makes it possible to find related lyric words, songs, and artists that are hard to find otherwise. Furthermore, we tried to issue various queries by blending and subtracting flavors of multiple aspects and confirmed the usefulness of this blending and subtracting. Some results were illustrated in Section 2.2. Another interesting result is that the blended flavor "Stevie Wonder" + "snow" − "spring" is similar to "California Christmas".

5. QUANTITATIVE EVALUATION

Since ground-truth annotations of similarities between different aspects do not exist, it is not easy to quantitatively evaluate the model in general, but we tried to evaluate the effectiveness of blending latent vector representations by comparing content-based distributions of retrieved results. In Table 1, for example, the results retrieved for the word "death" and the song "Amazing Grace" are contrasting and have different impressions, which means that the distance between their distributions is large. If we issue a query blending them and it works effectively, we expect to see somewhat intermediate results between them, which means that the distance between one of the above distributions and the new distribution after blending becomes smaller.

[1] The song audio tracks used in this table can be listened to at our demo page (`https://kentow.github.io/qbb/`).

	Input word: death		Input song: Amazing Grace		Input artist: Michael Jackson	
Similar words	blood	0.70	may	0.37	hypnotized	0.41
	dead	0.65	sadly	0.35	maurice	0.40
	flesh	0.65	gentle	0.34	babygirl	0.39
	reborn	0.63	meadow	0.34	confused	0.39
	mortal	0.62	lullaby	0.34	emotions	0.39
Similar songs	Burning Sermon (Rock)	0.49	I Wish I Was In England (World)	0.85	It's a Man's Man's Man's World (Pop)	0.50
	Neuro Osmosis (Metal)	0.48	North Country Maid (Pop)	0.84	Tight (Gospel)	0.49
	Scraping the Barrel (Metal)	0.48	Mary, Did You Know (Holiday)	0.83	Play with Bootsy (Rock)	0.48
	Coins Upon the Eyes (Metal)	0.47	No Turning Back (Alternative)	0.83	Unspeakable (Pop)	0.48
	The Harlot Ov the Saints (Rock)	0.47	Beloved (NewAge)	0.83	Dead Heat (Pop)	0.48
Similar artists	Gary Numan	0.40	Barbra Streisand	0.40	Janet Jackson	0.61
	L'Âme Immortelle	0.39	Ella Fitzgerald	0.39	Sarah Connor	0.52
	Cowboy Junkies	0.38	Linda Ronstadt	0.38	Luther Vandross	0.52
	Don Moen	0.38	Debby Boone	0.37	Kylie Minogue	0.52
	Savatage	0.37	Nana Mouskouri	0.35	Faith Evans	0.51

Table 1. The most similar words, songs, and artists obtained by Query-by-Blending. The genre tags are shown in parentheses.

We therefore quantitatively examined how much this distance decreases after blending as follows. (1) A word query w and a song query s for evaluation are sampled from the dataset. (2) We get the 100 most similar songs retrieved by the word query w and compute ψ_w that denotes a content-based distribution of all lyric words and audio-words (i.e., a histogram of them) appearing in those 100 songs. (3) We get the 100 most similar songs retrieved for the song query s and compute ψ_s that denotes a content-based distribution obtained from those 100 songs in the same way. (4) We then get the 100 most similar songs retrieved for the query blending w and s and compute ψ_{w+s} that denotes a content-based distribution obtained from those 100 songs in the same way. (5) We calculate the Jensen-Shannon (JS) divergence between every pair of distributions: (ψ_w, ψ_s), (ψ_w, ψ_{w+s}), and (ψ_s, ψ_{w+s}). If the JS divergences of (ψ_w, ψ_{w+s}) and (ψ_s, ψ_{w+s}) are smaller than that of (ψ_w, ψ_s), we can confirm that our query blending lyric word and song audio works effectively. In addition, we replace the song s with the artist a and repeat the above procedure.

[Experimental Setup] For the above step (1), we used 1,000 word queries (most frequent nouns, verbs, adjectives, and adverbs), 1,000 song queries, and 1,000 artist queries. For the step (4), we made one million word-song pairs and one million word-artist pairs for the blended queries.

[Results] Table 2 shows JS divergences that are averaged over pairs. We can see that, as expected, the JS divergences of (ψ_w, ψ_{w+s}) and (ψ_s, ψ_{w+s}) are smaller than the JS divergence of (ψ_w, ψ_s). The same applies to (ψ_w, ψ_a). We thus confirmed that our blended queries work effectively with regard to content-based distributions of retrieved results.

Distributions		JS divergence	Distributions		JS divergence
ψ_w	ψ_s	0.261	ψ_w	ψ_a	0.224
ψ_w	ψ_{w+s}	**0.057**	ψ_w	ψ_{w+a}	**0.147**
ψ_s	ψ_{w+s}	**0.227**	ψ_a	ψ_{w+a}	**0.119**

Table 2. Quantitative evaluation of blending flavors.

6. RELATED WORK

Several studies have dealt with the similarity between different aspects of music. McFee and Lanckriet [22], for example, developed a hypergraph of song nodes whose edges capture multi-aspect relationships. Although it can be used to calculate similarities between songs while considering multiple aspects, it does not deal with similarities between the multiple musical aspects.

Some studies embedded musical aspects into a high-dimensional vector space in an unsupervised manner. Weston et al. [35] proposed a model by which musical audio signals and artist tags are embedded assuming that songs created by the same artist are correlated. Wang et al. [34] modeled the relationships between songs by using the same architecture as word2vec under the assumption that songs played by the same listener are similar. There are also studies that embedded vectorized audio-words and social tags by using the singular value decomposition (SVD) [13, 20]. All these studies shared with ours the motivation of embedding multiple aspects into a vector space but dealt only with audio signals and metadata without lyrics even though lyrics are an important element that conveys messages and emotions of music. To the best of our knowledge, there is no study in which lyric word, song audio, and artist ID are embedded into the same vector space.

7. CONCLUSION

In this paper we proposed a novel interface, Query-by-Blending, that enables users to find unfamiliar but interesting music content by flexibly combining (blending) three musical aspects: lyric word, song audio, and artist. Our contributions are summarized as follows: (1) Query-by-Blending is the first interface that lets users iteratively issue various queries by blending and subtracting multiple musical aspects. (2) We developed the novel embedding method of constructing the unified latent multi-aspect vector space by using unsupervised learning. (3) We demonstrated that our vector space model captures the similarities between multiple aspects. We plan to conduct a user study evaluating the Query-by-Blending interface. We also plan to extend our model to blend other aspects, such as genre tags and album cover images.

8. ACKNOWLEDGMENTS

The authors appreciate SyncPower Corporation for providing lyrics data. This work was supported in part by JST ACCEL Grant Number JPMJAC1602, Japan.

9. REFERENCES

[1] Alo Allik, Florian Thalmann, and Mark B. Sandler. Musiclynx: Exploring music through artist similarity graphs. In *Proceedings of the Web Conference 2018 (WWW)*, pages 167–170, 2018.

[2] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM SIAM symposium on Discrete algorithms*, pages 1027–1035, 2007.

[3] David Bretherton, Daniel A. Smith, Monica M. C. Schraefel, Richard Polfreman, Mark Everist, Jeanice Brooks, and Joe Lambert. Integrating musicology's heterogeneous data sources for better exploration. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, pages 27–32, 2009.

[4] Michael Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.

[5] Roger B. Dannenberg and Ning Hu. Understanding search performance in query-by-humming systems. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, 2004.

[6] Daniel P. W. Ellis, Brian Whitman, Adam Berenzweig, and Steve Lawrence. The quest for ground truth in musical artist similarity. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR)*, 2002.

[7] Jonathan Foote, Matthew L. Cooper, and Unjung Nam. Audio retrieval by rhythmic similarity. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR)*, pages 265–266, 2002.

[8] Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C. Smith. Query by humming: Musical information retrieval in an audio database. In *Proceedings of the 3rd ACM International Conference on Multimedia (ACM Multimedia)*, pages 231–236, 1995.

[9] Masataka Goto and Takayuki Goto. Musicream: Integrated music-listening interface for active, flexible, and unexpected encounters with musical pieces. *Journal of Information Processing*, 17:292–305, 2009.

[10] Zellig S Harris. Distributional structure. *Word*, 10(2–3):146–162, 1954.

[11] Carles Fernandes Julià and Sergi Jordà. Songexplorer: A tabletop application for exploring large collections of songs. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, pages 675–680, 2009.

[12] Tetsuya Kageyama, Kazuhiro Mochizuki, and Yosuke Takashima. Melody retrieval with humming. In *Proceedings of the 1993 International Computer Music Conference (ICMC)*, pages 349–351, 1993.

[13] Giannis Karamanolakis, Elias Iosif, Athanasia Zlatintsi, Aggelos Pikrakis, and Alexandros Potamianos. Audio-based distributional representations of meaning using a fusion of feature encodings. In *Proceedings of the 17th Annual Conference of the International Speech Communication Association (Interspeech)*, pages 3658–3662, 2016.

[14] Joon Hee Kim, Brian Tomasik, and Douglas Turnbull. Using artist similarity to propagate semantic information. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, pages 375–380, 2009.

[15] Peter Knees, Tim Pohle, Markus Schedl, and Gerhard Widmer. A music search engine built upon audio-based and web-based similarity measures. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 447–454, 2007.

[16] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics (TACL)*, 3:211–225, 2015.

[17] Tao Li and Mitsunori Ogihara. Content-based music similarity search and emotion detection. In *Proceedings of the 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 705–708, 2004.

[18] Yang Liu, Wan-Lei Zhao, Chong-Wah Ngo, Chang-Sheng Xu, and Han-Qing Lu. Coherent bag-of audio words model for efficient large-scale video copy detection. In *Proceedings of the ACM International Conference on Image and Video Retrieval (ACM CIVR)*, pages 89–96, 2010.

[19] Beth Logan and Ariel Salomon. A music similarity function based on signal analysis. In *Proceedings of the 2001 IEEE International Conference on Multimedia and Expo (ICME)*, 2001.

[20] Alessandro Lopopolo and Emiel van Miltenburg. Sound-based distributional models. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS)*, pages 70–75, 2015.

[21] Dominik Lübbers and Matthias Jarke. Adaptive multimodal exploration of music collections. In *Proceedings*

of the 10th International Society for Music Information Retrieval Conference (ISMIR), pages 195–200, 2009.

[22] Brian McFee and Gert RG Lanckriet. Hypergraph models of playlist dialects. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 343–348, 2012.

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*, pages 3111–3119, 2013.

[24] Emilio Molina, Lorenzo J. Tardón, Isabel Barbancho, and Ana M. Barbancho. The importance of F0 tracking in query-by-singing-humming. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, pages 277–282, 2014.

[25] Elias Pampalk, Simon Dixon, and Gerhard Widmer. Exploring music collections by browsing different views. In *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR)*, 2003.

[26] Elias Pampalk and Masataka Goto. MusicRainbow: A new user interface to discover artists using audio-based similarity and web-based labeling. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, pages 367–370, 2006.

[27] Markus Schedl, Peter Knees, and Gerhard Widmer. Discovering and visualizing prototypical artists by web-based co-occurrence analysis. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, pages 21–28, 2005.

[28] Malcolm Slaney and William White. Similarity based on rating data. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, pages 479–484, 2007.

[29] Tomonari Sonoda, Masataka Goto, and Yoichi Muraoka. A www-based melody retrieval system. In *Proceedings of the 1998 International Computer Music Conference, (ICMC)*, 1998.

[30] Sebastian Stober and Andreas Nürnberger. Musicgalaxy – an adaptive user-interface for exploratory music retrieval. In *Proceedings of 7th Sound and Music Computing Conference (SMC)*, pages 23–30, 2011.

[31] Kosetsu Tsukuda, Keisuke Ishida, and Masataka Goto. Lyric jumper: A lyrics-based music exploratory web service by modeling lyrics generative process. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 544–551, 2017.

[32] Fabio Vignoli, i Rob van Gulik, and Huub van de Wetering. Mapping music in the palm of your hand, explore and discover your collection. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, 2004.

[33] Chung-Che Wang, Jyh-Shing Roger Jang, and Wennen Wang. An improved query by singing/humming system using melody and lyrics information. In *Proceedings of the 11th International Society for Music Information Retrieval Conference, (ISMIR)*, pages 45–50, 2010.

[34] Dongjing Wang, Shuiguang Deng, Xin Zhang, and Guandong Xu. Learning music embedding with metadata for context aware recommendation. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval (ACM ICMR)*, pages 249–253, 2016.

[35] Jason Weston, Samy Bengio, and Philippe Hamel. Multi-tasking with joint semantic spaces for large-scale music annotation and retrieval. *Journal of New Music Research*, 40(4):337–348, 2011.

IMPROVING STRUCTURE EVALUATION THROUGH AUTOMATIC HIERARCHY EXPANSION

Brian McFee
Music and Audio Research Lab &
Center for Data Science
New York University
brian.mcfee@nyu.edu

Katherine M. Kinnaird
Department of Computer Science &
Statistical and Data Sciences Program
Smith College
kkinnaird@smith.edu

ABSTRACT

Structural segmentation is the task of partitioning a recording into non-overlapping time intervals, and labeling each segment with an identifying marker such as A, B, or *verse*. Hierarchical structure annotation expands this idea to allow an annotator to segment a song with multiple levels of granularity. While there has been recent progress in developing evaluation criteria for comparing two hierarchical annotations of the same recording, the existing methods have known deficiencies when dealing with inexact label matchings and sequential label repetition.

In this article, we investigate methods for automatically enhancing structural annotations by inferring (and expanding) hierarchical information from the segment labels. The proposed method complements existing techniques for comparing hierarchical structural annotations by coarsening or refining labels with variation markers to either collapse similarly labeled segments together, or separate identically labeled segments from each other. Using the multi-level structure annotations provided in the SALAMI dataset, we demonstrate that automatic hierarchy expansion allows structure comparison methods to more accurately assess similarity between annotations.

1. INTRODUCTION

In the music information retrieval (MIR) literature, the problem of *musical structure analysis* broadly concerns methods for automatically inferring relationships between moments in time within a piece [1, 10]. Substantial effort has been expended to develop computational techniques to infer various structures in recorded music, and the existence of reliable reference data and evaluation methodology is critical to accurately assess the efficacy of these methods. More broadly, reliable methods for comparing interpretations of musical structure can be informative for understanding human perception of music [13, 14].

Musical structure can be represented in a variety of ways, depending on the intended use case, ranging from (symbolic) staff notation, to chord annotations, lead sheets, *etc*. MIR research typically focuses on the *segmentation* problem, where the time extent of a recording is partitioned, and each partition is labeled with a descriptor that can be used to indicate repetitions, such as A, B, A, C or *verse, chorus, verse, bridge*. Much of the computational work in this area models musical structure as *flat*, meaning that there is exactly one partitioning of the piece, and the elements of the partition (*segments*) cannot be merged or subdivided to form larger or smaller structures.

In contrast, there is a long tradition in music theory of modeling music with *hierarchies* that simultaneously represent structure at multiple levels of granularity [2, 3]. Indeed, even when instructed to produce a flat segmentation of a piece, expert annotators will often encode latent hierarchical information by using variation markers in their segment labels, *e.g.*, $A, \ldots, A'$ or *verse, ..., verse_(instrumental)* [9, 12]. Although an annotator's choice of segment label may clearly be informative in these cases, this information is ignored by standard segmentation comparison methods. This owes, primarily, to an inability to directly support multi-level segmentations, which could be used to simultaneously represent both the original and simplified annotation as a coherent structure.

In recent years, there has been increasing interest and progress in developing datasets [8, 12], computational methods [16], representations [7], and evaluation criteria [6] for hierarchically structured music segmentations. However, little attention has been paid to exposing *latent* hierarchical structure encoded by segment labels for use in conjunction with these methods.

1.1 Our contributions

In this work, we develop a method for automatically exposing latent multi-level structure encoded by label similarity in music segmentations. The proposed *automatic hierarchy expansion* method operates by simultaneously contracting similar (but distinct) segment labels, and refining identically labeled (but distinct) segments. The contraction and refinement are combined with the original annotation into a hierarchical annotation, which can then be compared to other hierarchies using existing techniques.

Using the SALAMI dataset as a test case, we demonstrate that the proposed method is effective at identifying similarities across annotations that are not captured by previous methods. Finally, we leverage insights gained in developing the method to explore issues of internal consistency within multi-level structure annotations.

1.2 Preliminaries

For a signal of duration T, we define a (flat) *segmentation* as a function $S : [0, T] \rightarrow V$ where V denotes a set of segment labels, e.g., $V = \{A, B, \ldots\}$. We define a *multi-level segmentation* (or *hierarchy*) as a sequence of segmentations $H = (S_0, S_1, \ldots)$, where S_0 maps to a single label, and subsequent segmentations S_i are ordered from *coarse* to *fine*. We assume that each segmentation S_i maps to a distinct vocabulary. Finally, we say that a hierarchy H is *monotonic* if for every level k, we have that

$$S_k(u) = S_k(v) \quad \Rightarrow \quad S_{k-1}(u) = S_{k-1}(v). \quad (1)$$

2. RELATED WORK

Methods for evaluating musical structure analysis algorithms, or more generally comparing two different (flat) structural annotations, broadly fall into two categories: boundary detection and label agreement. Boundary detection metrics capture the agreement between annotations in localizing moments of time when the piece transitions from one coherent segment to another [15]. Segment boundaries are entirely local phenomena, and the metrics do not attempt to encode any sense of long-term structure in the annotations.

Label agreement metrics, on the other hand, are globally informed, and derive from comparisons between the segment labels applied to short samples, typically 0.1s in duration. The *pairwise* precision metric [4] is defined by determining which time points i and j are both given the same label in the reference annotation, and checking to see if the estimate annotation also gives both time points the same label; the fraction of such pairs of time points determines a precision score. Exchanging the roles of the reference and estimate annotations yields definitions for recall and F_1-score. Similarly, the *normalized conditional entropy* measures [5] quantify agreement in terms of the mutual information between the two annotations.

Although label agreement metrics account for global structure, they have three notable shortcomings. First, they are sensitive to alignment errors: if two annotators are operating at different levels of granularity, this information can be obscured by the evaluation. Hierarchical structure evaluation measures address this by integrating multiple segmentations at different levels of granularity into a single hierarchy, and comparing hierarchies to one another [6].

Second, since they depend on frames in isolation from their surrounding context, label agreement metrics cannot distinguish a long segment A from two short segments aa that cover the same time extent. Typically, practitioners circumvent this issue by reporting boundary detection metrics as well as label agreement metrics, but the interactions between the two types of score are rarely easy to interpret.

Finally, label agreement metrics have no mechanism to exploit similarity encoded within segment *labels*: labels A and A' are considered equally distinct as A and B, even though the annotator is clearly implying some high-level similarity in the first case that is absent in the second. While one could modify such annotations directly and use a flat segment evaluation metric, doing so discards information that could still be useful for comparison purposes.

In this work, we address these three issues by deriving segment hierarchies which are informed by label similarity. Discarding variation markers ($A' \rightarrow A$) allows the evaluation to recover from superficially distinct segment labels, while adding counters ($aa \rightarrow a_0 a_1$) provides a way to distinguish a long segment from sequential repetitions of a short segment. By integrating these two modifications into a hierarchy, along with the original annotation, we preserve all of the information present in the annotation.

2.1 Hierarchical evaluation

The approach taken in this work is based on the L-measure method for multi-level segmentation comparison [6], summarized here for completeness. Given a hierarchy H, a *meet matrix* M is defined by the maximum level at which every pair of time instants (u, v) receive the same label:

$$M[u, v] := \max \{k \mid S_k(u) = S_k(v)\}. \quad (2)$$

The meet matrix induces a partial ordering over pairs of time instants, which is summarized by a set of triplets:

$$A(H) := \{(t, u, v) \mid M[t, u] > M[t, v]\}. \quad (3)$$

Finally, given two hierarchies H^R (the reference) and H^E (the estimate), a precision score is defined by comparing the two triplet sets:

$$\text{L-Precision}\left(H^R, H^E\right) := \frac{\left|A\left(H^R\right) \cap A\left(H^E\right)\right|}{\left|A\left(H^E\right)\right|}. \quad (4)$$

Recall is defined analogously by reversing the roles of reference and estimate, and an F_1-score (hereafter denoted as L-Measure) can be computed by the harmonic mean of precision and recall. The terms *reference* and *estimate* to distinguish between annotations derive from the method's use in comparing algorithm outputs to manual annotations. However, the method can generally compare between different annotations of equal status, *e.g.*, produced by two different human annotators. In this case, the terms *reference* and *estimate* are merely intended to identify the annotators, but not to confer privileged status to either.

Although defined for multi-level segmentations, the L-measure can also be applied to compare flat segmentations by including a vacuous segment S_0 which produces a single segment spanning the entire duration. This results in an evaluation which is similar to the pairwise frame similarity metric [4], differing only in that it compares triples rather than pairs. For consistency across experiments, we will employ this method (with the S_0 segment) when comparing flat segmentations in the remainder of this article.

3. METHODS

This work proposes a method for expanding flat annotations to include both more nuanced and coarser structural information. Such expansions seek to address the three shortcomings of label agreement metrics detailed in Section 2. We also explore the concept of *monotonicity* within the hierarchy resulting from applying our methods to several levels of flat annotations.

3.1 Automatic Hierarchy Expansion

Here, we propose an automatic hierarchical expansion for any 'flat' annotations. Our method expands a flat annotation into a hierarchy with three levels. The first level is a *contraction* of the variation markers. The second level is the original annotation. The third level is a *refinement* of the labels by making each instance of a label unique by adding counters to the label.

For a concrete example, consider Figure 1. The left part of the image shows the flat annotation which is repeated on the right side of the image as the middle level of the hierarchy. The contraction level, shown in green, removed the variation markers of the A repetition. The result is that the contraction part of the hierarchy has two kinds of repetitions instead of three.

The refined level of the hierarchy, shown in blue, has at most one block per line. For clarity, the refinement level is created directly from the contraction level of the hierarchy. For each instance of a label in the contraction level, we append a counter (starting with 0) to form a new label. If instead we had conducted this refinement starting at the middle level, we would have ended up with the annotation labels $\{A0, A'0, B0, B1, B2\}$ instead of $\{A0, A1, B0, B1, B2\}$. Both methods produce equivalent results, but the latter is easier to interpret.

Although the expansion described above is most easily understood when applied to flat inputs, it can also be applied to hierarchical inputs by expanding each level independently and combining the results. An example of this multi-level hierarchy expansion is given in Figure 2. We also note that whenever a contraction (or refinement) leaves the segmentation unchanged, the redundant level is omitted from the expanded hierarchy as it produces no additional content. Note that the expansions of the upper and lower annotations in Figure 2 only have two levels each; this is due to the lack of variation markers within the original annotations, meaning that there is nothing to contract.

3.2 Monotonicity

The L-measure described in section 2.1 hinges upon the definition of the meet matrix M (see eq. (2)), which can be interpreted as measuring the similarity between two time points by the depth in the hierarchy at which they receive the same label. When expanding a flat segmentation S, the result is guaranteed to be a monotonic hierarchy. If $S(u) \neq S(v)$, then the contraction level *may* assign u and v the same label, but the refinement level will not. Conversely,

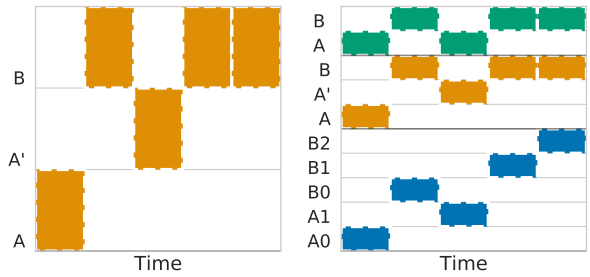


Figure 1. An example of automatic hierarchy expansion. A flat segmentation (left) with segments (A, B, A', B, B) is expanded into a three-level hierarchy (right). The contraction level (green, top) removes variation markers, while the refinement level (blue, bottom) adds counters to each instance of a segment label. The center level (orange) preserves the original annotation.

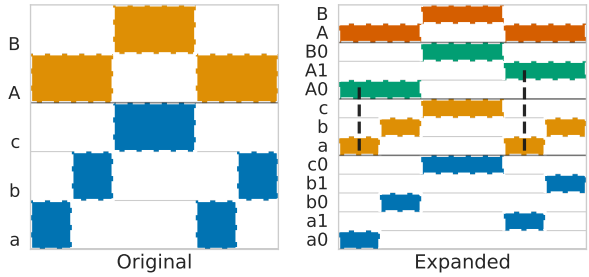


Figure 2. Automatic hierarchy expansion is not guaranteed to preserve monotonicity when applied independently to each level of a hierarchy (left). The dashed lines indicate two instants which receive the same label a in one level, but different labels ($A0$ and $A1$) in a preceding level.

if $S(u) = S(v)$, then the contraction level *must* preserve this equivalence, while the refinement level may not.

However, when applying automatic expansion independently to each level of a hierarchical annotation, the result may not be monotonic. Figure 2 illustrates this effect, where the refinement of the upper level (orange, left plot) results in violations of monotonicity, indicated by the dashed lines in the right plot.

While one could preserve monotonicity by only contracting at the highest level and refining at the lowest level, this is undesirable for three reasons. First, there may be informative structure encoded by variation markers in the intermediate levels which would be missed. Second, annotators may not be internally consistent (*i.e.*, monotonic) from upper to lower-level, so monotonicity would be violated from the start. And finally, because the L-measure depends only on the maximum level of agreement, it does not strictly *require* monotonicity to operate, though the results may be somewhat counter-intuitive. Still, the L-measure definition is most intuitive when the underlying annotations are monotonic, so it is worth investigating the effects of hierarchy expansion on monotonicity.

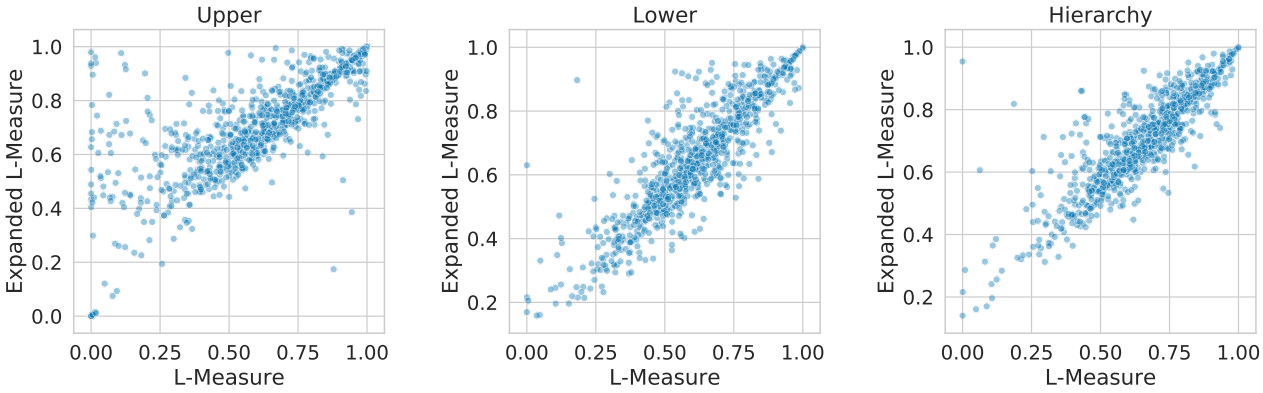


Figure 3. L-measure applied to pairs of annotations in the SALAMI dataset before and after automatic expansion. Left: upper-level annotations; middle: lower-level annotations; right: hierarchical annotations.

4. EXPERIMENTS

We evaluated the effect of automatic hierarchy expansion on flat annotations in the SALAMI dataset [12]. This dataset was selected for two reasons. First, it contains multiple reference annotations (by different annotators). Second, each annotation includes segmentations at different levels of granularity (*upper* and *lower*), which can be treated separately or combined into one hierarchy. All experiments were conducted using the L-measure implementation included in `mir_eval` version 0.5 [11].

4.1 Expansion on flat annotations

For each SALAMI track with two annotators, we first computed the L-measure between the two *upper* annotations. Because neither annotator has a privileged status as reference, we computed the "L-measure" as the harmonic mean of L-precision and L-recall. We then applied the hierarchy expansion procedure to each annotation, and the recomputed the L-measure on the expansions. Comparing the L-measure before and after expansion of a single level allows us to quantify the amount of structural similarity implicitly coded in the segment labels. This process was then repeated for the *lower*-level annotations.

Figure 3 (left and middle plots) summarizes the results of this experiment. As a general trend, expansion has substantial impact on the *upper* level, and less impact on the *lower* level. More specifically, expansion of the upper level produces a change in L-measure of 0.107 ± 0.168 (mean $\pm$ standard deviation), while expansion of the lower-level produces a change of 0.038 ± 0.09. The trend is generally positive at the upper level (with a few exceptions), while the lower-level changes are more symmetric.

Figure 4 illustrates two extreme cases where automatic hierarchy expansion dramatically changes the L-measure between *upper* annotations. [1] In the first case, track 242 improves from 0 to 0.979, because the refinement of the second annotation ($AA \rightarrow A_0A_1$) agrees with the first annotation (AA'), and the contraction of the first annotation

[1] Qualitatively similar examples can be observed for the *lower* annotations, which are omitted here for brevity.

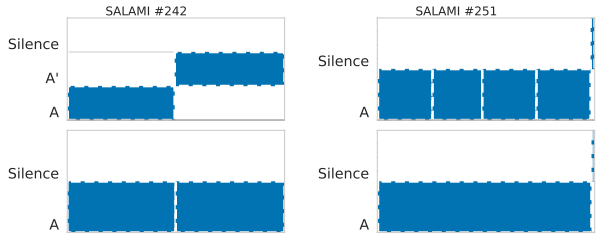


Figure 4. Two extreme examples where hierarchy expansion changes the L-measure from flat (upper) annotations. Left: track 242 increases by $+0.979$; right: track 251 decreases by -0.705.

($AA' \rightarrow AA$) matches with the second annotation. These annotations effectively encode the same information, differing only in the use of variation markers. The second case, track 251, decreased from 0.879 to 0.174 after expansion. This is explained by the first annotation explicitly marking repeated A sections, which are refined into unique sections $(A0, A1, A2, A3)$ by expansion. This structure is absent from the second annotation, which covers the entire duration by a single A segment. Prior to expansion, label-agreement metrics over-estimate the similarity between these annotations. Hierarchy expansion exposes this oversight, resulting in a more accurate comparison.

4.2 Expansion on hierarchical annotations

Extending the analysis of the previous section, we combined each annotator's upper and lower segmentations into a hierarchical annotation H. We then applied hierarchy expansion to each level of the hierarchy, resulting in a new hierarchy H^*. Finally, we computed the L-measure between pairs of hierarchies before and after expansion, which provides a more holistic view of how expansion affects measured agreement between annotators.

The results of expansion comparison for hierarchies are summarized in Figure 3 (right). Overall, the differences are qualitatively similar to the *lower*-level comparison, producing differences in L-measure of 0.048 ± 0.090. While

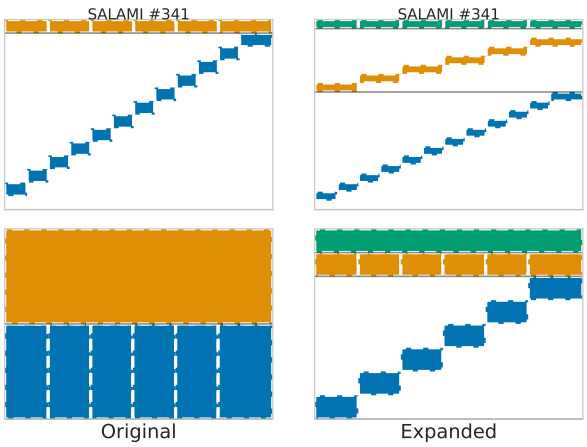


Figure 5. Automatic expansion significantly improves L-measure agreement between two hierarchical annotations of track 341 (increase of $+0.954$). Left: the original hierarchies; right: the expanded hierarchies. Segment labels are suppressed to enhance legibility.

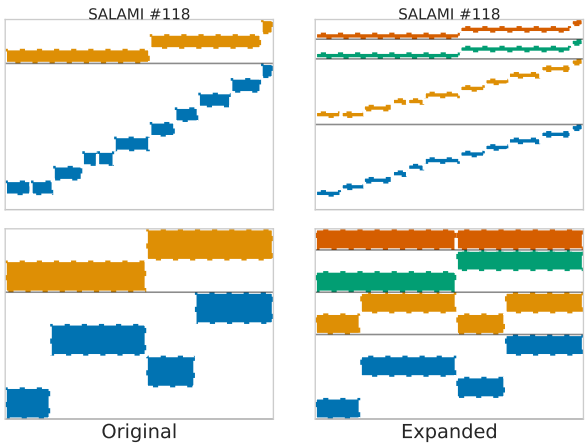


Figure 6. An example where automatic expansion significantly reduces L-measure agreement between two hierarchical annotations (decrease of -0.223). Left: the original hierarchies; right: the expanded hierarchies.

less dramatic than the *upper*-level comparison, the trend is still generally positive, with over 77% of comparisons increasing in value after applying hierarchy expansion. This indicates that even when evaluating with hierarchical annotations, there is still some latent structure encoded in the segment labels which current methods do not account for.

Figure 5 illustrates an example where expansion increases L-measure between two hierarchies, from 0 to 0.954. The second hierarchy (bottom left) assigns the same label to each segment, though the lower level is divided in to repetitions. Expansion separates these repeated segments in both annotations, exposing the common structure shared by both hierarchies (right two subplots).

Figure 6 illustrates the opposite case, where expansion exposes disagreement, decreasing score from 0.841 to 0.618. In this case, looking only at the upper level of the two annotations (left plots, orange level) would indicate considerable agreement between the two annotations, though the lower levels (blue) diverge significantly.

These selected examples are the extreme cases where L-measures deviated the most after hierarchy expansion. In both cases, we find that the divergence can be easily explained by visual inspection, which validates that hierarchy expansion behaves as expected. Note that these cases are relatively unusual, and most changes in scores are much smaller in magnitude. We therefore conclude that hierarchy expansion is effective at recovering from exceptional cases while not detrimentally affecting the common cases.

4.3 Quantifying monotonicity

The experiment described in Section 4.1 started with flat segmentations, and is therefore guaranteed to produce monotonic hierarchies. As noted in Section 3.2, this is not generally true when expanding hierarchies. This raises the question of the importance of monotonicity on hierarchical segmentation evaluation, and whether a given annotator is

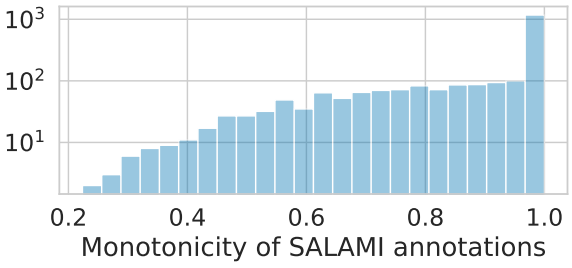


Figure 7. The distribution of monotonicity scores across all segment hierarchies in SALAMI (median: 0.98).

internally consistent between upper and lower levels.

The definition of monotonicity given in eq. (1) is binary, but it can be relaxed by instead measuring the *proportion* of time instants u and v where agreement at level k implies agreement at level $k-1$. This is calculated exactly by the pairwise recall measure [4], when S_k is treated as the reference and S_{k-1} is the estimate. We thus computed pairwise recall between lower and upper segmentations for each annotation: measures close to 1 are highly monotonic, and lower values indicate violations of monotonicity. The results are summarized by the distribution plot in Figure 7. Overall, the median monotonicity score across all annotations was 0.98, though there appears to be a heavy tail of non-monotonic annotations.

Looking more carefully into the data, we observed that a significant portion of monotonicity violations could be explained by the use of variation markers in the upper-level segmentation. These annotations are specifically problematic because the lower segment a may correspond to distinct upper labels A and A'. More than 60% of hierarchies that do not use variation markers in the upper level are perfectly monotonic, while only 27% of hierarchies with upper variations are monotonic.

Figure 8 illustrates the distribution of monotonicity

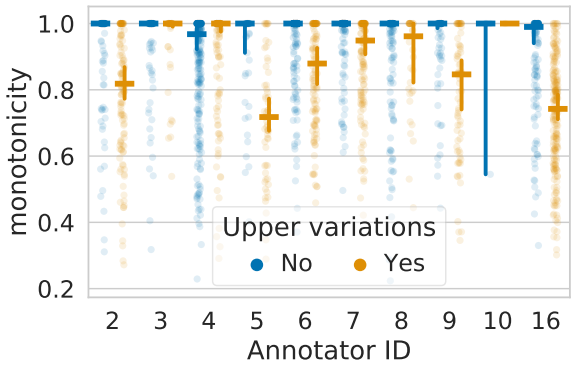


Figure 8. Monotonicity measurements for each SALAMI annotation, grouped by annotator. Median values and 95% bootstrap confidence intervals are indicated by bars.

scores for each individual annotator, sub-divided according to the presence or absence of variation markers in the upper level. Figure 8 shows that use of upper variation markers consistently coincide with lower monotonicity score.

In these experiments, we identified that variation markers can introduce unnecessary differences between sections. What is more, our investigations suggest that the use of variation markers coincides with reduction in monotonicity. The contraction level in the automatic hierarchy expansion seeks to address this issue. However, expanding multiple levels can introduce monotonicity violations. Combining the investigations in this section with the results from Section 4.2, we conclude any new violations created by the contraction of the lower level and the refinement of the upper levels are not substantially detrimental compared to the overall improvements conferred by automatic hierarchy expansion.

4.4 Permutation stability

It is natural to ask whether the previous results are due to introducing multiple hierarchical levels (independent of labels), or if the specific manner in which the contraction and refinement levels are constructed matters. If the effects of hierarchy expansion on L-measure are primarily due to additional levels, but not their specific label structure, we expect that expanding the segmentation with randomly permuted labels should produce comparable results.

To test this idea, we took inspiration from statistical permutation testing, and conducted the following experiment on each level of flat segmentations.

- For each annotation S, construct its hierarchy expansion H and compute the L-recall from S to H.
- (Repeat): randomly permute the labels of S to produce new flat segmentation P, and expand P to new hierarchy H^P. Compute L-recall from S to H^P.

We then compared the distribution of recall scores arising from the (S, H) comparisons to distribution arising from (S, H^P) comparisons. Since the expansion H contains S, the recall score will be identically 1.[2] Note that the expansion H^P will have equivalent refinement level to that of H because each segment is uniquely labeled, so the differences induced by permutation are confined to the middle and upper (contraction) levels.

Level	Mean (original)	Mean (permutation)	KS
Upper	0.992	0.603	0.940
Lower	0.996	0.468	0.977

Table 1. Results of the permutation-expansion test on upper- and lower-level segmentations. *KS* reports the 2-sample Kolmogorov-Smirnov test statistic between expansion and permuted expansion comparisons.

For each annotation, 20 independent permutations were generated. For each level, we report the mean L-recall over original expansions and permuted expansions. We then calculated the 2-sample Kolmogorov-Smirnov test statistic (KS) to determine if the two samples could plausibly be generated from the same underlying distribution. Table 1 summarizes the results of the experiment. In both cases, this null hypothesis was rejected with p-value numerically indistinguishable from 0, indicating that the effects of the hierarchy expansion on the L-measure depend on both the additional hierarchical information and its specific label structure.

5. CONCLUSION

The automatic hierarchy expansion method proposed in this article provides a flexible framework for retaining subtle differences in annotations, while simultaneously exposing coarse similarity. By leveraging ideas from hierarchical structure evaluation, the proposed method is able to illuminate detailed structure latent in the annotations, and recover from problematic edge cases not handled by previous methods. Moreover, the segment refinement technique provides a way to simultaneously evaluate segment boundaries and repetitions, which has been problematic for previous, frame-based evaluations.

Although our focus of the experiments in this work relies on structural segmentation labels in the SALAMI dataset, the general ideas may be applied more broadly, *e.g.*, to the functional section labels used in the Isophonics and TUT annotations [9]. Alternatively, this automatic hierarchy expansion could be applied to other musical concepts where hierarchies naturally occur, such as chord labels (*root, quality, extensions*) or instrumentation (*family, instrument, register*). This could provide a robust alternative evaluation technique for classification problems where adhering to a flat vocabulary is problematic, but where modeling full taxonomies might also be intractable.

[2] When S consists of a single segment spanning the entire duration, it will produce an L-recall of 0. However, we note that the permutation procedure on such annotations will have no effect.

6. REFERENCES

[1] Roger B Dannenberg and Masataka Goto. Music structure analysis from acoustic signals. In *Handbook of Signal Processing in Acoustics*, pages 305–331. Springer, 2008.

[2] Katherine M. Kinnaird. Aligned hierarchies: A multi-scale structure-based representation for music-based data streams. In *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*, pages 337–343, 2016.

[3] Fred Lerdahl and Ray S. Jackendoff. *A generative theory of tonal music*. MIT press, 1985.

[4] Mark Levy and Mark Sandler. Structural segmentation of musical audio by constrained clustering. *IEEE transactions on audio, speech, and language processing*, 16(2):318–326, 2008.

[5] Hanna M. Lukashevich. Towards quantitative measures of evaluating song segmentation. In *ISMIR 2008, 9th International Conference on Music Information Retrieval, Drexel University, Philadelphia, PA, USA, September 14-18, 2008*, pages 375–380, 2008.

[6] Brian McFee, Oriol Nieto, Morwaread M. Farbood, and Juan Pablo Bello. Evaluating hierarchical structure in music annotations. *Frontiers in Psychology*, 8:1337, 2017.

[7] Melissa R. McGuirl, Katherine M. Kinnaird, Claire Savard, and Erin H. Bugbee. SE and SNL diagrams: Flexible data structures for MIR. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, pages 341–347, 2018.

[8] Oriol Nieto and Juan Pablo Bello. Systematic exploration of computational music structure research. In *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*, pages 547–553, 2016.

[9] Jouni Paulus and Anssi Klapuri. Music structure analysis by finding repeated parts. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 59–68. ACM, 2006.

[10] Jouni Paulus, Meinard Müller, and Anssi Klapuri. State of the art report: Audio-based music structure analysis. In *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, Utrecht, Netherlands, August 9-13, 2010*, pages 625–636, 2010.

[11] Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P. W. Ellis. MIR_EVAL: A transparent implementation of common MIR metrics. In *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, pages 367–372, 2014.

[12] Jordan B. L. Smith, John Ashley Burgoyne, Ichiro Fujinaga, David De Roure, and J. Stephen Downie. Design and creation of a large-scale database of structural annotations. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*, pages 555–560, 2011.

[13] Jordan B. L. Smith and Elaine Chew. Automatic interpretation of music structure analyses: A validated technique for post-hoc estimation of the rationale for an annotation. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pages 435–441, 2017.

[14] Jordan B. L. Smith, Ching-Hua Chuan, and Elaine Chew. Audio properties of perceived boundaries in music. *IEEE Transactions on Multimedia*, 16(5):1219–1228, Aug 2014.

[15] Douglas Turnbull, Gert R. G. Lanckriet, Elias Pampalk, and Masataka Goto. A supervised approach for detecting boundaries in music using difference features and boosting. In *Proceedings of the 8th International Conference on Music Information Retrieval, ISMIR 2007, Vienna, Austria, September 23-27, 2007*, pages 51–54, 2007.

[16] Karen Ullrich, Jan Schlüter, and Thomas Grill. Boundary detection in music structure analysis using convolutional neural networks. In *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, pages 417–422, 2014.

CONDITIONED-U-NET: INTRODUCING A CONTROL MECHANISM IN THE U-NET FOR MULTIPLE SOURCE SEPARATIONS

Gabriel Meseguer-Brocal
Ircam Lab, CNRS, Sorbonne Université
75004 Paris, France
gabriel.meseguerbrocal@ircam.fr

Geoffroy Peeters
LTCI, Télécom Paris, Institut Polytechnique de Paris,
75013, Paris, France
geoffroy.peeters@telecom-paris.fr

ABSTRACT

Data-driven models for audio source separation such as U-Net or Wave-U-Net are usually models dedicated to and specifically trained for a single task, e.g. a particular instrument isolation. Training them for various tasks at once commonly results in worse performances than training them for a single specialized task. In this work, we introduce the Conditioned-U-Net (C-U-Net) which adds a control mechanism to the standard U-Net. The control mechanism allows us to train a unique and generic U-Net to perform the separation of various instruments. The C-U-Net decides the instrument to isolate according to a one-hot-encoding input vector. The input vector is embedded to obtain the parameters that control Feature-wise Linear Modulation (FiLM) layers. FiLM layers modify the U-Net feature maps in order to separate the desired instrument via affine transformations. The C-U-Net performs different instrument separations, all with a single model achieving the same performances as the dedicated ones at a lower cost.

1. INTRODUCTION

Generally, in Music Information Retrieval (MIR) we develop dedicated systems for specific tasks. Facing new (but similar) tasks require the development of new (but similar) specific systems. This is the case of data-driven music source separation systems. Source separation aims to isolate the different instruments that appear in an audio mixture (a mixed music track) i.e., reversing the mixing process. Data-driven methods use supervised learning where the mixture signals and the isolated instruments are available for training. The usual approach is to build dedicated models for each task to isolate [1, 19]. This has been proved to show great results. However, since isolating an instrument requires a specific system, we can easily run into problems such as scaling issues (100 instruments = 100 systems). Besides, these models do not use the commonalities between instruments. If we modify them to do various tasks at once i.e., adding more filters for the last layers and having fix numbers of outputs, they reduce their performance [19].

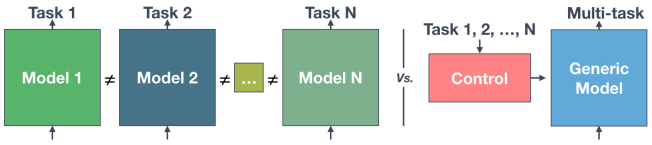


Figure 1: [Left part] Traditional approach: a dedicated U-Net is trained to separate a specific source. [Right part] Our proposition based on conditioning learning. The problem is divided in two: a standard U-Net (which provides generic source separation filters) and a control mechanism. This division allows the same model to deal with different tasks using the commonalities between them.

Conditioning learning has appeared as a solution to problems that need the integration of multiple resources of information. Concretely, when we want to process one in the context of another i.e., modulating a system computation by the presence of external data. Conditioning learning divides problems into two elements: a **generic system** and a **control mechanism** that governs it according to external data. Although there is a large diversity of domains that use it, it has been developed mainly in the image processing field for tasks such as visual reasoning or style transfer. There, it has been proved very effective, improving the state of the art results [3, 13, 20]. This paradigm can be integrated into source separation creating a generic model that adapts to isolate a particular instrument via a control mechanism. We also believe that this paradigm can benefit to a great diversity of MIR tasks such as multi-pitch estimation, music transcription or music generation.

In this work, we propose the application of conditioning learning for music source separation. Our system relies on a **standard U-Net system** not specialized in a specific task but rather in finding a set of generic source separation filters, that we **control** differently for isolating a particular instrument, as illustrated in Figure 1. Our system takes as input the spectrogram of the mixed audio signal and the control vector. It gives as output (only one) the separated instrument defined by the control vector. The main advantages of our approach are - direct use of commonalities between different instruments, - a constant number of parameters no matter how many instruments the system is dealing with - and scalable architecture, in the sense that

new instruments can be potentially added without training from scratch a new system. Our key contributions are:

1. the Conditioned-U-Net (C-U-Net), a joint model that changes its behavior depending on external data and performs for any task as good as a dedicated model trained for it. C-U-Net has a fixed number of parameters no matter the number of output sources.

2. The C-U-Net proves that conditioning learning (via Feature-wise Linear Modulation (FiLM) layers) is an efficient way of inserting external information to MIR problems.

3. A new FiLM layer that works as good as the original one but with a lower cost (fewer parameters).

2. RELATED WORK

We review only works related to conditioning in audio and to data-driven source separation methods.

Conditioning in audio. It has been mainly explored in **speech generation**. In the WaveNet approach [22, 23] the speaker identity is fed to a conditional distribution adding a learnable bias to the gated activation units. A WaveNet modified version is presented in [18]. The time-domain waveform generation is conditioned by a sequence of Mel spectrogram computed from an input character sequence (using a recurrent sequence-to-sequence network with attention). In **speech recognition** conditions are used in [10], applying conditional normalisation to a deep bidirectional LSTM (Long Short Term Memory) for dynamically generating the parameters in the normalisation layer. This model adapts itself to different acoustic scenarios. In [10], the conditions do not come from any external source but rather from utterance information of the model itself. They have been also used in **music generation** for accompaniments conditioned on melodies [6] or incorporating history information (melody and chords) from previous measures in a generative adversarial network (GAN) [25]. Finally, it has been also proved to be very efficient for **piano transcription** [5]: the pitch onset detection is internally concatenated to the frame-wise pitch prediction controlling if a new pitch starts or not. Both, onset detection and frame-wise prediction are trained together.

Source separation based on supervised learning. We refer the reader to [15] for an extensive overview of the different source separation techniques. We review only the data-driven approaches. Here, the neural networks have taken the lead. Although architectures such as RNN [7] or CNN [2] have been studied, the most successful one use a deep U-Net architecture (also called U-Net). In [1], the U-Net is applied to a spectrogram to separate the vocal and accompaniment components, training a specific model for each task. Since the output is the spectrogram, they need to reconstruct the audio signal which potentially leads to artifacts. For this reason, Wave-U-Net proposes to apply the U-Net to the audio-waveform [19]. They also adapt their model for isolating different sources at once by adding to their dedicated version as many outputs as sources to separate. However, this multi-instruments version performs worse than the dedicated one (for vocal isolation) and has to be retrained to different source combinations.

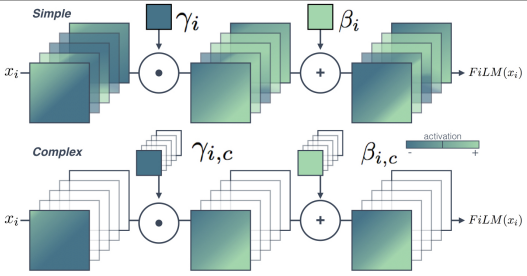


Figure 2: [Top part] *FiLM simple* layer applies the same affine transformation to all the input feature maps x. [Bottom part] In the *FiLM complex* layer, independent affine transformations are applied to each feature map c.

The closest work to ours is [9]. In there, they propose to use multi-channel audio as input to a Variational Auto-Encoder (VAE) to separate 4 different speakers. The VAE is conditioned on the ID of the speaker to be separated. The proposed method outperforms its baseline.

3. CONDITIONING LEARNING METHODOLOGY

3.1 Conditioning mechanism.

There are many ways to condition a network (see [4] for a wide overview) but most of them can be formalized as affine transformations denoted by the acronym FiLM (Feature-wise Linear Modulation) [13]. FiLM permits to modulate any neural network architecture inserting one or several FiLM layers at any depth of the original model. A FiLM layer conditions the network computation by applying an affine transformation to intermediate features:

$$FiLM(x) = \gamma(z) \cdot x + \beta(z) \quad (1)$$

where x is the input of the FiLM layer (i.e., the intermediate feature we want to modify), γ and β are parameters to be learned. They scale and shift x based on the external information, z. The output of a FiLM layer has the same dimension as the intermediate feature input x. FiLM layers can be inserted at any depth i in the controlled network.

As described in Figure 2, the original FiLM layer applies an independent affine transformation to each feature map c [1]: $\gamma_{i,c}$ and $\beta_{i,c}$ [13]. We call this a *FiLM complex* layer (**Co**). We propose a simpler version that applies the same γ_i and β_i to all the feature maps (therefore γ and β do not depend on c). We call it a *FiLM simple* layer (**Si**). The *FiLM simple* layer decreases the degrees of freedom of the transformations to be carried out forcing them to be generic and less specialized. It also reduces drastically the number of parameters to be trained. As FiLM layers do not change the shape of x, FiLM is transparent and can be used in any particular architecture providing flexibility to the network by adding a control mechanism.

[1] Or element-wise.

3.2 Conditioning architecture.

A conditioning architecture has two components:

The conditioned network. It is the network that carries out the core computation and obtains the final output. It is usually a generic network that we want to behave differently according to external data. Its behavior is altered by the condition parameters, $\gamma_{i,(c)}$ and $\beta_{i,(c)}$ via FiLM layers.

The control mechanism - condition generator. It is the system that produces the parameters (γ's and β's) for the FiLM layers with respect to the external information z: the input conditions. It codifies the task at hand and provides the instructions to control the conditioned network. The condition generator can be trained jointly [13, 20] or separately with the conditioned network [3].

This paradigm clearly separates the tasks description and control instructions from the main core computation.

4. CONDITIONED-U-NET FOR MULTITASK SOURCE SEPARATION

We formalize source separation as a multi-tasks problem where one task corresponds to the isolation of one instrument. We assume that while the tasks are different they share many similarities, hence they will benefit from a conditioned architecture. We name our approach the **Conditioned-U-Net** (C-U-net). It differs from the previous works where a dedicated model is trained for a single task [1] or where it has a fixed number of outputs [19].

As in [1, 19], our **conditioned network** is a standard U-Net that computes a set of generic source separation filters that we use to separate the various instruments. It adapts itself through the control mechanism (the condition generator) with FiLM layers inserted at different depths. Our external data is a condition vector $\overline{z}$ (a one-hot-encoding) which specify the instrument to be separated. For example, $\overline{z} = [0, 1, 0, 0]$ corresponds to the drums. The vector $\overline{z}$ is the input to the control mechanism/condition generator that has to learn the best $\gamma_{i,c}$ and $\beta_{i,c}$ values such that, when they modify the feature maps (in the FiLM layers) the C-U-Net separates the indicated instrument i.e., it decides which features maps information is useful to get each instrument. The control mechanism/condition generator is itself a neural network that embeds $\overline{z}$ into the best $\gamma_{i,c}$ and $\beta_{i,c}$. The conditioned network and the condition generator are trained jointly. A diagram is shown in Figure3.

Our C-U-Net can perform different instrument source separations as it alters its behavior depending on the value of the external condition vector $\overline{z}$. The inputs of our system are the mixture and the vector $\overline{z}$. There is only one output, which corresponds to the isolated instrument defined by $\overline{z}$. While training, the output corresponds to the desired isolated instrument that matches the $\overline{z}$ activation.

4.1 Conditioned network: U-Net architecture

We used the U-Net architecture proposed for vocal separation [1], which is an adaptation of the microscopic images

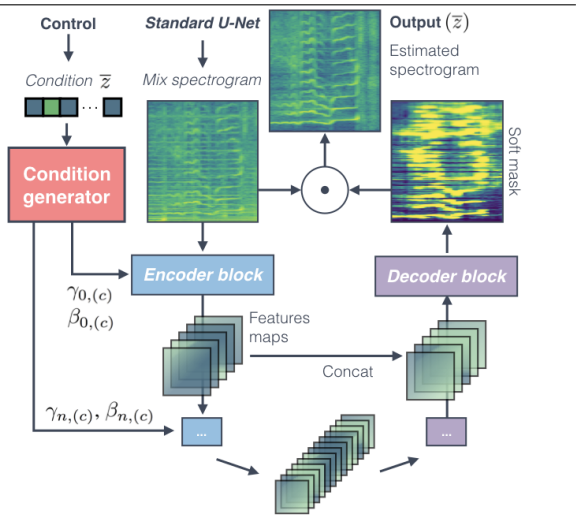


Figure 3: The C-U-Net has two distinct parts: the condition generator and a standard U-Net. The former codifies the input the condition vector, $\overline{z}$ (with the instrument to isolate) for getting the needed $\gamma_{i,(c)}$ and $\beta_{i,(c)}$. The generic U-Net has as input the magnitude spectrum. It adapts its conduct via FiLM layers inserted in the encoder part. The system outputs the desired instrument defined by $\overline{z}$.

U-Net [17]. The input and output are magnitude spectrograms of the monophonic mixture and the instrument to isolate. The U-Net follows an encoder-decoder architecture and adds a skip connection to it.

Encoder. It creates a compressed and deep representation of the input by reducing its dimensionality while preserving the relevant information for the separation. It consists of a stack of convolutional layers, where each layer halves the size of the input but doubles the number of channels.

Decoder. It reconstructs and interprets the deep features and transforms it into the final spectrogram. It consists of a stack of deconvolutional layers.

Skip-connections. As the encoder and decoder are symmetric i.e., feature maps at the same depth have the same shape, the U-Net adds skip-connections between layers of the encoder and decoder of the same depth. This refines the reconstruction by progressively providing finer-grained information from the encoder to the decoder. Namely, feature maps of a layer in the encoder are concatenated to the equivalent ones in the decoder.

The final layer is **a soft mask** (sigmoid function $\in [0, 1]$) $f(X, \theta)$ which is applied to the input X to get the isolated source Y. The loss of the U-Net is defined as:

$$\mathcal{L}(X, Y; \theta) = \|f(X, \theta) \odot X - Y\|_{1,1} \qquad (2)$$

where θ are the parameters of the system.

Architecture details. Our implementation mimics the original one [1]. The **encoder** consists in 6 encoder blocks. Each one is made of a 2D convolution with 5x5 filters, stride 2, batch normalisation, and leaky rectified linear

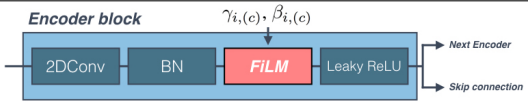


Figure 4: FiLM layers are placed after the batch normalisation. The output of a encoding block is connected to both, the next encoding block and the equivalent layer in the decoder via the skip connections.

units (ReLU) with leakiness 0.2. The first layer has 16 filters and we double them for each new block. The **decoder** maps the encoder, with 6 decoders blocks with stride deconvolution, stride 2 and a 5x5 kernel, batch normalisation, plain ReLU, and a 50% dropout in the first three. The final one, the soft mask, uses a sigmoid activation. The model is trained using the ADAM optimiser [11] and a 0.001 learning rate. As in [1], we downsample to 8192 Hz, compute the Short Time Fourier Transform with a window size of 1024 and hop length of 768 frames. The input is a patch of 128 frames (roughly 11 seconds) from the normalised (per song to [0, 1]) magnitude spectrogram for both the mixture spectrogram and the isolated instrument.

Inserting FiLM. The U-Net has two well differentiated stages: the **encoder** and **decoder**. The **enconder** is the part that transforms the mixture magnitude input into a deep representation capturing the key elements to isolate an instrument. The **decoder** interprets this representation for reconstructing the final audio. We hypothesise that, if we can have a different way of encoding each instrument i.e., obtaining different deep representations, we can use a common 'universal' decoder to interpret all of them. Following this reasoning, we decided to condition only the U-Net encoder part. In the C-U-Net, a FiLM layer is inserted inside each encoding block after the batch normalisation and before the Leaky ReLU, as described in Figure 4. This decision relies on previous works where feature are modified after the normalisation [3,10,13]. Batch normalisation normalises each feature map so that it has zero mean and unit variance [8]. Applying FiLM after batch normalisation re-scale and re-shift feature maps after the activations. This allows the net to specialise itself to different tasks. As the output of our encoding blocks is transformed by the FiLM layer the data that flows through the skip connections carries on also the transformations. If we use the *FiLM complex* layer, the control mechanism/condition generator needs to generate 2016 parameters (1008 $\gamma_{i,c}$ and 1008 $\beta_{i,c}$). On the other hand, *FiLM simple* layers imply 12 parameters: one γ_i and one β_i for each of the 6 different encoding blocks, which means 2002 parameters less than for *FiLM complex* layers.

4.2 Condition generator: Embedding nets

The control mechanism/condition generator computes the $\gamma_{i,(c)}(\overline{z})$ and $\beta_{i,(c)}(\overline{z})$ that modify our standard U-net behavior. Its architecture has to be flexible and robust to generate the best possible parameters. It has also to be able to find relationships between instruments. That is to say, we want it to produce similar $\gamma_{i,(c)}$ and $\beta_{i,(c)}$ for instruments that have similar spectrogram characteristics. Hence, we explore two different embeddings: a fully connected version and a convolutional one (CNN). Each one is adapted for the *FiLM complex* layer as well as for the *FiLM simple* layer. In every control mechanism/condition generator configuration, the last layer is always two concatenated fully connected layers. Each one has as many parameters (γ's or β's) as needed. With this distinction we can control $\gamma_{i,(c)}$ and $\beta_{i,(c)}$ individually (different activations).

Table 1: Params number in millions. With dedicated U-Nets, each task needs a model with 10M params. C-U-Nets are multi-task and the number of params remains constant.

MODEL	Non-conditioned	SiF	CoF	SiC	CoC
PARAM	39,30 (4 tasks x 9,825)	9,85	12	9,84	10,42

Fully-Connected embedding (F): it is formed of a first dense layer of 16 neurons and two fully connected blocks (dense layer, a 50% dropout and batch normalised) with 64 and 256 neurons for the *FiLM simple* version and 256 and 1024 for the *FiLM complex* one. All the neurons have relu activations. The last fully connected block is connected with the final control mechanism/condition generator layer i.e., the two fully connected ones. We call the C-U-Net that uses these architectures *C-U-Net-SiF* and *C-U-Net-CoF*.

CNN embedding (C): similarly to the previous one and inspired by [18], this embedding consists in a 1D convolution with $lenght(\overline{z})$ filters followed by two convolution blocks (1d convolutional with also $lenght(\overline{z})$ filters, 50% dropout and batch normalized). The first two convolutions have 'same' padding and the last one, 'valid'. Activations are also relu. The number of filters are 16, 32 and 64 for the *FiLM simple* version and 32, 64, 252 for the *FiLM complex* one. Again, the last CNN block is connected with the two fully connected ones. The C-U-Net that uses these architectures are called *C-U-Net-SiC* and *C-U-Net-CoC*. This embedding is specially designed for dealing with several instruments because it seems more appropriated to find common $\gamma_{i,(c)}$ and $\beta_{i,(c)}$ values for similar instruments.

The various control mechanisms only introduce a reduced number of parameters to the standard U-Net architecture remaining constant regardless of the instruments to separate, Table 1. Additionally, they make direct use of the commonalities between instruments.

5. EVALUATION

Our objective is to prove that conditioned learning via FiLM (generic model+control) allows us to transform the U-Net into a multi-task system without losing performances. In Section 5.1 we review our experiment design aspects and we detail the experiment to validate the multi-task capability of the C-U-Net in Section 5.2.

5.1 Evaluation protocol

Dataset. We use the Musdb18 dataset [16]. It consists of 150 tracks with a defined split of 100 tracks for training

Table 2: Overall performance (mean ± std) for the 4 tasks. *Si*= simple FiLM, *Co*= complex FiLM, *F*= Fully-embed and *C*= CNN-embed, *p*= progressive train or *np*= not.

MODEL	Total		
	SIR	SAR	SDR
Fix-U-Net(x4)	7.31 ± 4.04	5.70 ± 3.10	2.36 ± 3.96
C-U-Net-SiC-np	7.35 ± 4.13	5.74 ± 3.18	2.34 ± 3.69
C-U-Net-SiC-p	**8.00** ± 4.37	**5.74** ± 3.63	**2.54** ± 4.07
C-U-Net-CoC-np	7.27 ± 4.24	5.60 ± 2.88	2.36 ± 3.81
C-U-Net-CoC-p	7.49 ± 4.54	5.67 ± 3.03	2.42 ± 4.21
C-U-Net-SiF-np	7.23 ± 3.97	5.59 ± 3.01	2.22 ± 3.67
C-U-Net-SiF-p	7.64 ± 4.05	5.73 ± 2.88	2.46 ± 3.88
C-U-Net-CoF-np	7.42 ± 4.20	5.59 ± 3.07	2.32 ± 3.85
C-U-Net-CoF-p	7.52 ± 4.04	5.71 ± 2.99	2.42 ± 3.97

and 50 for testing. From the 100 tracks, we use 95 (randomly assigned) for training, and the remaining 5 for the validation set, which is used for early stopping. The performance is evaluated on the 50 test tracks. In Musdb18, mixtures are divided into four different sources: **Vocals**, **Bass**, **Drums** and **Rest** of instruments. The 'Rest' task mixes every instrument that it is not vocal, bass or drums. Consequently, the C-U-Net is trained for four tasks (one task per instrument) and $\overline{z}$ has four elements.

Evaluation metrics. We evaluate the performances of the separation using the mir evaltoolbox [14]. We compute three metrics: Source-to-Interference Ratios (SIR), Source-to-Artifact Ratios (SAR) and Source-to-Distortion Ratios (SDR) [24]. To compute the three measure we also need the predicted 'accompaniment' (the mixture part that does not correspond to the target source). Each task has a different accompaniment e.g., for the drums the accompaniment is rest+vocals+bass. We create the accompaniments by adding the audio signal of the needed sources.

Audio Reconstruction method. The system works exclusively on the magnitude of audio spectrograms. The output magnitude is obtained by applying the mask to the mixture magnitude. As in [1], the final predicted source (the isolated audio signal) is reconstructed concatenating temporally (without overlap) the output magnitude spectrums and using the original mix phase unaltered. We compute the predicted accompaniment subtracting the predicted isolated signal to the original mixture. Despite there are better phase reconstruction techniques such as [12], errors due to this step are common to both methods (U-Net and C-U-Net) and do not affect our main goal: to validate conditioning learning for source separation.

Activation function for γ and β. One of the most important design choices is the activation function for $\gamma_{i,(c)}$ and $\beta_{i,(c)}$. We tested all the possible combinations of three activation functions (linear, sigmoid and tanh) in the *C-U-Net-SiF* configuration. As in [13], the C-U-Net works better when $\gamma_{i,(c)}$ and $\beta_{i,(c)}$ are linear. Hence, our γ's and β's have always linear activations.

Training flexibility. The conditioning mechanism gives the flexibility to have continuous values in the input $\overline{z} \in [0, 1]$,

Table 3: Task comparison between the *dedicated U-Nets* and the *C-U-Net-CoF*. Results indicate that they perform similarly for all the tasks. We also add the multi-instrument Wave-U-Net (M) results (median in parenthesis) and when possible the dedicated version (D). For vocals isolation the Wave-U-Net-M performs worse than the Wave-U-Net-D.

	Model	SIR	SAR	SDR
Vocals	*Fix-U-Net(x4)*	10.70 ± 4.26	5.39 ± 3.58	3.52 ± 4.88 (4.72)
	C-U-Net-CoF	10.76 ± 4.39	5.32 ± 3.27	3.50 ± 4.37 (4.65)
	Wave-U-Net-D	-	-	0.55 ± 13.67 (4.58)
	Wave-U-Net-M	-	-	-2.10 ± 15.41 (3.0)
Drums	*Fix-U-Net(x4)*	10.08 ± 4.28	6.42 ± 3.28	4.28 ± 3.65 (4.13)
	C-U-Net-CoF	10.03 ± 4.34	6.80 ± 3.25	4.30 ± 3.81 (4.38)
	Wave-U-Net-M	-	-	2.88 ± 7.68 (4.15)
Bass	*Fix-U-Net(x4)*	4.64 ± 4.76	6.51 ± 2.68	1.46 ± 4.31 (2.48)
	C-U-Net-CoF	5.30 ± 4.73	6.29 ± 2.39	1.65 ± 4.07 (2.60)
	Wave-U-Net-M	-	-	-0.30 ± 13.50 (2.91)
Rest	*Fix-U-Net(x4)*	3.83 ± 2.84	4.47 ± 2.85	0.19 ± 3.00 (0.97)
	C-U-Net-CoF	4.00 ± 2.70	4.37 ± 3.06	0.24 ± 3.64 (1.71)
	Wave-U-Net-M	-	-	1.68 ± 6.14 (2.03)

which weights the target output Y by the same value. We call this training method **progressive**. In practice, while training, we randomly weight $\overline{z}$ and Y by a value between 0 and 1 every 5 instances. This is a way of dealing with ablations by making the control mechanism robust to noise. As shown in Table 2, this training procedure *(p)* improves the models. Thus, we adopt it in our training. Moreover, preliminary results (not reported) show that the C-U-Net can be trained for complex tasks like bass+drums or voice+drums. These complex tasks could benefit from 'in between-class learning' method [21] where $\overline{z}$ will have different intermediate instrument combinations.

5.2 Multitask experiment

We want to prove that a given C-U-Net can isolate the **Vocals**, **Drums**, **Bass**, and **Rest** as good as four dedicated U-Net trained specifically for each task [2] We call this set of dedicated U-Nets, *Fix-U-Nets*. Each C-U-Nets version (one model) is compared with the Fix-U-Nets set (four models). We review the results at Table 2 and show a comparison per task in Table 3.

Results in Table 2 for all 4 instruments highlight that *FiLM simple* layers work as good as the complex ones. This is quite interesting because it means that applying 6 affine transformations with just 12 scalars (6 γ_i and 6 β_i) at a precise point allows the C-U-Net to do several source separations. With *FiLM complex* layers it is intuitive to think that treating each feature map individually let the C-U-Net learn several deep representations in the encoder. However, we have no intuitive explanation for *FiLM simple* layers. We did the Tukey test with no significant differences between the *Fix-U-Nets* and the *C-U-Nets* for any task and metric. Another remark is that the four C-U-Nets benefit from the *progressive* training. Nevertheless, it impacts more the simple layers than in the complex ones. We think that the restriction of the former (fewer parameters) helps them to find an optimal state.

[2] with the same learning rate and optimizer as the C-U-Nets.

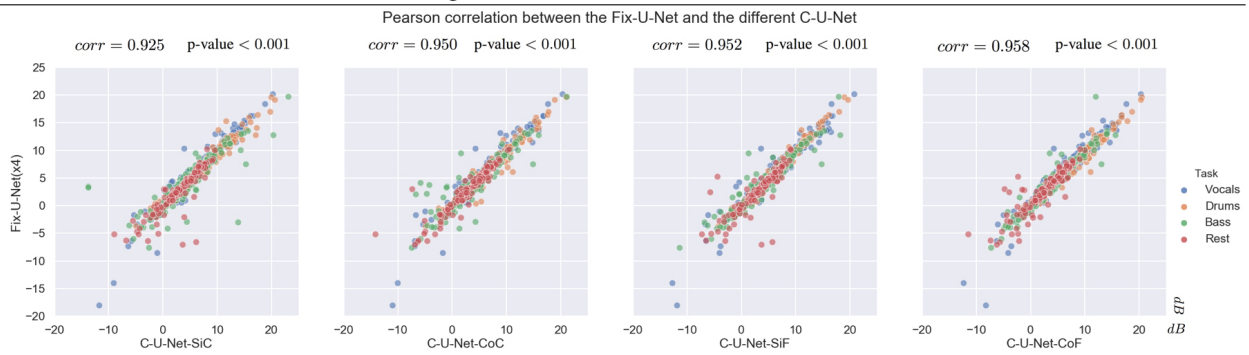


Figure 5: Each graph correlates the performance of two models. On top of it, we show the correlation and p-value. The 'y' axis represents the fixed version (the four dedicated U-Nets) and the 'x' one a different C-U-Net version (with progressive train). The coordinates of each dots correspond to the models' performance i.e., 'y' position for the Fix-U-Net performance and 'x' for C-U-Net. There are three dots per song one per metric (SIR, SAR, and SDR) which does a total of 600 (50 songs x 3 metrics x 4 instruments). The dots alignment in the diagonal implies a strong correlation between models: if one works well, the others too and vice versa. Each color highlights the points of each source separation task.

However, these results do not prove nor discard the significant similarity between systems. For demonstrating that we have carried out a Pearson correlation experiment. The results are detailed in Figure 5. The Pearson coefficient measures the linear relationship between two sets of results (+1 implies an exact linear relationship). It also computes the p-value that indicates the probability that uncorrelated systems have produces them. Our distinct C-U-Net configurations have a global $corr > .9$ and p-value < 0.001. Which means that there is always more than 90% correlation between the performance of the four dedicated *U-Nets* and the (various) conditional version(s). Additionally, there is almost no probability that a C-U-Net version is not correlated with the dedicated ones. We have also computed the Pearson coefficient and p-value per task and per metric with the same results. In Figure 5 shows a strong correlation between the *Fix-U-Net* results and the distinct *C-U-Nets* (independently of the task or metric). Thus, if one works well, the others too and vice versa.

In Table 3 we detail the results per task and metric for the *Fix-U-Net* and the *C-U-Net-CoF* which is not the best C-U-Net but the one with the highest correlation with the dedicated ones. There we can see how their performances are almost identical. Nevertheless, our vocal isolation (in any case) is not as good as the one reported in [1], we believe that this is mainly due to the lack of data. These results can only be compared with the Wave-U-Net [19]. Although they report the results (only the SDR) for the four tasks in the multi-instrument version (multiple outputs layers) they only have a dedicated version for vocals. For vocal separation, the performance of the multi-instrument version decreases more than 2.5 dB in mean, 1.5 dB in the median and the std increase in almost 2 dB. Furthermore, the C-U-Net performs better than the multi-instrument in three out of four tasks (vocals, bass, and drums) [3] . For the 'Rest' task, the multi-instrument wave-u-net outperforms our C-U-Nets. This is normal because the dedicated U-Net has already problems with this class and the C-U-Nets inherits the same issues. We believe that they come from the vague definition of this class with many different instruments combinations at once.

This proves that the various C-U-Nets behave in the same way as the dedicated U-Nets for each task and metric. It also demonstrates that conditioned learning via FiLM is robust to diverse control mechanisms/condition generators and FiLM layers. Moreover, it does not introduce any limitations which are due to other factors.

6. CONCLUSIONS AND FUTURE WORK

We have applied conditioning learning to the problem of instrument source separations by adding a control mechanism to the U-Net architecture. The C-U-Nets can do several source separation tasks without losing performance as it does not introduce any limitation and makes use of the commonalities of the distinct instruments. It has a fixed number of parameters (much lower than the dedicated approach) independently of the number of instruments to separate. Finally, we showed that *progressive training* improves the C-U-Nets and introduced the *FiLM simple*, a new conditioning layer that works as good as the original one but requires less γ's and β's.

Conditioning learning faces problems providing a generic model and a control mechanism. This gives flexibility to the systems but introduces new challenges. We plan to extend the C-U-Net to more instruments to find its restrictions and to explore the performance for complex tasks i.e., separating two or more instruments combinations (e.g., vocals+drums). Likewise, we are exploring ways of adding new conditions (namely new instrument isolation) to a trained C-U-Net and how to detach the joint training. Additionally, we intend to integrate it other source separation architectures such as Wave-U-Net.

Lastly, we believe that conditioning learning via FiLM will benefit many MIR problems because it defines a transparent and direct way of inserting external data to modify the behavior of a network.

[3] Our experiment conditions are different in training data size (95 Vs 75) and in sampling rate (8192 Hz Vs 22050 Hz) than Wave-U-Net.

Acknowledgement. This research has received funding from the French National Research Agency under the contract ANR-16-CE23-0017-01 (WASABI project). Implementation and audio examples available at https://github.com/gabolsgabs/cunet

7. REFERENCES

[1] N. Montecchio R. Bittner A. Kumar T. Weyde A. Jansson, E. J. Humphrey. Singing voice separation with deep u-net convolutional networks. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Suzhou, China, 2017.

[2] P. Chandna, M. Miron, J. Janer, and E. Gómez. Monoaural audio source separation using deep convolutional neural networks. In *Proc. of LVA/ICA (International Conference on Latent Variable Analysis and Signal Separation)*, Grenoble, France, 2017.

[3] H. de Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville. Modulating early visual processing by language. In *Proc. of NIPS (Annual Conference on Neural Information Processing Systems)*, Long Beach, CA, USA, 2017.

[4] V. Dumoulin, E. Perez, N. Schucher, F. Strub, H. de Vries, Aaron Courville, and Y. Bengio. Feature-wise transformations. *Distill*, 2018. https://distill.pub/2018/feature-wise-transformations.

[5] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck. Onsets and frames: Dual-objective piano transcription. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Suzhou, China, 2018.

[6] C. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman, and D. Eck. An improved relative self-attention mechanism for transformer with application to music generation. *CoRR*, abs/1809.04281, 2018.

[7] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Joint optimization of masks and deep recurrent neural networks for monaural source separation. *IEEE/ACM TASLP (Transactions on Audio Speech and Language Processing)*, 23(12), 2015.

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of ICML (International Conference on Machine Learning)*, 2015.

[9] H. Kameoka, Li Li, S. Inoue, and S. Makino. Semi-blind source separation with multichannel variational autoencoder. *CoRR*, abs/1808.00892, 2018.

[10] T. Kim, I. Song, and Y. Bengio. Dynamic layer normalization for adaptive neural acoustic modeling in speech recognition. *CoRR*, abs/1707.06065, 2017.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of ICLR (International Conference on Learning Representations)*, Banff, Canada, 2014.

[12] F. Mayer, D. Williamson, P. Mowlaee, and D. Wang. Impact of phase estimation on single-channel speech separation based on time-frequency masking. *The Journal of the Acoustical Society of America*, 141:4668–4679, 2017.

[13] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville. Film: Visual reasoning with a general conditioning layer. In *Proc. of AAAI (Conference on Artificial Intelligence)*, New Orleans, LA, USA, 2018.

[14] C. Raffel, B. Mcfee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis. mir_eval: a transparent implementation of common mir metrics. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Porto, Portugal, 2014.

[15] Z. Rafii, A. Liutkus, F.-R. Stöter, S. Ioannis Mimilakis, D. Fitzgerald, and B. Pardo. An Overview of Lead and Accompaniment Separation in Music. *IEEE/ACM TASLP (Transactions on Audio Speech and Language Processing)*, 26(8), 2018.

[16] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. The MUSDB18 corpus for music separation, 2017. https://zenodo.org/record/1117372.

[17] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. of MICCAI (International Conference on Medical Image Computing and Computer Assisted Intervention)*, Munich, Germany, 2015.

[18] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. J. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu. Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions. In *Proc. of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*, Calgary, Canada, 2018.

[19] D. Stoller, S. Ewert, and S. Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Paris, France, 2018.

[20] F. Strub, M. Seurin, E. Perez, H. de Vries, J. Mary, P. Preux, A. C. Courville, and O. Pietquin. Visual reasoning with multi-hop feature modulation. In *Proc. of ECCV (European Conference on Computer Vision)*, Munich, Germany, 2018.

[21] Y. Tokozume, Y. Ushiku, and T. Harada. Between-class learning for image classification. In *Proc. of CVPR (Conference on Computer Vision and Pattern Recognition)*, Salt Lake City, UT, USA, 2018.

[22] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016.

[23] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis. Parallel wavenet: Fast high-fidelity speech synthesis. *CoRR*, abs/1711.10433, 2017.

[24] E. Vincent, R. Gribonval, and C. Févotte. Performance measurement in blind audio source separation. *IEEE/ACM TASLP (Transactions on Audio Speech and Language Processing)*, 14(4), 2006.

[25] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation using 1d and 2d conditions. *CoRR*, abs/1703.10847, 2017.

AN INITIAL COMPUTATIONAL MODEL FOR MUSICAL SCHEMATA THEORY

Andreas Katsiavalos[1] **Tom Collins**[2,3] **Bret Battey**[1]
[1] Music, Technology and Innovation Research Centre, De Montfort University, UK
[2] Music, Science and Technology Research Cluster, Department of Music, University of York, UK
[3] Music Artificial Intelligence Algorithms, Inc., Davis, CA, USA
andreas.katsiavalos@gmail.com, tomthecollins@gmail.com

ABSTRACT

Musical schemata theory entails the classification of subphrase-length progressions in melodic, harmonic and metric feature-sets as named entities (e.g., 'Romanesca', 'Meyer', 'Cadence', etc.), where a musical schema is characterized by factors such as music content and form, position and tonal function within phrase structure, and interrelation with other schemata. To examine and automate the task of musical schemata classification, we developed a novel musical schemata classifier. First, we tested methods for exact and approximate matching of user-defined schemata prototypes, to establish the notions of identity and similarity between composite music patterns. Next, we examined methods for schemata prototype extraction from collections of same-labelled annotated examples, performing training and testing sessions similar to supervised learning approaches. The performance of the above tasks was verified using the same annotated dataset of 40 keyboard sonata excerpts from pre-Classical and Classical periods. Our evaluation of the classifier sheds light on: (a) ability to parse and interpret music information, (b) similarity methods for composite music patterns, (c) categorization methods for polyphonic music.

1. INTRODUCTION

Schemata have been characterised by psychologists as 'the building blocks of cognition' [17], enabling an understanding of the world in packages of knowledge. Similarly, musical schemata can be thought of as 'minimal meaningful' entities, enabling coherent interpretations of Classical phrases. Musical schemata theory is studied and developed by musicologists as a means of classifying short passages in musical works, mainly from the Classical period [2, 10].

Aiming to model musical schemata theory, we consider the development of computational systems that create and update definitions for prototypes of schemata categories from annotated music examples. This work builds on a relatively small amount of existing research in computational modeling of musical schemata theory [8, 9, 18].

The novel system will enable the study of higher-level operations and reasoning in content-based music information retrieval than has been possible to date, and facilitate further research with machine learning approaches in music pattern extraction [15, 19, 20].

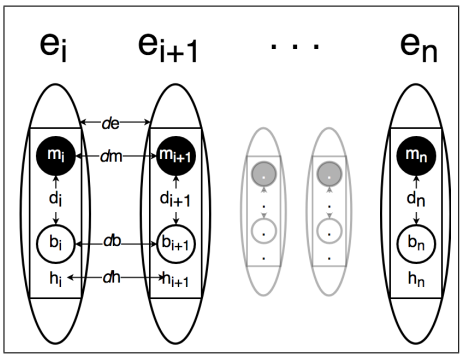


Figure 1. Musical schemata are considered as sequences of schema-events.

2. TASK DESCRIPTION AND BACKGROUND

2.1 Musical schemata in Classical keyboard works

A musical schema is defined as a stereotypical progression of *schema-event* elements (Figure 1): a feature-set consisting of notes from two melodic movements (melody and bass) and harmonic and metric information [7, 10]. A schema is characterized by its content, that is, the number and type of its constituent schema-events, but also as part of phrases and even greater morphological entities such as paragraphs, periods, etc., as well as its position, tonal function, and any interrelations with other schemata.

An interesting aspect of the theory is the notion of a prototype for a set of similar progressions. The extraction of schemata prototypes is a learning process and musical schemata theory is an example-based approach, meaning that knowledge about a schema prototype is obtained and updated from examples and not by rules.

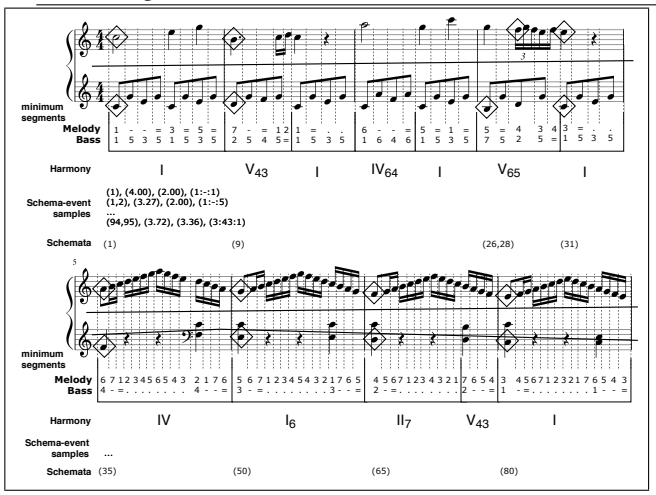


Figure 2. Annotated example with melodic movement separation (horizontal lines), harmonic regions, and schemata notes (diamonds). The first line of the score has a 'Meyer' instance and the second, a 'Prinner'. Excerpt from Wolfgang Amadeus Mozart, Piano Sonata no.16 in C major K545 1st movement Allegro mm.1-8.

2.2 Modeling musical schemata theory

We implemented example-based machine learning with the development of a musical schemata classifier. The classifier works in three steps: making observations of the music data, comparing these observations to stored schema prototypes, and identifying schema-class similarity.

2.2.1 Observations

To classify music patterns such as musical schemata, the input needs to be processed so that only relevant, 'schematic' material is considered. As described previously, musical schemata are viewed as progressions of schema-events, and for that reason we consider two types of 'observations': a) schema-states, and, b) schema-state combinations/progressions. We will refer to these elements as 'low-' and 'high-observations' respectively. The schema-states are schema-voice and schema-event samples, a kind of low-level form of music understanding for small feature-sets. Combining these schema-state elements, we can create schemata instances/samples that are comparable to musical schemata prototypes and thus, perform similarity and classification tasks.

Creating the schema-state observations is a preprocessing step, independent from the tasks of recognition and learning, but the observation process can utilize information from stored prototypes to select only known (stored) schema-states and, thus, reduce the number of schema-event state observations.

2.2.2 Similarity

A fundamental task of the classifier is to perform comparisons of same-class information (e.g. states or schemata) from different sources (e.g. observations, prototypes). The matching of observations from a source score and a stored schema prototype is the recognition process. To evaluate our musical schemata classifier, these recognition results are compared/validated against schemata annotations, a 'ground-truth', giving measures of recall and precision.

To perform comparisons, the idea is to define a similarity metric for multi-feature elements and sequences, such as schema-states and complete schemata, and utilize it to identify and categorize them according to their content.

2.2.3 Class similarity

In addition to recognition and evaluation similarity, class similarity is another type of similarity whereby common relations amongst schema states and schemata observations of the same schema type are identified, hence, describing the properties of a schema family type. In this study, a schema class is defined by a set of examples, an example-base of high-level observations, and a class (similarity) function that validates all its existing examples, acting as the identity validator for all of them.

An interesting aspect of our classification method for schemata is that, contrary to matching methods that create a search-space for specific targets, our approach can go beyond merely finding matches to existing schemata prototypes to automatically identifying potential schemata.

3. METHODOLOGY

3.1 Method overview

The musical schemata classifier was developed by application to incrementally more complex scenarios:

- Identification (labelled I hereafter). Initially, the classifier performed musical schemata identification with pattern matching techniques, identifying the exact positioning of user-defined schema-family prototypes in complete music parts;
- Recognition (labelled R and F). Next, we examined the task of multiple and approximate recognition of user-defined schema-family prototypes, to study similarity-approximation methods for both single-family prototypes (R) and collections of multiple variants for a single schema class (F);
- Learning (labelled L). Finally, we tested the schemata classifier on learning of schemata prototypes from schema-annotated music examples.

The annotations were of a dataset of 40 keyboard sonata parts (10 from Haydn, 10 from Mozart, and 20 from Beethoven), containing annotations for 3 schemata types ('Meyer', 'Prinner', and, 'Cadence'). In total the number of unique annotations are: 17 generic and 23 'Meyer' variations, 22 generic and 8 'Prinner' variations, and, 40 generic and 15 'Cadence' variations.

3.2 Model overview

The details of the model's implementation are beyond the scope of this paper and therefore the following provides

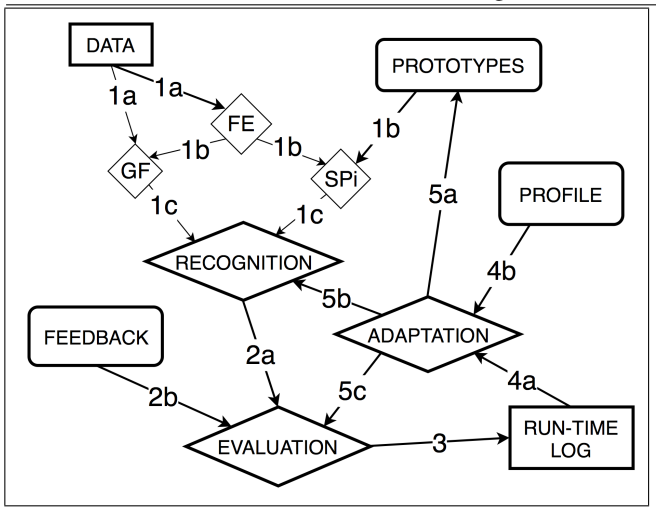


Figure 3. Top view of the schemata classifier model.

only a high-level overview of the approach. [1] The classifier runs in sessions where information from up to five sources is processed and combined (please see Figure 3). The *music-data* source (Figure 3, DATA) supplies complete sonata parts in MusicXML format sequentially, and each input file follows the same information processing path, where the input file is first converted into operational representations to perform score analysis and feature extraction, and then extract state samples (schema-events and -voices). The *memory-data* source (Figure 3, PROTOTYPES) is where schemata prototypes reside and are being recalled for recognition. The *feedback* source (Figure 3, FEEDBACK) inputs non-musical information to the model and is utilized to pass annotations, such as schemata labels for measure ranges. The run-time information of a session is written to a *session log* (Figure 3, RUN-TIME LOG). The learning operations are governed by a system *profile* (Figure 3, PROFILE).

In general, the classifier creates and compares two classes of information: a) *schematic states*, either schema-events or schema-voices, and b) combinations and/or progressions of (a). The three basic operations that connect these information sources all relate to similarity and are: i) recognition, during which states and schemata from memory and data are compared, ii) evaluations, where the results from (i) are compared with annotations, and, iii) adaptation, where the class similarity function of a schema type is updated to include new entries to the example base (the inclusion of a training observation).

The model also employs a number of task-specific functions with smaller scope that aid the similarity functions described above. For example, the results of feature extraction (Figure 3, FE) are sent to both grouping functions (Figure 3, GF, to extract schema-state and progression samples) and a schemata prototype instantiation function (Figure 3, SPi, to create comparable instances from schemata prototypes).

[1] See `https://tinyurl.com/y5x2d99j` for code and data.

3.3 Making music observations

The processing path for each music input element (a complete sonata part) starts with the conversion of the MusicXML file into operational encodings (datapoints and 'minimal segments' [16]) to perform tonal and harmonic score analysis and feature extraction, utilizing formalisms in symbolic music processing from [14] and [4]. Next, stylistic reduction, a combination of the aforementioned analyses for the extraction of schematic information, selects material from the score for the sampling of schema-states (schema-voice and -event feature sets).

In addition to notated information available from the MusicXML encoding, information about metric, tonal, and harmonic properties is extracted utilizing task-specific algorithms. Rhythm is extracted from the notated time signature of the input file and is embedded to operational representations in the form of *metric strength* using [5]. Tonality is extracted from notation using 'key' and 'mode' attributes from MusicXML, but also using probe-tone profiles [12]. Harmony is extracted in segments using the *HarmAn* algorithm [16]. Complex voice separation is not undertaken – rather, the outer notes, based on absolute pitch, are considered to comprise melodic and bass movements.

3.3.1 Sampling schematic-states

After score analysis and feature extraction, the system has enough information to extract schema-state samples. A schema-voice is considered a monophonic sequence of datapoints whose adjacent temporal interval is constrained by extracted features that relate with metric information. The extraction of schema-voices, often termed as music n-grams, is in compliance with voice-leading rules [11]. The schema-event sampling algorithm starts with score analysis information and 'minimal segments', and generates *schema-events*, a type of temporal reduction with similarities to *time-span* reduction described in [13]. First, each 'minimal segment' is assigned positional and (adjacent) transitional 'significance' from two algorithms that weight parameters from the 'minimal segment' properties and harmony. Then, a 'minimal segment' merger creates samples from pairs of 'minimal segment' elements by combining and selecting values for the schema-event characteristics.

The positional 'significance' of each 'minimal segment' is found using (1) and the weights in Table 1:

$$poSig = \tfrac{1}{3}(dpQ + bStr + card) + outVp \qquad (1)$$

where *poSig* is the significance of the 'minimal segment', *dpQ* is the overall quality of datapoints in the 'minimal segment', *bStr* is the beat-strength value, *card* is the cardinality, and *outVp* is the bonus from movement in outer voices.

Transitional 'significance' is calculated in a similar manner but also considering the changes between the adjacent 'minimal segment' elements, mainly in harmony.

A schema-event sample has ontime and duration and all the properties of a prototype schema-event but most importantly, each schema-event sample is rated with the

Feature	Factor
beat strength	2
harmony	2
cardinality	1.5
outer voices	2
complete datapoint in 'minimal segment'	1
starting datapoint in 'minimal segment'	1
ending datapoint in 'minimal segment'	0.25
middle datapoint in 'minimal segment'	0.125

Table 1. The manual weights for *positional* 'significance' (poSig) of single 'minimal segment' elements.

combined 'significance' ratings of its 'minimal segment' elements, allowing further thresholding and selection.

For example, the total number of datapoints and 'minimal segment' elements for the excerpt in Figure 2 is 124 and 95, respectively. Thresholding 'minimal segment' elements using above-average positional 'significance' ratings, event-sampling returns a total of 107 schema-event samples in the following form: ((measure, time signature, beat), (pitch-value of each voice's datapoints: [melody, bass]), (positional 'significance'), (transitional 'significance'), (melody:harmony:bass, in scale-degrees)). For example, the first sample of the excerpt in Figure 2 is:

$$((1, 4/4, 1.0), [C5*, C4], (4.00), (2.00), (1 : - : 1))$$

At a later stage, the type of schemata and the position in which they appear in prototypes are also added to each sample.

With the extraction of schematic material, in the form of schema-voices and schema-events, we have extracted low-level observations, i.e. schema-states. These elements are utilized later for the creation of complete schemata-instances (high-level observations).

3.4 The memory module

The *memory module* stores the schemata prototypes and handles creation, update and instantiation operations of each prototype. It is the knowledge-base, a repository of generic prototypes starting with definitions from [10], where each schema prototype is represented as a sequence of schema-events (please see Figure 1).

3.5 Recognition

A core task of the schemata classifier is to assign/identify prototype schema-states in low-level observations and complete schemata prototypes in high-level observations, with the latter consisting of pairs of schema-voices and progressions of schema-events.

From the data part, post-stylistic analysis (please refer to the data processing path in 3.3), the information of input elements is converted into a set of state samples for voices and events. From the memory part, a target space consisting of schema-states and prototype schemata instances is created according to extracted features (Figure 3, SPi).

3.5.1 Creating and thresholding schemata search space

After the extraction of schema-states, the classifier applies an observation method to select and group state-samples in schema-instances that are comparable to schemata prototypes. Initially, schema-state samples are thresholded based on their 'significance' ratings and, optionally, according to their similarity with prototype states. When forming schemata samples, state-groups are filtered by number of events (minimum/maximum), a minimum/maximum duration, and temporal regularity [6, 18].

3.5.2 Similarity between prototypes and constructs

When comparing extracted and prototype states and schemata, these can either be exact or different, with the latter case being comparable with a similarity metric.

3.5.3 Proto-state similarity

A schema-voice has a single melodic movement and is characterized by the number of notes it contains. Matching a schema-voice sample with a prototype schema-voice is a sequence-similarity task and two voice-states are the same, if their content is the same, regardless of the temporal relations of the datapoints that comprise them. Considering the order of appearance, the position of a datapoint in each voice-state, the difference between schema-voice samples and prototype schema-voices is measured as the sum of differences of same position datapoints, also known as the Hamming distance. For example, the distance d_H between schema-voices (in scale-degrees): a) 1,7,5,3, b) 1,7,4,3, and c) 6,5,4,3, is: $d_H(a, b) = 1$, $d_H(a, c) = 3$, and, $d_H(b, c) = 2$.

The similarity between event-states is hierarchical, considering two layers of similarity for harmonic and melodic information, and Boolean, meaning that differences are not quantified. The harmonic information of two schema-events can differ in type of harmony, expressed within the tonal context (e.g., I, II, etc.) and in the arrangement of the notes within the chord, the type of chord inversions (e.g., 53, 63, etc.). To get a single numeric value from the multi-feature comparisons of schema-events and maintain the types of difference, we consider different decimal powers for each type of difference. Thus, difference in harmonic type equals to 10^3, in chord type, to 10^2, in melody, 10^1, and in bass, 10^0. Therefore, the possible values from a schema-event state comparison are equal to 2^4 (0, 1, 10, 11,. . ., 1110, and 1111). For example, the comparison between schema-events c_{SE} with values (bass, bass-intervals, melody) a) (1, 53, 5), b) (1, 63, 3), and c) (3, 63, 1), is $c_{SE}(a, b) = 1110$, $c_{SE}(a, c) = 111$, and $c_{SE}(b, c) = 1011$. Temporal and metric information is not considered for single event-states comparisons.

After the comparison with prototypical schema-states, each schema-state sample can be tagged with the schema-label and index it appears in prototypes.

3.5.4 Musical schemata similarity

Musical schemata similarity is handled as a sequence similarity problem, similarly to voice-state similarity, but in-

stead of counting the number of pitch-differences in same order datapoints, schema-event differences are counted instead. Thus, similarity between two schemata structures can be expressed as the overall percentage of common schema-events.

Approximation in schemata recognition is similar to the methods presented in [3], using local, γ-, and global, δ- variability thresholds, for schema-events and progressions respectively. Thus, when comparing schema-event progressions, there are two approximation thresholds, one limiting the number and type of differences between schema-events of the same position, and another limiting the number of differences in the complete schema-event progression.

3.5.5 Recognition workflow

The recognition process begins with the comparisons between prototypes and extracted state-samples. First, each schema-state sample is tagged with the id and index of their matching prototypes. Then schema-sampling occurs, creating high-observations – progressions of schema-event samples based on regularity and 'significance' rating thresholds.

3.5.6 Schemata recognition output

The recognition process returns schema-labelled segments of the score and a set of high-observations, rated with the degrees of matching (percentage) with stored prototypes.

3.6 Learning prototypes

Learning musical schemata prototypes is about the extraction and update of schema-event progressions from annotated observations. To achieve this goal, we consider an example-base for each schema prototype class – a repository of observations within a music excerpt. After the recognition process, if the input from feedback suggests a label for a temporal region, then all observations of that segment become training observations and are added to the example-base for that particular schema-label. Maintaining an example-base for each schema-type, we retain access to all the information therein. Processing the elements of the example-base, we examine a class function that extracts relations that are common to all observations that are stored in the example-base repository of a schema. The class function validates all the example-base and is also used for recognition of unlabelled observations.

3.6.1 Class similarity function

To extract prototypes from an example-base of a schema class, we need to identify the harmonic and melodic relations that are common in all exemplars. The algorithm first generates schema-samples and returns variable-length progressions of schema-events that are then converted into harmonic and melodic progressions within the tonal context of the segment (in chord and scale degrees). These contextualized progressions are then sorted by number of schema-events, and the progressions with the highest frequency of appearance and maximum number of events are selected for prototypes.

3.7 Evaluation

The tasks of recognition and learning are evaluated in terms of recall and precision with schemata annotations in measure-level detail. The comparison of the recognition results is Boolean, meaning that the temporal range and label of a recognized schema must match exactly; otherwise they will be considered false-positives.

4. COMPUTATIONAL EXPERIMENTS

We tested four configurations of the musical schemata classifier to gradually achieve the goal of prototype extraction. First, we tested a pattern matching configuration aiming for maximum accuracy in identifying a single schema-type (label I). The second scenario examined approximate matching of a single-schema (label R). The third configuration tests approximate matching of a schema-family, including variations (label F). The last scenario tests the learning algorithm and the extraction of a prototype for annotated examples (label L).

4.1 Maximum accuracy for a user-defined schema

The first model configuration for the schemata classifier examined methods for the extraction of schema-voice states and exact schemata matching for two schemata types, namely 'Meyer' and 'Prinner' (please see Figure 2 for their generic types). The configuration uses datapoints, extracted tonalities, and no harmonic information. The algorithm creates a search-space of schema-voice samples and filters them in pairs, with temporal regularity, to compare them with schemata prototypes. There is no variation in matching and no information preservation after each input element. A recognition example of the algorithm and its configuration is shown in Figure 4.

4.2 Approximate recognition for multiple user-defined schemata targets

The task of approximate recognition of multiple schemata prototypes is examined with two model configurations, matching 'Meyer', 'Prinner', and 'Cadence' schemata. The first case searches for generic approximations (omissions and thresholds in similarity metrics) of each schema family prototype, by first tagging all the extracted schema-states with labels and positions of similar corresponding elements in prototypes and then recognizes complete schemata by combining elements with the same schema class tag under temporal limitations. The second case recognizes approximations of schema families (collections of variants), performing comparisons between schema-samples and class functions that validate all the variants of a schema class. The type of matching approximation of the second case includes those of the first case, but also hierarchical comparisons, considering harmonic similarity first, as a prerequisite, and voice similarity second.

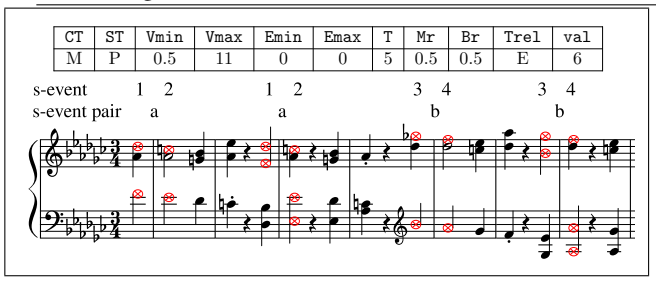


Figure 4. Identification of a generic 'Meyer' instance. The parameters from left to right: Manual search configuration for generic 'Meyer' prototype with temporal distance between events in a range between 0.5 and 11 beats, and 0 for schema-event durations, for a single tonality (5) and 0.5 regularity threshold for temporal intervals in each melodic movement, and maximum (6) thresholding in inter-event temporal regularity. From Ludwig van Beethoven, Piano Sonata in C-sharp minor op.27 no.2 2nd mvt mm.1-7.

4.3 Approximate recognition of extracted schema family type from examples

This computational experiment examines prototype extraction and classification. The algorithm maintains repositories of exemplars for each schema-type separately, from which prototype forms of schemata are extracted utilizing a class function. The validity of the extracted prototypes is examined by evaluating the recognition performance with the extracted prototypes on previously unseen examples.

5. RESULTS

The identification task (Figure 5, $I_{<SCHEMA>}$) achieved high recall and precision for both schemata types, indicating that basic music information processing of the classifier is working as it should.

The recognition models, due to approximate matching, have increased recall but lower precision (Figure 5, (Figure 5, $R_{<SCHEMA>}$, $F_{<SCHEMA>}$). The first configuration that tests family-prototype generic approximation (Figure 5, $R_{<SCHEMA>}$) can be very imprecise, as approximating the prototypical form can yield completely unrelated patterns. The second configuration (Figure 5, $F_{<SCHEMA>}$) has slightly lower recall from the first but increased precision, due to more targeted approximations with the use of a class function.

The learning model (Figure 5, $L_{<SCHEMA>}$) achieved low recall, due to occasional erroneous prototype extractions and even lower precision, because of the highly-generalised extractions of prototypes.

6. FINDINGS AND DISCUSSION

This paper presented a prototype model architecture for musical schemata classification. It is among a small handful of computational models for extracting instances of such high-level music-theoretic concepts from input staff notation.

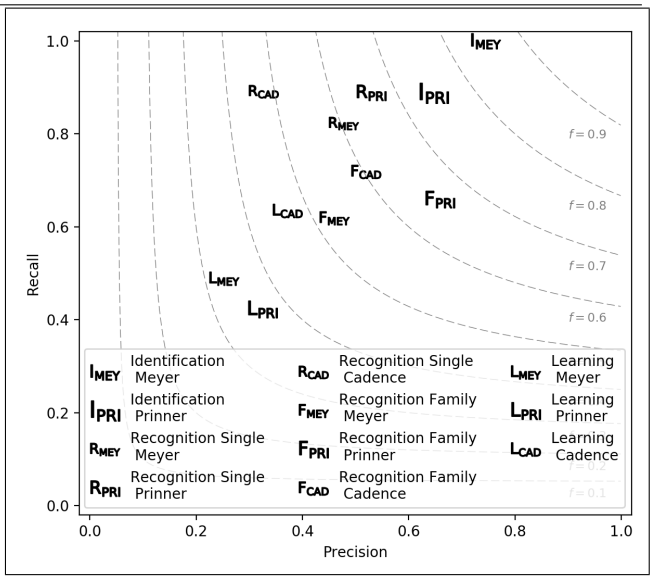


Figure 5. The F_1 score of all musical schemata identification models.

The computational model that was developed for a musical schemata classifier was tested under different tasks and configurations, and proves to be a reliable framework that can facilitate classification operations regarding musical patterns. The use of structured music information (i.e. the 'low' and 'high' observations) enabled high-level operations such as comparisons of abstract patterns, and the methods that were developed to produce them can be further fine-tuned. Moreover, the maintenance of an example-base proved very helpful in prototype extraction, as it provides transparency of operations. Furthermore, the repository of a learning session can be reused in other sessions, transferring learned knowledge.

The model was tested on a small dataset and its expansion both in number and types of schemata annotations is a natural future task. There are numerous improvements that can be applied in this prototype model by fine-tuning individual functions. For example, the sampling mechanisms for schema-states and progressions can become adaptive to local context, instead of global, part-wise features, and thus become more accurate. Another interesting and informative development would be the comparison of this model's performance with popular classifiers for sequential data, such as convolutional and recurrent neural networks. In addition, extending the classification capabilities of the model to include information concerning the position and function of schemata within phrases could be useful for both learning and discovery of schemata. Lastly, even the current abilities of the schema classifier can be utilized to support content-based retrieval on digital score libraries. The sampling method is independent from the model and, thus, large databases of digital scores can be represented as samples, or even symbolic fingerprints [1]. In addition, on top of a database representation for music scores, a query language for music patterns (as musical schemata) would provide a flexible interface for a user-defined content-based music information retrieval.

7. ACKNOWLEDGEMENTS

This research has been funded by the Faculty of Computing, Engineering and Media at De Montfort University.

8. REFERENCES

[1] Andreas Arzt, Sebastian Böck, and Gerhard Widmer. Fast identification of piece and score position via symbolic fingerprinting. In *13th International Society for Music Information Retrieval*, pages 433–438, Porto, Portugal, 2012.

[2] Vasili Byros. Meyer's Anvil: Revisiting the Schema Concept. *Music Analysis*, 31(3):273–346, 2012.

[3] Emilios Cambouropoulos, Andreas Katsiavalos, and Costas Tsougras. Idiom-independent harmonic pattern recognition based on a novel chord transition representation. In *3rd International Workshop on Folk Music Analysis*, Amsterdam, Netherlands, June 2013.

[4] Darrell Conklin and Ian H Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995.

[5] Michael Scott Cuthbert and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. Utrecht, Netherlands, August 2010.

[6] Katrien Foubert, Tom Collins, and Jos De Backer. Impaired maintenance of interpersonal synchronization in musical improvisations of patients with borderline personality disorder. *Frontiers in Psychology*, 8, 2017.

[7] Robert Gjerdingen. *A classic turn of phrase: Music and the psychology of convention*. University of Pennsylvania Press, 1988.

[8] Robert Gjerdingen. Using connectionist models to explore complex musical patterns. *Computer Music Journal*, pages 67–75, 1989.

[9] Robert Gjerdingen. Categorization of musical patterns by self-organizing neuronlike networks. *Music Perception: An Interdisciplinary Journal*, 7(4):339–369, 1990.

[10] Robert Gjerdingen. *Music in the galant style*. Oxford University Press, 2007.

[11] David Huron. Tone and voice: A derivation of the rules of voice-leading from perceptual principles. *Music Perception: An Interdisciplinary Journal*, 19(1):1–64, 2001.

[12] Carol L Krumhansl. *Cognitive foundations of musical pitch*, volume 17. Oxford University Press, New York, 1990.

[13] Fred Lerdahl and Ray Jackendoff. *A generative theory of tonal music*. MIT press, 1985.

[14] David Lewin. *Generalized musical intervals and transformations*. Oxford University Press, 2007.

[15] Alan Marsden. Schenkerian analysis by computer: A proof of concept. *Journal of New Music Research*, 39(3):269–289, 2010.

[16] Bryan Pardo and William P Birmingham. Algorithms for chordal analysis. *Computer Music Journal*, 26(2):27–49, 2002.

[17] David E Rumelhart. Schemata: The building blocks of cognition. In *Theoretical issues in reading comprehension*, pages 33–58. Lawrence Erlbraum Associates, 1980.

[18] James Symons. Temporal regularity as a key to uncovering statistically significant schemas in an eighteenthcentury corpus. In *annual meeting of the Society for Music Theory, New Orleans, LA, November*, 2012.

[19] Tillman Weyde. Modelling cognitive and analytic musical structures in the musitech framework. In *UCM 2005 5th Conference" Understanding and Creating Music", Caserta*, pages 27–30, 2005.

[20] Tillman Weyde. Automatic semantic annotation of music with harmonic structure. 2007.

Session B

EVOLUTION OF THE INFORMATIONAL COMPLEXITY OF CONTEMPORARY WESTERN MUSIC

Thomas Parmer, Yong-Yeol Ahn
School of Informatics, Computing, and Engineering
Indiana University, Bloomington, IN, USA
tjparmer@indiana.edu, yyahn@iu.edu

ABSTRACT

We measure the complexity of songs in the Million Song Dataset (MSD) in terms of pitch, timbre, loudness, and rhythm to investigate their evolution from 1960 to 2010. By comparing the Billboard Hot 100 with random samples, we find that the complexity of popular songs tends to be more narrowly distributed around the mean, supporting the idea of an inverted U-shaped relationship between complexity and hedonistic value. We then examine the temporal evolution of complexity, reporting consistent changes across decades, such as a decrease in average loudness complexity since the 1960s, and an increase in timbre complexity overall but not for popular songs. We also show, in contrast to claims that popular songs sound more alike over time, that they are not more similar than they were 50 years ago in terms of pitch or rhythm, although similarity in timbre shows distinctive patterns across eras and similarity in loudness has been increasing. Finally, we show that musical genres can be differentiated by their distinctive complexity profiles.

1. INTRODUCTION

Our everyday life is surrounded by cultural products; we wake up to a song, read a book on the subway, watch a movie with friends, or even travel far to admire a piece of art. Despite such pervasiveness, we cannot fully explain why we like a particular song over others or what makes something a great piece of art. Although the perceived quality of a piece is affected by numerous contextual factors, including one's cultural, social, and emotional background, theories suggest that preference, or 'hedonistic value', may also be affected by innate properties of the products, such as *novelty* and *complexity* [1, 18, 28]. In particular, a popular theory suggests there is a *Goldilocks principle* — that just the right amount of novelty or complexity elicits the largest amount of pleasure, whereas pieces with too little or too much complexity are less interesting and enjoyable [3]. On the other hand, cultural products are also *fashionable* — what is popular now may be completely out of fashion next month. Such seemingly contrasting observations prompt us to ask the following questions: *as fads come and go, is there still a consistent preference towards the optimal amount of complexity in cultural products? How has the complexity of contemporary cultural products changed over time?*

This question may apply to any type of cultural product, but we focus here on the complexity of contemporary Western songs. Although various studies have already reported evidence of the 'inverted U-shaped' relationship between perceived complexity and the pleasantness of music in terms of individual-level preference [1, 11, 32], evidence of this preference at the population-level is unclear [7, 22, 30], and many past studies have been limited by the size or extent of the data, in terms of genres or temporal range.

Recently, datasets such as the Million Song Dataset (MSD) began to allow researchers to systematically analyze patterns in music at a massive scale [6, 8, 26]. For example, Serra *et al.* used musical 'codewords' based on song segments in the MSD to identify changes in pitch, timbre, and loudness over time, finding that newer songs restrict pitch transitions, homogenize timbre, and increase loudness (without increasing the variability in loudness) [26]. Mauch *et al.* used a corpus of 17,000 songs from the Billboard Hot 100 to analyze how popular music has evolved between 1960 and 2010 in the United States; using timbral and harmonic features derived from songs on the Hot 100, they identified three stylistic revolutions that occurred in 1964, 1983, and 1991 [15].

In this paper, we analyze the large-scale evolution of complexity in contemporary music in terms of pitch, loudness, timbre, and rhythm, during the period from 1960 to 2010 using the Million Song Dataset (MSD) [4]. We find that complexity does seem to constrain popularity, as evidenced by the most popular songs (those on the Billboard Hot 100) clustering around average values and exhibiting smaller variance compared to a random sample. However, complexity values do fluctuate over time, as long-term trends are seen in loudness, timbre, and rhythm complexity and in the similarity between songs on the Billboard Hot 100. Finally, we compare the complexity of different genres and find that genres have characteristic complexity profiles, leading us to hypothesize that complexity may be a factor in an individual's musical selection.

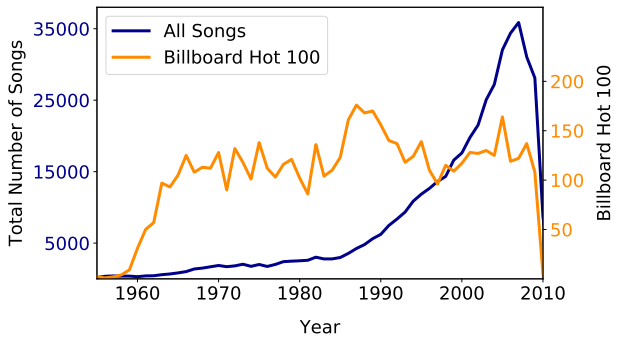


Figure 1. The number of songs per year. 'All Songs' refers to the filtered MSD dataset, while 'Billboard Hot 100' refers to those songs whose title and artist we matched with songs on the Hot 100 as identified in [15].

2. METHODS

2.1 Data

The MSD is a dataset of one million songs created by Columbia University's LabROSA in collaboration with The Echo Nest [4]. Each song in the dataset is divided into small temporal segments (based on note onsets) with detailed data derived from the song's audio signal, and includes metadata such as title, artist, year, duration, and genre terms.

Prior to analysis, we filtered the MSD to remove duplicates, songs with missing genre or duration metadata, and songs likely to be commentary pieces (whose title included the tokens 'interview', 'commentary', 'introduction', 'discuss', 'conference', or 'intro'), resulting in a dataset of 905,896 songs. Some songs also did not have all data types — pitch, loudness, timbre, rhythm, or year — that we examine here and were thus left out of the corresponding calculations. Due to a limited amount of data from the early years, we restricted our analysis to the period from 1960 to 2010. The genre of each song was determined by the term (Echo Nest tag) with the strongest weight, although we note that terms are assigned at the artist level so all songs by the same artist are grouped into the same genre.

To discover the most popular musical pieces in our dataset, we found 6,661 songs which charted on the Billboard Hot 100 as identified in a previous study [15]. The number of songs per year in our final dataset is shown in Figure 1.

2.2 Codewords

To estimate complexity, we defined "codewords" for each song across four dimensions (pitch, loudness, timbre, and rhythm), similar to a previous study [26]. Each codeword is based on a segment of the song. Pitch and timbre codewords are vectors containing the pitches (based on the binary presence of each of 12 pitches in the chromatic scale) and timbres (based on analysis of the audio signal, with 11 components thresholded into three bins) present in the segment. Loudness codewords are equal to the binned maximum decibel value of the segment. Similarly, rhythm codewords are defined as the number of average sixteenth notes between segments, where the average sixteenth note is based on the time signature. We then defined a measure of complexity for each feature per song based on the conditional entropy of each type of codeword. [1]

2.3 Measuring Complexity

Although many studies have examined the relationship between the complexity of a piece and the derived pleasure from it [1, 19–21, 31, 32], there is no universally adopted way to measure the complexity of a song. Existing definitions of complexity include hierarchical complexity, dynamic complexity, information-theoretic complexity, and cognitive complexity [10,23,27,34]. Information-theoretic measures are attractive because they capture the surprise inherent in a pattern, such as the notes played in a musical piece. Theories propose that music can be understood as the kinetics of expectation and surprise, and that composers seek to elicit emotions by fulfilling or denying these expectations [1, 17, 35]. In particular, Implication-realization (IR) theory posits that open intervals evoke expectations in a listener and the surprises of these expectations may be related to complexity [18, 32, 35].

Information-theoretic measures include Shannon entropy, joint entropy, conditional entropy, compression or algorithmic complexity [10, 16, 27, 29], and more complicated techniques such as pairwise predictability between time series, Hidden Markov Models, Normalized Compression Distance, and predictive information rate [1, 8, 9]. Previous studies have used information-theoretic quantities to estimate perceived complexity, identify piece similarity, derive psycho-acoustic features, and classify genres [5, 10, 13, 25, 27, 30].

We use conditional entropy as our measure of complexity, dependent on the immediately preceding symbol, as it is known that events during even short preceding intervals are enough to evoke strong expectations in the listener [1, 12, 35]. Other information-theoretic measures are either more complicated (e.g. predictive information measures), can only be approximated in practice (e.g. Kolmogorov or algorithmic complexity), or do not take past information into account (e.g. Shannon entropy).

Each song was assigned a complexity value for pitch, loudness, timbre, and rhythm, which is equal to the conditional entropy of the feature codewords:

$$H(Y|X) = \sum_{x \epsilon X} p(x) H(Y|X = x) = \\ - \sum_{x \epsilon X} p(x) \sum_{y \epsilon Y} p(y|x) \log_2 p(y|x) \quad (1)$$

where X and Y are possible codewords, $p(x)$ is the probability of observing codeword x and $p(y|x)$ is the probability of observing codeword y given the previous codeword x.

[1] Code is available at https://github.com/tjparmer/music-complexity.

3. RESULTS

3.1 Complexity and Popularity

The complexity distribution of songs is approximately bell-shaped, although timbre is skewed towards zero complexity — unlike the other features, timbre becomes easily predictable after only one previous codeword. The distributions of the Hot 100 songs are similar, although the Hot 100 tends to exhibit statistically lower complexity in pitch and timbre and higher loudness complexity, compared to 95% confidence intervals of 1,000 bootstrap random samples of the same size (see Fig. 2). Furthermore, we found that the variances of the Hot 100 complexity values are smaller than for other songs (based on 95% confidence intervals of 1,000 bootstrap samples from the Hot 100 compared to 1,000 bootstrap samples from the overall distribution); thus, the popular songs tend to be located in a narrower range near the mean across pitch, loudness, and rhythm complexity. This result supports the theory for an inverted U-shaped curve where global popularity is maximized by medium complexity.

3.2 Complexity Across Time

To examine the evolution of song complexity, we calculate the mean complexity values for each year (for all songs and the Hot 100 songs separately) in Fig. 3, which shows several long-term trends. Later years mark the appearance of songs with low loudness and rhythm complexity and songs with high timbre complexity, but they were not reflected strongly in the Hot 100 songs. The low loudness complexity may be due to the trend often called the "loudness war" [26], which describes the tendency to produce the entire song to be as loud as possible. Another possible reason may be the emergence of low complexity genres in recent years. For instance, terms associated with low loudness complexity outliers include 'grindcore', 'hip hop', and 'black metal', all of which are relatively newer genres in the dataset. Low rhythm and high timbre complexity may be due to pop or electronic music that contain modern production techniques with many different synthesized textures and strong dance beats. Terms associated with low rhythm outliers include 'tech house', 'techno', and 'hard trance', while terms associated with high timbre complexity outliers include 'tech house', 'techno', and 'deep house'.

Previous research has indicated that the evolution of Western popular music experienced significant changes during three musical 'revolutions' in 1964, 1983, and 1991 [15]. The first was associated with rock and soul music, the second with disco, new wave, and hard rock, and the third with the emerging popularity of rap music over rock music. These three revolutions split our period of analysis into three 'epochs': 1964-1983, 1983-1991, and 1991-2010. With this reference frame, we examine our measures of complexity.

If we consider the entire dataset, each aspect of complexity shows a different pattern. The pitch complexity has been more or less stable across the whole period; the loudness complexity has been decreasing overall, although the period from 1983 to 1991 shows a slight increase; the timbre complexity has been steadily increasing and reached a plateau after the 1990s; finally, the rhythm complexity was decreasing through the period from 1964 to 1983, and then stabilized.

Meanwhile, we find that the temporal evolution of the Hot 100 songs does not follow the overall pattern. The largest difference can be observed in the timbre complexity. While the timbre complexity of the entire dataset has been steadily increasing, it has been almost completely flat for the most popular songs, diverging from the overall trend. This may indicate that the emergence of new genres with high timbre complexity primarily happened for more niche musical tastes. Pitch and loudness complexity, by contrast, have been higher for popular songs in recent years, while rhythm complexity was lower until the 2000s.

3.3 Popular Song Similarity

The analysis of complexity over time suggests that modern day popular songs (at least from the 2000s) are more likely to have higher pitch, loudness, and rhythm complexity (and lower timbre complexity) than their less popular contemporaries. However, while this suggests that popular songs are not simpler than the average song, it does not necessarily indicate whether they sound more or less similar to their popular contemporaries (that is, other Hot 100 songs that are released in the same year). A recent report suggests that popular songs are sounding more and more similar to other songs on the charts [33]. In contrast, other research finds that songs that perform well on the charts do not sound too similar to their contemporaries but often have an optimal level of differentiation [2]. To analyze whether popular songs become more similar to their contemporaries over time, we measure the Kullback-Leibler (KL) divergence [14] from each song in the Hot 100 to other popular songs that were released in the same year. KL divergence captures the unexpectedness of a song's codewords given the codewords present in other popular songs and thus indicates the spread of codeword usage per year.

In Fig. 4, we show the KL divergence per year for each feature, with each epoch marked. Larger KL divergence in the figure suggests that the songs in that year are more different from their contemporaries as compared to other years. Our measurement shows that the 1964-1983 and 1991-2010 period are similar to each other while 1983-1991 shows a reversing trend. Across features, the KL divergence was either decreasing (songs are more similar to each other) or stable during 1964-1983 and 1991-2010, while 1983-1991 marked either a positive trend reversal or a slow-down of the decreasing trend. These changing trends suggest that musical revolutions are not born equal; some may have spurred diversity among popular songs while some may have homogenized the field.

Despite these fluctuations, the divergence trend is roughly stable over time for pitch and rhythm, while timbre rebounds after similarity decreases; thus, our findings are consistent with research that songs that perform well

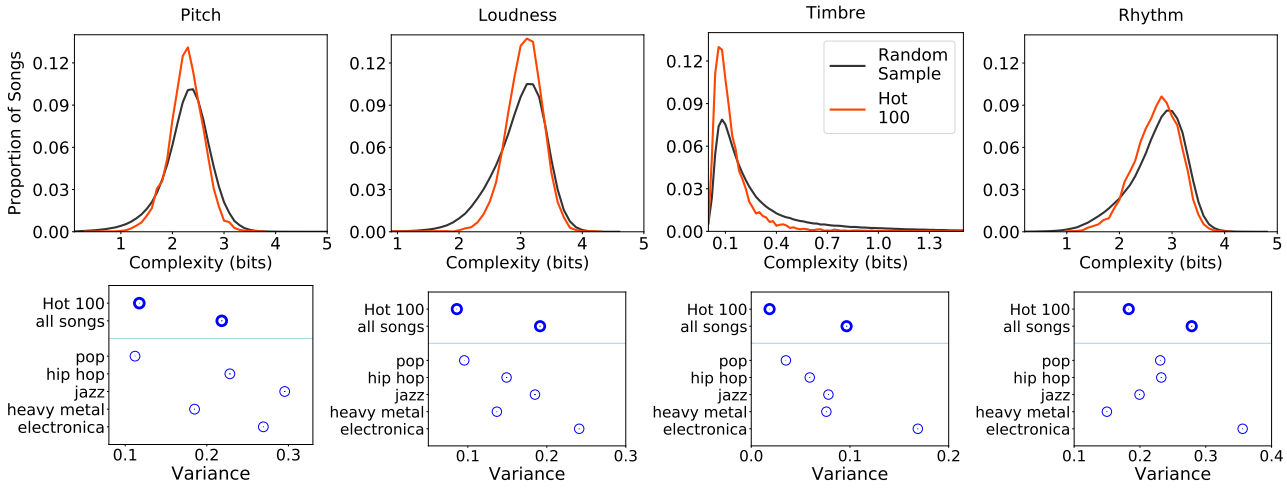


Figure 2. Feature complexities and variances. Complexity distributions are shown (in bins of 0.1 bits, except for timbre which is in bins of 0.02 bits). The variance plots include 95% confidence intervals in black (although confidence intervals are smaller than the symbol and not visible), based on 1000 bootstrap samples of 1000 songs from the respective genre.

on the charts do not sound too similar to their contemporaries but rather maintain a degree of uniqueness which is statistically consistent over time.

3.4 Complexity Across Genres

Let us turn our attention to musical genres and their complexity. As some genres may be characterized by complex harmonic structures or simple, repeated patterns, we expect to see differences across different genres in terms of complexity. For example, jazz is often considered to have complex patterns whereas dance music may be assumed to use simpler rhythmic patterns. Our measurement concurs with such speculation, but finds that different subsets of genres may be relatively complex across one or two features but not others. For instance, electronic and dance styles tend to have high pitch complexity values, whereas jazz and blues have high loudness complexity values. The highest timbre complexity values belong to electronic genres, although metal also scores highly, but electronic genres have reduced rhythmic complexity which is instead maximal in jazz, progressive and vocal genres.

We found that a variety of common genres were significantly different from a random sample drawn from the overall distribution in terms of each feature complexity (based on a two-sample Kolmogorov-Smirnoff nonparametric test as well as 95% confidence intervals of the means of each feature), with the exception that pop was not rhythmically distinct. Thus each genre seems to have distinctive complexity features that describe its songs: jazz is relatively complex (except in terms of timbre), hip hop has higher than average pitch and loudness complexity, heavy metal has high rhythm complexity but low pitch and loudness complexity, and electronica has high timbre complexity but low rhythm complexity (Fig. 6).

This pattern may be indicative of some trade-offs that listeners make. If they prefer timbre at the expense of rhythmic complexity, they may prefer electronic genres. If they prefer pitch and loudness complexity, they may prefer hip hop or jazz. If they care about timbre and rhythm over pitch and loudness, they may prefer heavy metal. There is a positive correlation between pitch and loudness complexity (Pearson's r=0.77) across all songs, suggesting that genres tend to have high pitch and loudness complexity (e.g. hip hop, jazz) or low pitch and loudness complexity (e.g. heavy metal). There is also a negative correlation seen between timbre and rhythm complexity (Pearson's r=-0.55), suggesting that rhythmic complexity decreases with higher timbre complexity (although this is not true for metal genres).

Interestingly, the Hot 100 is similar to the pop genre in feature means and variances (although statistically different). Both pop music (whose songs are given no genre-specific term with higher weight than 'pop') and the Hot 100 (whose songs are primarily classified as genres other than 'pop') have near average values of pitch, loudness, and rhythm complexity, and lower than average values of timbre complexity, while also having smaller variance than the other selected genres (refer to Figures 2 and 6). This may suggest that listeners expect the same from listening to the Hot 100 as they do when listening to music labeled as 'pop': mildly surprising songs that do not vary too much in complexity and which are sonically predictable.

One may also expect similar genres to share similar complexity scores. We used agglomerative clustering on genres represented with over 5000 songs in the dataset (a total of 41 genres), using Euclidean distance between the genre mean complexity scores of each feature, and then used the silhouette coefficient [24, 28] to find nine optimal communities. The result matches intuitive expectations, such that rock genres are grouped together as are electronic genres; interestingly jazz is grouped with hip hop and rap, due to these genres having similar complexity scores. These results suggest that a genre may be defined, to some degree, by its pitch, loudness, timbre, and rhythmic complexity.

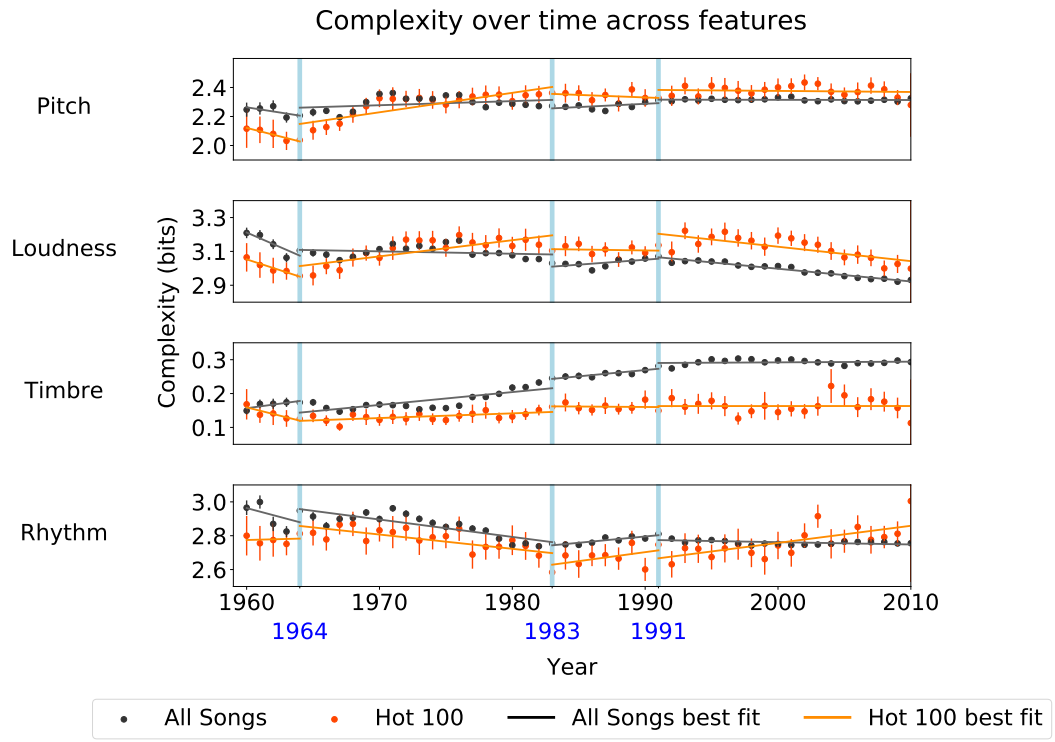


Figure 3. Average complexity over time across features. Linear trend lines (obtained using OLS linear regression) are shown for each epoch, along with 95% confidence intervals of the mean. Light blue lines indicate the musical revolutions found in [15].

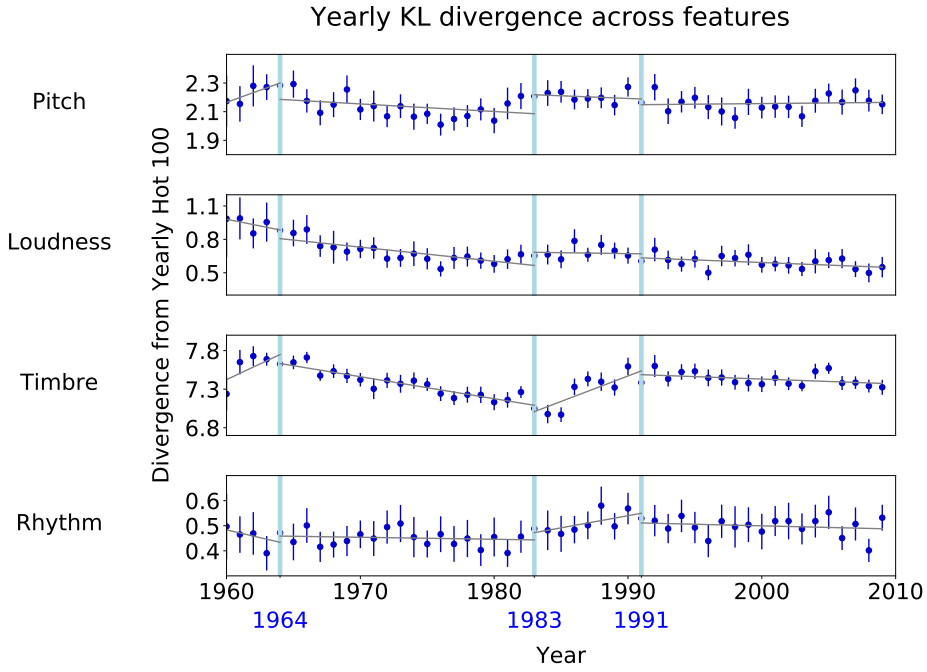


Figure 4. Yearly KL divergence of Hot 100 songs across features. Complexity variance, as measured by KL divergence between a song's codewords and codewords from other songs released in that year, shows changing trends at the time of certain musical 'revolutions' in 1964, 1983, and 1991 (indicated by light blue lines). Linear trend lines (obtained using OLS linear regression) are shown for each epoch, along with 95% confidence intervals of the mean.

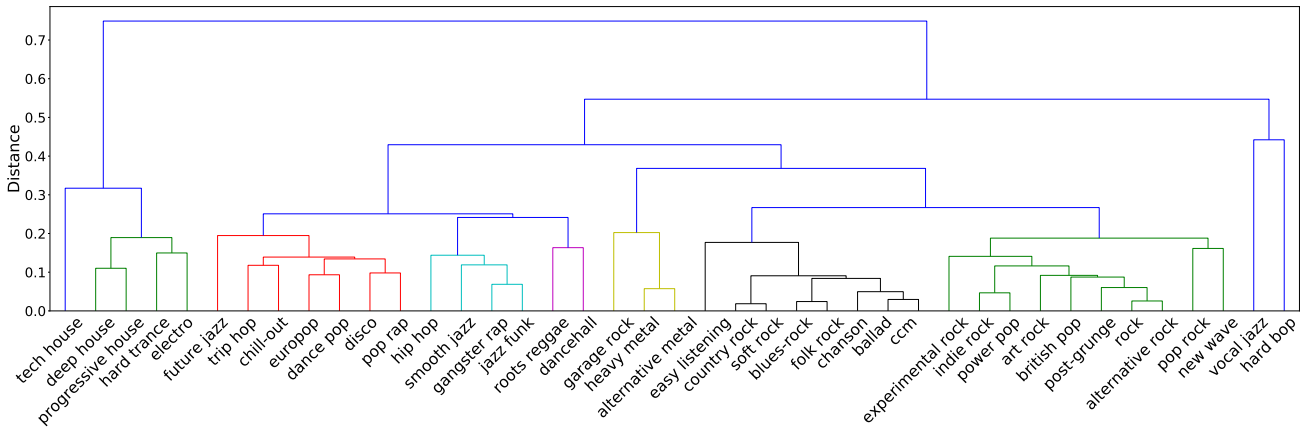


Figure 5. Dendrogram of top genres. Clustering is based on the Euclidean distance between complexity features. Colors indicate different communities.

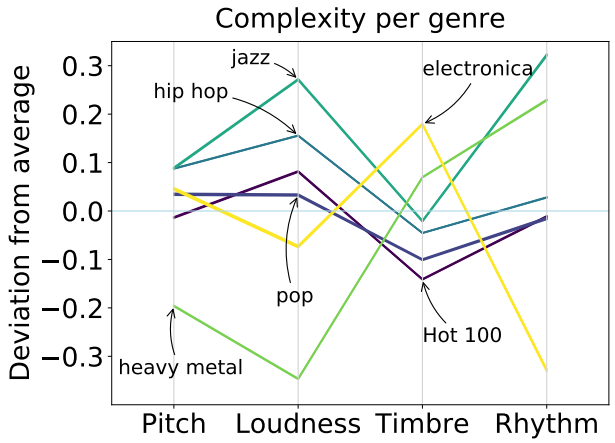


Figure 6. Complexity of select genres across features. For each feature, the deviation of the genre complexity mean from the mean of 100 bootstrapped samples of the same size as the genre is shown. Note the similarity between the Hot 100 and the pop genre.

4. DISCUSSION

Understanding and characterizing the complexity of music is an important area of study with both cultural and economic significance. Although music may seem complicated, songs quickly become predictable as you take previous knowledge into account. This suggests that conditional entropy may be a useful way to characterize musical complexity, although our approach here assumes that the uniform distribution of codewords is the state of maximum uncertainty and expectations are made based on only one previous symbol, which cannot distinguish counts of repeating patterns or phrasing [1]. Our approach is thus intrinsic to the song itself and ignores any *a priori* contextual information.

Using this measure, we find that pitch complexity has been generally stable over the period from 1960 to 2010, while loudness and rhythm complexity have decreased and timbre complexity has increased. Complexity norms seem to constrain the most popular songs, as those on the Hot 100 are distributed around the overall feature means with small variance in complexity, the exception being timbre which is lower than average. Indeed, the Hot 100 is similar to songs labeled as 'pop', in that pop also has average pitch, loudness, and rhythm complexity and low variance.

This result provides evidence of a global, inverted U-shaped relationship between popularity and complexity, where popular songs are, on average, the most pleasant to the population. Listeners may expect popular songs to be mildly complex, but not to deviate far from expected timbre or complexity norms. Complexity of the Hot 100 has in fact been consistent over fifty years in pitch and timbre, while increasing recently in rhythm and decreasing in loudness. Similarly, popular songs continue to maintain a consistent level of differentiation from their contemporaries in terms of pitch, timbre, and rhythm.

Certain genres do differ significantly across complexity features, suggesting that they have specific complexity profiles that help define them. We hypothesize that certain genres may 'make up' for lack of complexity in one area by increased complexity in another. Perhaps fans of electronic genres prioritize complexity in timbre but rhythmically simple dance beats, or metal fans prioritize rhythmic complexity and high volume at the expense of loudness complexity.

More research needs to be done to fully elucidate the relationship between complexity and musical appreciation. For example, future research can relate musical complexity to the listening habits of people on a large scale to determine a more fine-grained measure of song popularity. The consistency of popular songs over time suggests that, collectively, people tend towards songs that are a certain optimal level of complexity rather than being too simple or complicated. However, it remains an open question to what degree complexity plays a role in people's cognitive appreciation of music.

5. REFERENCES

[1] Samer Abdallah and Mark Plumbley. Information dynamics: patterns of expectation and surprise in the per-

ception of music. *Connection Science*, 21(2-3):89–117, 2009.

[2] Noah Askin and Michael Mauskapf. What makes popular culture popular? Product features and optimal differentiation in music. *American Sociological Review*, 82(5):910–944, 2017.

[3] Daniel E Berlyne. Novelty, complexity, and hedonic value. *Perception & Psychophysics*, 8(5):279–286, 1970.

[4] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, pages 591–596, 2011.

[5] Rudi Cilibrasi, Paul Vitányi, and Ronald de Wolf. Algorithmic clustering of music based on string compression. *Computer Music Journal*, 28(4):49–67, 2004.

[6] Sander Dieleman, Philémon Brakel, and Benjamin Schrauwen. Audio-based music classification with a pretrained convolutional network. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, pages 669–674, 2011.

[7] Tuomas Eerola and Adrian C North. Expectancy-based model of melodic complexity. In *Proceedings of the Sixth International Conference on Music Perception and Cognition*, 2000.

[8] Peter Foster, Simon Dixon, and Anssi Klapuri. Identifying cover songs using information-theoretic measures of similarity. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(6):993–1005, 2015.

[9] Peter Foster, Anssi Klapuri, and Simon Dixon. A method for identifying repetition structure in musical audio based on time series prediction. In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 1299–1303. IEEE, 2012.

[10] Barbra Gregory. *Entropy and complexity in music: some examples*. PhD thesis, University of North Carolina at Chapel Hill, 2005.

[11] Ronald G Heyduk. Rated preference for musical compositions as it relates to complexity and exposure frequency. *Perception & Psychophysics*, 17(1):84–90, 1975.

[12] David Brian Huron. *Sweet anticipation: Music and the psychology of expectation*. MIT press, 2006.

[13] Heather D Jennings, Plamen Ch Ivanov, Allan de M Martins, PC da Silva, and GM Viswanathan. Variance fluctuations in nonstationary time series: a comparative study of music genres. *Physica A: Statistical Mechanics and its Applications*, 336(3-4):585–594, 2004.

[14] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[15] Matthias Mauch, Robert M MacCallum, Mark Levy, and Armand M Leroi. The evolution of popular music: USA 1960–2010. *Royal Society open science*, 2(5):150081, 2015.

[16] David Meredith. Music analysis and kolmogorov complexity. In *Proceedings of the 19th Colloquio d'Informatica Musicale (XIX CIM)*, 2012.

[17] Leonard B Meyer. *Music, the Arts and Ideas: Patterns and Predictions in Twentieth-century Culture*. University of Chicago Press, 1967.

[18] Eugene Narmour. *The analysis and cognition of basic melodic structures: The implication-realization model.* University of Chicago Press, 1990.

[19] Adrian C North and David J Hargreaves. Experimental aesthetics and everyday music listening. In *The social psychology of music*, pages 84–103. Oxford University Press, Oxford, 1997.

[20] Adrian C North and David J Hargreaves. Liking, arousal potential, and the emotions expressed by music. *Scandinavian journal of psychology*, 38(1):45–53, 1997.

[21] Mark G Orr and Stellan Ohlsson. Relationship between complexity and liking as a function of expertise. *Music Perception: An Interdisciplinary Journal*, 22(4):583–611, 2005.

[22] Robert Mitchell Parry. Musical complexity and top 40 chart performance. Technical report, Georgia Institute of Technology, 2004.

[23] Jeffrey Pressing. Cognitive complexity and the structure of musical patterns. In *Proceedings of the 4th Conference of the Australasian Cognitive Science Society*, 1999.

[24] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[25] Eric D Scheirer, Richard B Watson, and Barry L Vercoe. On the perceived complexity of short musical segments. In *Proceedings of the 2000 International Conference on Music Perception and Cognition*, 2000.

[26] Joan Serrà, Álvaro Corral, Marián Boguñá, Martín Haro, and Josep Ll Arcos. Measuring the evolution of contemporary western popular music. *Scientific reports*, 2:521, 2012.

[27] Ilya Shmulevich and D-J Povel. Measures of temporal pattern complexity. *Journal of New Music Research*, 29(1):61–69, 2000.

[28] Higor YD Sigaki, Matjaž Perc, and Haroldo V Ribeiro. History of art paintings through the lens of entropy and complexity. *Proceedings of the National Academy of Sciences*, 115(37):E8585–E8594, 2018.

[29] S.J. Simon. *A multi-dimensional entropy model of jazz improvisation for music information retrieval*. PhD thesis, University of North Texas, 2005.

[30] Dean Keith Simonton. Computer content analysis of melodic structure: Classical composers and their compositions. *Psychology of Music*, 22(1):31–43, 1994.

[31] Loren Steck and Pavel Machotka. Preference for musical complexity: Effects of context. *Journal of Experimental Psychology: Human Perception and Performance*, 1(2):170, 1975.

[32] Sebastian Streich et al. *Music complexity: a multi-faceted description of audio content*. Universitat Pompeu Fabra Barcelona, Spain, 2006.

[33] Andrew Thompson and Matt Daniels. The musical diversity of pop songs, 2018. Available from: https://pudding.cool/2018/05/similarity/.

[34] Eric Thul and Godfried T Toussaint. Rhythm complexity measures: A comparison of mathematical models of human perception and performance. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, pages 663–668, 2008.

[35] Pablo H Rodriguez Zivic, Favio Shifres, and Guillermo A Cecchi. Perceptual basis of evolving western musical styles. *Proceedings of the National Academy of Sciences*, 110(24):10034–10038, 2013.

DEEP UNSUPERVISED DRUM TRANSCRIPTION

Keunwoo Choi
Spotify
keunwooc@spotify.com

Kyunghyun Cho
New York University, Facebook AI Research
kyunghyun.cho@nyu.edu

ABSTRACT

We introduce DrummerNet, a drum transcription system that is trained in an unsupervised manner. DrummerNet does not require any ground-truth transcription and, with the data-scalability of deep neural networks, learns from a large unlabeled dataset. In DrummerNet, the target drum signal is first passed to a (trainable) transcriber, then reconstructed in a (fixed) synthesizer according to the transcription estimate. By training the system to minimize the distance between the input and the output audio signals, the transcriber learns to transcribe without ground truth transcription. Our experiment shows that DrummerNet performs favorably compared to many other recent drum transcription systems, both supervised and unsupervised.

1. INTRODUCTION

Transcription is a music information retrieval task with the goal of estimating the score y when input audio x is given. The majority of the recent transcription systems is based on supervised learning, where the transcriber is an *analysis* system $\hat{y} = F_a(x)$ that is trained with annotated pairs $\{(x_m, y_m)\}_{m=1}^M$ to minimize the distance between y and $\hat{y}$ [6, 7, 27, 31, 33, 34, 37, 38].

The trend is similar in drum transcription on which we focus in this paper. Supervised learning approaches may incorporate models based on frame-based feature extraction and classification [15], non-negative matrix factorization (NMF) for pattern matching [10], or hidden-Markov model [25]. More attention has been given recently to deep learning based models such as convolutional neural networks (CNNs, [13, 34]) and recurrent neural networks (RNNs, [33, 37, 38]), all of which have greatly improved transcription systems.

However, the lack of a large-scale annotated dataset is one of the most frequently mentioned obstacles that hinder further improvement. In practice, this limits the generalizeability of supervised learning systems, as will be discussed in Section 4, and using synthetic data is one way to address this issue [7, 39]. Although there have been proposals to use unlabeled data [42, 43], the issue remains as they still rely on supervised learning combined with teacher-student learning [16]. Parallel to those approaches, an annotation-free and, therefore, a more scalable and generalizable alternative would be unsupervised learning.

Unsurprisingly, one of the humans' music learning procedures, self-taught by trial-and-error, is very similar to unsupervised learning. For example, musicians learn to transcribe by (a) listening, (b) playing an instrument, (c) identifying differences, and (d) making adjustments. *Can this be done without any supervision? Yes*, if the person can spot the pitch difference (e.g., the pitch should be higher or lower). Consistent with this logic, developing a transcription system based on unsupervised learning would be feasible if the system can test the estimation, measure the error, and correct itself accordingly.

To implement such an unsupervised transcription system, we need a *synthesis* system, $\hat{x} = F_s(\hat{y})$, making the overall system $\hat{x} = F_s(F_a(x))$. During its training, the system is given $\{x\}_{m=1}^M$ and trained to minimize the distance between x and $\hat{x}$. There have been few systems relying on unsupervised learning as explained above. In MIR, the system in [1] utilized sparse coding to learn a dictionary of time-frequency templates of piano and harpsicord, assuming a (matrix-)multiplication model with additive noise, $F_s(\mathbf{y}) = \mathbf{A}\mathbf{y}+\mathbf{e}$. Yoshii et al. proposed to use sparse coding in a jointly-learned chord recognition and transcription system [44]. Berg et al. designed a probabilistic graphical model that parameterizes the spectral and temporal envelopes, note events, and note activations, in order to transcribe piano by inferring their parameters [2]. In drum transcription, many systems have used NMF to decompose a drum track spectrum into spectral templates and their temporal activations (or transcription) [26, 41]. Several variants of NMF were proposed to address the limits of the fixed spectrum template of NMF [19, 20, 29]. Lastly, a similar system can be found in computer vision, where the parameters of input images are estimated by reconstruction using an image renderer [18].

In this paper, we introduce DrummerNet, a deep neural network based drum transcription system that is trained by unsupervised learning. With a more end-to-end approach, DrummerNet is distinguished from previous research [1, 2, 44], which has strong priors on the target sounds. In §2, we present the system design principle behind DrummerNet, followed by its details in §3. In §4, the evaluation results are discussed along with the ablation study. We present our conclusion, the problems of our system, and the future direction towards fully unsupervised learning in transcription/MIR in §5.

Name	Description	Note
n, N	The temporal index/length of audio input	
k, K	The index/total number of drum components	K=11
x, y	Mixture and transcription	$\in \mathbb{R}^N$
$\hat{x}, \hat{y}$	Estimations of mixture/transcription	$\in \mathbb{R}^N$

Table 1: Symbols used in this paper

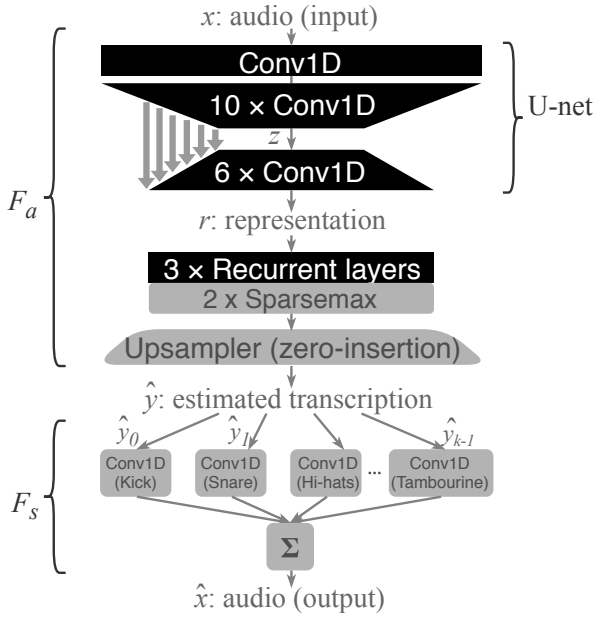


Figure 1: Block diagrams of DrummerNet structure. Trainable modules are illustrated as black boxes and fixed modules as rounded grey boxes.

2. SYSTEM DESIGN PRINCIPLES

Training the proposed DrummerNet is similar to the previous unsupervised learning approaches in music [1, 2, 44], as they all train a system to output $\hat{x}$ that reconstructs the original signal x. The difference between $\hat{x}$ and x works as a proxy of the difference between $\hat{y}$ to y.

There are three conditions under which unsupervised learning of a transcriber can be achieved successfully. First, the output of the analysis module F_a must be in the form of transcription – a set of discrete events representing the timing and intensity of the notes. Second, the synthesis module F_s must synthesize the audio signal given the transcription input $\hat{y}$. Third, all the components in the network must be differentiable as we rely on backpropagation to train it.

3. DRUMMERNET

In this section, we introduce the proposed system structure. We specify the number of channels, kernel size, and stride as `(channel, kernel, stride)`. All the convolutional and recurrent layers use an exponential linear unit as an activation function [9]. [1]

[1] The implementation of DrummerNet is available on `https://github.com/keunwoochoi/DrummerNet`

3.1 Analysis module F_a

The analysis module F_a, as illustrated in the top half of Figure 1, takes the audio signal x as an input and processes it through a series of U-net variant [30], recurrent layers, and gated Sparsemax activation [21]. After training, this module is used as a transcriber (with peak-picking).

U-net The U-net consists of 1D convolutional layers, max-pooling layers, and concatenations between the encoder and the decoder. The encoder consists of a convolutional layer `(128, 3, 1)` followed by 10 convolutional layers `(50, 3, 1)` interleaved with max-pooling of size 2. As a result, it outputs $z \in \mathbb{R}^{N/1024}$ which has a receptive field size of 3,072 time steps.

The decoder has only 6 convolutional layers `(50, 3, 1)` interleaved with a concatenation with the feature map at the same depth as in the encoder and a $\times 2$ bi-linear interpolation. We call the output of decoder $r \in \mathbb{R}^{N/16}$, the representation based on which the transcription is estimated. The asymmetry between the encoder and the decoder makes the length r to be shorter by a factor of $4^2 = 16$ compared to that of input x. Assuming the input audio is sampled at 16 kHz, [2] r would have a sampling rate of 1,000 Hz.

Recurrent layers We use three recurrent layers: (GRUs [8]) {along time-axis, bi-directional, 100-channel}, {along time-axis, uni-directional, 50-channel}, and {along channel-axis, uni-directional, K-channel}. These three recurrent layers have properties of i) being bi-directional so that the onset at n can be determined by the vicinity of n (both the past and the future), ii) enforcing temporal dependency, and iii) enforcing component-wise dependency, respectively. The width (or the hidden vector size) of the third recurrent layer is equal to K, the number of drum components in the synthesizer, to map each channel to each drum component.

Sparsemax In an ideal case of transcription, there would be local sparsity along both the time and channel-axes because the drum events would not repeat with a rate of 1,000 Hz (which is faster than 16-beat on 240 BPM), nor would all the K drum components be activated simultaneously. Although sparsity is one of the properties that *can* be achieved by the autoregressive nature of the recurrent layers, we add Sparsemax [21] activation to encourage it explicitly. The output of Sparsemax has two important properties: i) it always sums to 1 (same as Softmax) and ii) it is highly likely to be sparse with actual zeros (unlike Softmax). In DrummerNet, two Sparsemax layers are applied in parallel, one along channel-axis (=instrument-axis) and the other time-axis within a non-overlapping window size of 64. This design choice is based on the assumption that there are only a few onsets among notes (channel-axis sparsity) and within 64 samples at $\hat{y}$, or 64 ms (temporal sparsity). The outputs from these two Sparsemax layers are then multiplied element-wise.

[2] This is the sampling rate of input audio in our experiment.

Class	Subclass	Description
KD	KD	Kick drum
SD	SD	Snare drum
HH	CHH, PHH	Closed/pedalled hi-hat
	OHH	Open hi-hat
TT	HIT, MHT,	High/high-mid/
	HFT, LFT*	high-floor/low-floor tom
CY	RDC, RDB*	Ride cymbal, ride cymbal bell
	CRC, CHC*,	Crash/china cymbal
	SPC*	splash cymbal
OT	SST*, TMB	side stick, tambourine

Table 2: A drum component hierarchy [36]. The synthesizer F_s consists of 11 classes, following Subclass of the table with omitting ones marked with asterisks *.

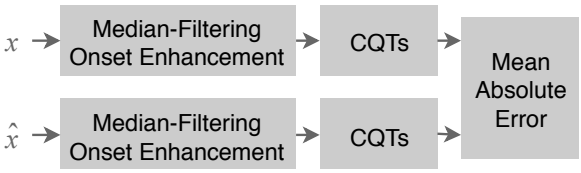


Figure 2: The block diagrams of loss calculation

Upsampler Finally, the low temporal resolution of the Sparsemax output is addressed by zero-insertion upsampling by the factor of 16. According to this, we modify the temporal quantization rate of events, unlike the upsampling of a digital signal.

3.2 Synthesis module F_s

The synthesis module F_s consists of K parallel 1D convolutional layers and a channel-wise summing operator. The kernel of each layer is not trained but fixed to the known waveform of each drum component to convert a transcription of a component $\hat{y}_k$ into a track $\hat{x}_k$. The tracks are summed to generate the final output $\hat{x}$ ($= \sum_{k=1}^{K} \hat{x}_k$), the synthesized audio signal. This module is only used during training.

In the implementation, we use $K = 11$, using Subclass in Table 2, following [36]. Ones marked with asterisks were excluded due to their scarcities in our source of isolated drum recordings, which consisted of 12 virtual drum instruments provided by Logic Pro X. Multiple drum kits, including rock, pop, funk, and soul [3] , were used to prevent the network from overfitting to a specific drum kit. During training, a drum kit was randomly assigned for every batch.

3.3 Learning

Unable to directly compute the transcription loss during unsupervised learning, we carefully designed a loss function at the audio level, $L_x(x, \hat{x})$, as minimizing it would also minimize the transcription loss, $L_y(y, \hat{y})$. To do so,

[3] Brooklyn, Heavy, Liverpool, Neo Soul, Detroit Garage, Motown Revisited, Portland, Sunset, Speakeasy, SoCal, Smash, and Slow Jam. All with velocity=98.

Module	Input (size)	Output (size)
U-net encoder		
Conv1D	x:$(1, N)$	$(C*, N)$
10× Conv1D	$(C*, N)$	$(C*, N/1024)$
U-net decoder		
6× Conv1D	$(C*, N/1024)$	r:$(C*, N/16)$
Recurrent layers	$(C*, N/16)$	$(K, N/16)$
Sparsemax	$(K, N/16)$	$(K, N/16)$
Upsampler	$(K, N/16)$	$\hat{y}$:(K, N)
Synthesis module		
Channel splitter	$\hat{y}$:(K, N)	$K \times \hat{y}_k$:$(1, N)$
Each Conv1D	$\hat{y}_k$:$(1, N)$	$\hat{x}_k$:$(1, N)$
Sum (mixer)	$K \times \hat{x}_k$:$(1, N)$	$\hat{x}$:(K, N)

Table 3: The shapes of inputs/outputs of the module in DrummerNet. $C*$ indicates the number of channels but unspecified.

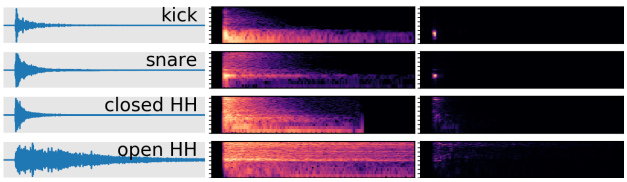


Figure 3: The effect of drum extraction for kick, snare, close hi-hat, and open hi-hat, from top to bottom. Columns are from left to right: original waveform, original spectrum, and onset-enhanced spectrum

L_x should be able to differentiate the drum components – kick drum (KD), snare drum (SD), and hi-hat (HH) – while being invariant to the varying drum kits. Perceptually, there are clear differences between KD, SD, and HH. Although both impulsive, KD is in the low-frequency band while SD is in the mid-frequency band. SD is also relatively tonal and has a longer envelope. HH is more complicated to describe due to its variation from its playing technique. For example, closed and pedalled-HH's are in the high-frequency band, impulsive, and with relatively low energy, while open-HH's are similar except louder with a longer noisy envelope.

We thus define and use *onset spectrum similarity*, which is designed to represent the similarity based on the onset part of sounds in the spectrum domain. As illustrated in 2, it is measured by i) applying median-filtering based drum extraction [12] which enhances onsets (with a FFT size of 1024 and median filter length of 31 on both axes), ii) converting to multi-resolution CQTs (constant-Q transform) for both x and $\hat{x}$, and then iii) calculating the mean absolute difference between them.

Among many spectral magnitude representations, we use (log-magnitude) CQT since the logarithmic frequency scale is known to match well to human auditory perception [23]. We followed the implementation of Pseudo-CQT [4] which multiplies linear-to-octave filterbanks to an STFT. As a result, the CQT covered nearly 8-octave bands

[4] `http://librosa.github.io/librosa/`

from 32.07 Hz (C1) to 8 kHz (the Nyquist frequency of our experiment) with a 12-band/octave resolution. This implementation is differentiable.

Figure 3 shows the effect of onset enhancement. It preserves the characteristics of the drum components in the transient part while removing the after-onset components. This process makes L_x and L_y more similar, as the non-transient parts vary more among drum kits due to their random and noisy nature. In a preliminary experiment, for example, the network tried to reconstruct all the non-transient components of SD using tom-toms and HHs, resulting in non-sparse and severe false-positive detection of onsets.

4. EXPERIMENTS AND ANALYSIS

4.1 Setup

For the training of DrummerNet, we used an in-house dataset of drum stems that are crawled from many websites. The dataset consisted of 3,940 unique tracks averaging 225 seconds each for a total of 249 hours. Since the dataset was crawled from various websites, some details, such as the distribution of drum components, are hard to identify. The tracks were mostly popular western rock/pop music. Alternatives to this in-house dataset can be found in [7] (3,758 drum sample recordings ($\times$8 second = over 8 hours) or 60,000 synthesized drum loops ($\times$8 second = over 133 hours)) and [39] (4,197 drum tracks (259 hours)). We opted for the in-house dataset because it provided more diversity as it was not synthesized.

Each audio file was resampled to 16 kHz and down-mixed to mono. The training batch size was 16, and for each audio file, we randomly selected a 2-second segment. On average, there were 112.5 segments in a track, and therefore training with 443,250 (=3,940 $\times$ 112.5) items would be approximately one epoch. With a Nvidia Tesla P100 and a batch size of 32, it took about 9 hours to train a single epoch. We implemented DrummerNet using Pytorch 1.0 [24] and used Librosa 0.6.3 [22] and Madmom 0.16 [4] for audio processing and peak-picking.

We used a heuristic peak-picking method introduced in [5]. This method selects a peak $\hat{y}[n]$ at n if it satisfies the three conditions in Eq. (1),

$$\begin{aligned} \hat{y}[n] &= \max(x[n-w_m], ..., x[n+w_m]) \\ \hat{y}[n] &\geq \text{average}(x[n-w_a], ..., x[n+w_a]) + \delta \\ n &> n_{lp} + w_w, \end{aligned} \quad (1)$$

where the max window w_m=50 ms, average window w_a=100 ms, threshold δ=0.2, waiting window w_w=50 ms, and n_{lp} is the last detected peak. We mainly use F1 score along with Precision and Recall using mir_eval [28]. The tolerance window is 50 ms.

After training, we test the system on three public datasets: IDMT-SMT-Drums (SMT, 104 drum tracks, total 130 minutes [10]), Medley-DB Drums (MDB, 23 tracks, total 20 minutes [36]), and ENST-drums (ENST, 61 minutes [14], drum-only tracks known as 'wet-mix' of 'minus-one' subset). According to [40], a task is DTD [5] if tracks are drum-only, more precisely KD/SD/HH-only, and the system annotates KD/SD/HH events. This is the case for the SMT dataset. A task with the system annotating KD/SD/HH but with drum tracks consisting of more than those three components, e.g., tom-toms and cymbals, is named DTP [6] in [40]. Following this convention, we evaluate DTD with SMT (Section 4.3), and DTP with MDB/ENST. We did not fine-tune for any dataset in any experiment and used the whole datasets for evaluation only.

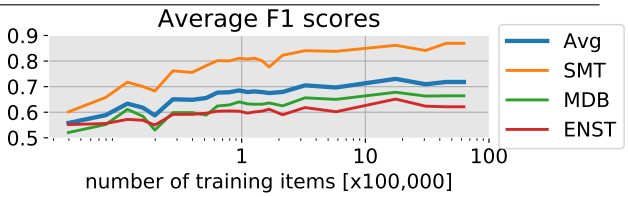


Figure 4: The F1 scores of DrummerNet over training items on each dataset (SMT, MDB, ENST), averaged over KD, SD, and HH. AVG indicates the overall average F1 scores of three datasets.

4.2 Trend of Performance over Training

We did not employ a stopping strategy but trained the network for 6×10^6 items (about 13 epochs). As illustrated in Figure 4, the overall performance gradually increases as the training proceeds and approaches converging towards the end of training. This indicates that the proposed loss function is a good proxy of transcription loss. After the initial phase of training, the performance differences among datasets remain consistent, probably due to the different characteristics of drum tracks in each dataset, as will be discussed in Section 4.4.

4.3 Relative Performance against Baselines

In this experiment, we trained our system on the in-house training set without any annotation and evaluated it on a separate test set (also known as 'eval-cross (trained on DTP)', [40]), which is a stronger condition than a usual train/test split scenario in supervised learning ('eval-subset', [40]). This setup allows us to measure the generalization capabilities across the datasets. Specifically, our experiment is equivalent to DTD, 'eval-cross (trained on DTP)' experiment in [40]. [7], which is only available on SMT. Therefore, only the performances on SMT are compared in this Section. Overall, the performance of DrummerNet is favorable to that of recent drum transcription systems. With an average F1 score of 0.869 on SMT, the proposed unsupervised DrummerNet outperformed 9 out of 10 systems. The nine systems include ones with deep neural networks and supervised approach (ReLUts, RNN, lstmpB, tanhB, and GRUts [33,34,37,38]), as well as ones with NMF and unsupervised approach (AM1, AM2,

[5] DTD: drum transcription of drum-only recordings

[6] DTP: drum transcription in the presence of percussion

[7] Numbers are omitted in the paper but are available online: https://www.audiolabs-erlangen.de/resources/MIR/2017-DrumTranscription-Survey.

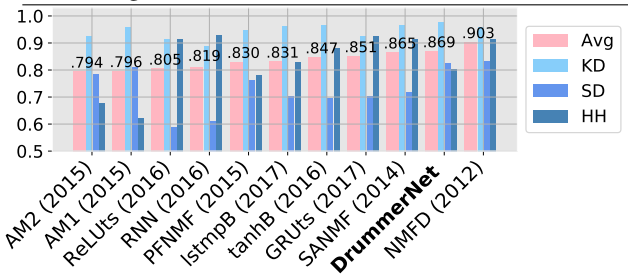


Figure 5: The F1 scores of DrummerNet and other systems on SMT with 'eval-cross' setup, sorted by the ascending order of the overall average. The system names follow [40].

PFNMF, and SANMF [10, 41]). It did not outperform NMFD [20], a system based on the convolutive NMF.

The comparison between DrummerNet and the NMF/unsupervised learning-based systems [10, 41] implies that the proposed deep neural network structure effectively learns relevant representations. Furthermore, DrummerNet allows constant-time inference, unlike NMF and other factorization-based approaches which require iterative optimization in the test time.

What is more interesting is its *generalizability*. All the deep learning based systems [8] present deteriorating performance in the transfer learning scenario (eval-cross) compared to the dataset split scenario (eval-subset). [9] However, less data-driven approaches [10] present similar or even increased performances in eval-cross. This implies that the distributions within datasets are fairly different and biased to certain types of drum tracks and therefore, a transcription system trained with those datasets will be also biased accordingly. This limitation may be attributed to the small sizes of those datasets. Theoretically, supervised deep learning systems may generalize better if trained on a very large dataset, which lacks practicality due to the high annotation cost. In contrast, it is relatively easy to *unbias* DrummerNet. One only needs to control the distribution of drum tracks by their style/genre/sounds without annotating every note.

4.4 Qualitative Analysis

In this section, we will analyze the performance and the behavior of DrummerNet by components, datasets, and metrics, as illustrated in Figure 6. Here, we notice two clear trends. First, across all of the three datasets and the metrics, detecting KD was the easiest, followed by SD and HH (except the precision on SMT). Second, SMT seems to be the easiest, followed by MDB and ENST. *What could be the reasons?*

The first trend is strongly related the proposed loss function. KD has the least within-class variability while being the most distinguishable component (the largest mutual-class variability) due to its solitary frequency range. SD and HH share both the mid and high-frequency ranges

[8] RNN, tanhB, ReLUts, lstmpB, GRUts - RNN-based systems

[9] See Figure 10 (b) of [40]. Note that most of the reported scores in papers also follow eval-subset setup.

[10] SANMF, NMF, PFNMF, AM1, AM2 - NMF-based systems

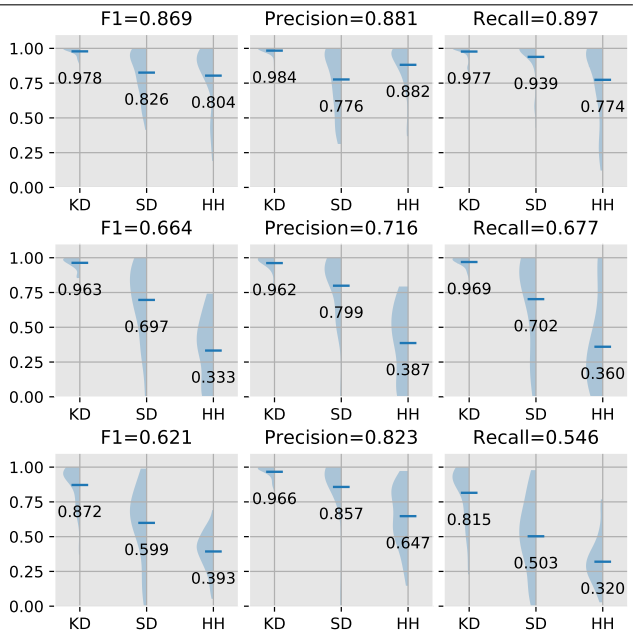


Figure 6: Evaluation of DrummerNet on SMT (top), MDB (middle), and ENST (bottom) datasets.

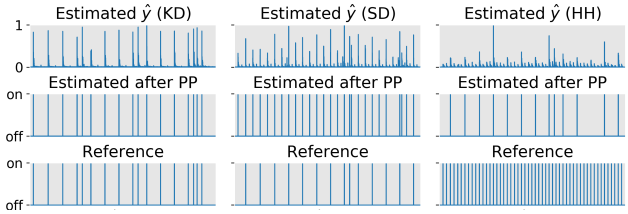


Figure 7: A transcription example of DrummerNet, 'Real Drum 01-12' in SMT - the output of analysis module (top), after peak-picking (middle), and ground truth (bottom); KD, SD, HH (left to right).

and their sounds can vary significantly across drum kits – i.e., larger within-class variability and smaller mutual-class variability. A common pattern, consequently, is the false positive of HH due to SD and vice versa. This is presented in Figure 7, where SD has many false positives due to HH.

The second trend is caused by the mixed use of the probability and the onset velocity in the DrummerNet. Although transcription $\hat{y}$ *is* the estimated amplitude of drum components, the peak-picking method treats $\hat{y}$ as if it was a probability. This discrepancy becomes problematic when the velocities of drum events in a track vary drastically as in the case of MDB and ENST. A failure case is demonstrated in Figure 7, where the HH with strong accents on several occasion caused DrummerNet to miss many of the other HH peaks.

4.5 Ablation Study

We conducted an ablation study where the performance of DrummerNet is compared with that of its variants. Figure 8 shows the reported F1 scores averaged over datasets and components. Please refer to the caption in Figure 8 for the definitions of the system names.

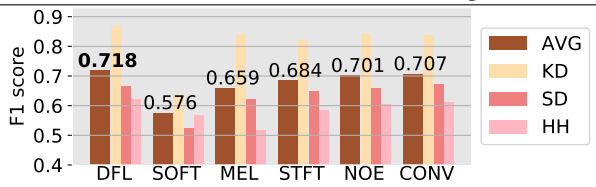


Figure 8: The ablation study results, F1 scores averaged over three datasets per component (KD, SD, HH) and their overall average (AVG). The label indicates as follow: DFL (default DrummerNet as introduced), SOFT (two Softmax layers instead of Sparsemax), MEL (use 128-band melspectrogram instead of CQTs), STFT (use 1024-point STFT instead of CQTs), NOE (not onset enhancement in loss), CONV (3-layer convolutional layers instead of recurrent layers).

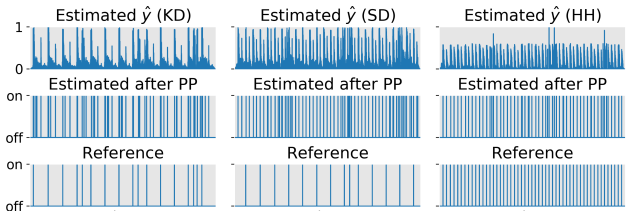


Figure 9: A transcription example of SOFT (DrummerNet with Softmax) , 'Real Drum 01-12' in SMT - the output of analysis module (top), after peak-picking (middle), and ground truth (bottom); KD, SD, HH (left to right).

Sparsemax (DFL vs. SOFT) Among all the variants in this experiment, we observe the most dramatic change in the performance when we replaced Sparsemax with Softmax (SOFT), mostly in a negative way. In SOFT, the two Softmax layers were applied in sequence instead of in-parallel and multiplied, which we tested, but the training was unstable. The transcription $\hat{y}$ of SOFT tends to be much noisier with many false positives, as presented in Figure 9. We conclude that the sparsity induced by Sparsemax is a crucial factor behind the success of the proposed unsupervised transcription.

Figure 9 provides a good example of the performance degradation pattern for each component. As in Figure 8, although the scores of all the three components decrease in SOFT, the degradation is not as critical for HH as in the case of KD/SD. This observation reflects the underlying properties of the different components. KD and SD are sparser than HH, and thus may benefit more from the introduction of Sparsemax.

CQT (DFL vs. MEL vs. STFT) Replacing CQTs with either melspectrograms (MEL) or short-time Fourier transform magnitudes (STFT) results in decreased performance. Unlike CQTs, where different numbers of FFT are used for each octave range, melspectrograms are computed based on single-resolution STFT. This implies that DrummerNet benefits from CQTs which consider multiple temporal and spectral resolutions.

Comparing MEL and STFT, the melfrequency compression helps with the better detection of KD but not SD nor HH. This is explained by the different frequency band weighting of STFT and melspectrogram. Since melfrequency is linear below 1 kHz and logarithmic above 1 kHz [32], melspectrogram allocates relatively more bins below 1 kHz. This means that the loss function in MEL is biased towards the low-frequency range, resulting in training that favors KD over the others.

Onset Enhancement (DFL vs. NOE) The onset enhancement is shown to be boosting the performance, but not significantly (0.017). In the learning curve, we observe that removing the onset enhancement from the loss function results in a large performance degradation during the initial phase of training. This is mainly due to false-positives in the non-transient part.

Recurrent layers (DFL vs. CONV) Overall, replacing three recurrent layers with three convolutional layers does not make significant differences (0.011). This may means i) a long-term relationship may not provide additional information, probably because the transcription largely depends on local information, and ii) the mutual conditioning in the last recurrent layer is not effective in our experiment. In an informal analysis, we observed that with recurrent layers, $\hat{y}$ still has some local temporal correlation, e.g., the activations are smeared over time, probably because that is better to reconstruct the input audio.

5. CONCLUSION

We introduced DrummerNet, a deep neural network that is trained to transcribe drum tracks without a labeled dataset. In the experiment, DrummerNet achieved strong performance compared to existing systems trained with supervised learning, showing its generalizability towards a real-world drum transcription scenario. Our ablation study showed that Sparsemax and CQT played a crucial role in the successful training of DrummerNet.

The experiment also revealed room for further improvements. Considering the discreteness of the musical notes, a reinforcement learning approach may be more suitable [35], making the prediction more sparse and replacing the peak-picking with trainable action. The onset-enhancement on audio similarity is a function carefully-chosen in order to approximate L_y when x and $\hat{x}$ are given. Unfortunately, the approximation is limited because the exact drum sounds in x are not given, and therefore a perfect reconstruct of (onsets of) the input audio ($L_x(x, \hat{x}) = 0$) does not lead to a perfect transcription ($L_y(y, \hat{y}) = 0$). An alternative way would be measuring a similarity on a (perceptual) representation domain instead of the audio, for example, by learning a loss using forward-backward consistency (also known as a cyclic loss [17]) or known audio features. Lastly, the current synthesizer module is limited to drums as it does not handle the duration of notes. A trainable synthesizer can be used to expand DrummerNet to other instruments [3, 11], eventually leading to an unsupervised universal transcription system combined with instrument recognition.

6. ACKNOWLEDGEMENT

We thank Tristan Jehan and Sebastian Ewert for their valuable comments and discussions. We would also like to express our sincere gratitude to Chih-Wei Wu for sharing his insight with us.

7. REFERENCES

[1] Samer A Abdallah and Mark D Plumbley. Unsupervised analysis of polyphonic music by sparse coding. *IEEE Transactions on neural Networks*, 17(1):179–196, 2006.

[2] Taylor Berg-Kirkpatrick, Jacob Andreas, and Dan Klein. Unsupervised transcription of piano music. In *Advances in neural information processing systems*, pages 1538–1546, 2014.

[3] Merlijn Blaauw and Jordi Bonada. A neural parametric singing synthesizer. *arXiv preprint arXiv:1704.03809*, 2017.

[4] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, 10 2016.

[5] Sebastian Böck, Florian Krebs, and Markus Schedl. Evaluating the online capabilities of onset detection methods. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 49–54, 2012.

[6] Sebastian Böck and Markus Schedl. Polyphonic piano note transcription with recurrent neural networks. In *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 121–124. IEEE, 2012.

[7] Mark Cartwright and Juan Pablo Bello. Increasing drum transcription vocabulary using data synthesis. *Proc. of the 21st Int. Conference on Digital Audio Effects (DAFx-18). Aveiro, Portugal*, 2018.

[8] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NeurIPS - Workshop on Deep Learning, December 2014*, 2014.

[9] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[10] Christian Dittmar and Daniel Gärtner. Real-time transcription and separation of drum recordings based on nmf decomposition. In *Proceedings of the conference on Digital Audio Effects (DAFx)*, pages 187–194, 2014.

[11] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. Neural audio synthesis of musical notes with wavenet autoencoders. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1068–1077. JMLR. org, 2017.

[12] Derry Fitzgerald. Harmonic percussive separation using median filtering. *Proceedings of the conference on Digital Audio Effects (DAFx), Graz, Austria, 2010*, 2010.

[13] Nicolai Gajhede, Oliver Beck, and Hendrik Purwins. Convolutional neural networks with batch normalization for classifying hi-hat, snare, and bass percussion sound samples. In *Proceedings of the Audio Mostly 2016*, pages 111–115. ACM, 2016.

[14] Olivier Gillet and Gaël Richard. Enst-drums: an extensive audio-visual database for drum signals processing. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 156–159, 2006.

[15] Fabien Gouyon, François Pachet, Olivier Delerue, et al. On the use of zero-crossing rate for an application of classification of percussive sounds. In *Proceedings of the conference on Digital Audio Effects (DAFx-00), Verona, Italy*, 2000.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[17] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Forward-backward error: Automatic detection of tracking failures. In *20th International Conference on Pattern Recognition*, pages 2756–2759. IEEE, 2010.

[18] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 371–386, 2018.

[19] Clément Laroche, Hélène Papadopoulos, Matthieu Kowalski, and Gaël Richard. Drum extraction in single channel audio signals using multi-layer non negative matrix factor deconvolution. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 46–50. IEEE, 2017.

[20] Henry Lindsay-Smith, Skot McDonald, and Mark Sandler. Drumkit transcription via convolutive nmf. In *International Conference on Digital Audio Effects (DAFx), York, UK*, 2012.

[21] Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623, 2016.

[22] Brian McFee, Matt McVicar, Stefan Balke, Vincent Lostanlen, Carl Thom, Colin Raffel, Dana Lee, Kyungyun Lee, Oriol Nieto, Frank Zalkow, Dan Ellis, Eric Battenberg, Ryuichi Yamamoto, Josh Moore, Ziyao Wei, Rachel Bittner, Keunwoo Choi, nullmightybofo, Pius Friesch, Fabian-Robert Stter, Thassilo, Matt Vollrath, Siddhartha Kumar Golu, nehz, Simon Waloschek, Seth, Rimvydas Naktinis, Douglas Repetto, Curtis "Fjord" Hawthorne, and CJ Carr. librosa/librosa: 0.6.3, February 2019.

[23] Brian CJ Moore. *An introduction to the psychology of hearing*. Brill, 2012.

[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Advances in neural information processing systems 2017 Workshop*, 2017.

[25] Jouni Paulus and Anssi Klapuri. Drum sound detection in polyphonic music with hidden markov models. *EURASIP Journal on Audio, Speech, and Music Processing*, 2009:14, 2009.

[26] Jouni Paulus and Tuomas Virtanen. Drum transcription with non-negative spectrogram factorisation. In *Signal Processing Conference, 2005 13th European*, pages 1–4. IEEE, 2005.

[27] Graham E Poliner and Daniel PW Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, page 048317, 2006.

[28] Colin Raffel, Brian Mcfee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel P. W. Ellis, C Colin Raffel, and Eric J. Humphrey. mir_eval: a transparent implementation of common mir metrics. In *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*, 2014.

[29] Axel Roebel, Jordi Pons, Marco Liuni, and Mathieu Lagrangey. On automatic drum transcription using non-negative matrix deconvolution and itakura saito divergence. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 414–418. IEEE, 2015.

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[31] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):927–939, 2016.

[32] Malcolm Slaney. Auditory toolbox. *Interval Research Corporation, Tech. Rep*, 10(1998), 1998.

[33] Carl Southall, Ryan Stables, and Jason Hockman. Automatic drum transcription using bi-directional recurrent neural networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 591–597, 2016.

[34] Carl Southall, Ryan Stables, and Jason Hockman. Automatic drum transcription for polyphonic recordings using soft attention mechanisms and convolutional neural networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 606–612, 2017.

[35] Carl Southall, Ryan Stables, and Jason Hockman. Player vs transcriber: A game approach to data manipulation for automatic drum transcription. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR), Paris, France*, 2018.

[36] Carl Southall, Chih-Wei Wu, Alexander Lerch, and Jason Hockman. MDB drums–an annotated subset of medleydb for automatic drum transcription. In *International Society for Music Information Retrieval Conference (ISMIR) Late-breaking/demo session*, 2017.

[37] Richard Vogl, Matthias Dorfer, and Peter Knees. Recurrent neural networks for drum transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 730–736, 2016.

[38] Richard Vogl, Matthias Dorfer, and Peter Knees. Drum transcription from polyphonic music with recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 201–205. IEEE, 2017.

[39] Richard Vogl, Gerhard Widmer, and Peter Knees. Towards multi-instrument drum transcription. *Proc. of the 21st Int. Conference on Digital Audio Effects (DAFx-18). Aveiro, Portugal*, 2018.

[40] Chih-Wei Wu, Christian Dittmar, Carl Southall, Richard Vogl, Gerhard Widmer, Jason Hockman, Meinard Muller, and Alexander Lerch. A review of automatic drum transcription. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 26(9):1457–1483, 2018.

[41] Chih-Wei Wu and Alexander Lerch. Drum transcription using partially fixed non-negative matrix factorization with template adaptation. *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 257–263, 2015.

[42] Chih-Wei Wu and Alexander Lerch. Automatic drum transcription using the student-teacher learning paradigm with unlabeled music data. In *Proc. Int. Soc. Music Inf. Retrieval Conf.*, pages 613–620, 2017.

[43] Chih-Wei Wu and Alexander Lerch. From labeled to unlabeled data–on the data challenge in automatic drum transcription. *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[44] Kazuyoshi Yoshii and Masataka Goto. Unsupervised music understanding based on nonparametric bayesian models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5353–5356, 2012.

ESTIMATING UNOBSERVED AUDIO FEATURES FOR TARGET-BASED ORCHESTRATION

Jon Gillick[1] **Carmine-Emanuele Cella**[2] **David Bamman**[1]

[1]School of Information, University of California, Berkeley
[2]CNMAT, University of California, Berkeley

jongillick@berkeley.edu, carmine.cella@berkeley.edu, dbamman@berkeley.edu

ABSTRACT

Target-based assisted orchestration can be thought of as the process of searching for optimal combinations of sounds to match a target sound, given a database of samples, a similarity metric, and a set of constraints. A typical solution to this problem is a proposed orchestral score where candidates are ranked by similarity in some feature space between the target sound and the mixture of audio samples in the database corresponding to the notes in the score; in the orchestral setting, valid scores may contain dozens of instruments sounding simultaneously.

Generally, target-based assisted orchestration systems consist of a combinatorial optimization algorithm and a constraint solver that are jointly optimized to find valid solutions. A key step in the optimization involves generating a large number of combinations of sounds from the database and then comparing the features of each mixture of sounds with the target sound. Because of the high computational cost required to synthesize a new audio file and then compute features for every combination of sounds, in practice, existing systems instead estimate the features of each new mixture using precomputed features of the individual source files making up the combination. Currently, state-of-the-art systems use a simple linear combination to make these predictions, even if the features in use are not themselves linear.

In this work, we explore neural models for estimating the features of a mixture of sounds from the features of the component sounds, finding that standard features can be estimated with accuracy significantly better than that of the methods currently used in assisted orchestration systems. We present quantitative comparisons and discuss the implications of our findings for target-based orchestration problems.

1. INTRODUCTION

In many music information retrieval and signal processing contexts, we are required to reason about signals that are themselves the sum of multiple sources. Whether the summing comes from instruments in a multi-track recording, voices in a group conversation, or simply from noise in the signal, we generally need to consider the full set of sources that make up an audio signal.

Much work in MIR deals with pulling apart the sources in a signal, either in the most straightforward sense via source separation [3,4], or through any of a number of classification tasks such as tagging [11, 31], instrument recognition [14, 18], or automatic transcription [16, 26]. A separate body of work deals with the inverse problem, that of putting sources together: work in applications like assisted orchestration [8] and automatic mixing [13, 25] aims to guide people through the task of combining signals together with the help of a machine in the loop.

In the cases of both separation and combination, tasks can be solved presumably because the source components and the summed signal are sufficiently correlated; the more correlated a source is with the mixture, the easier it is to identify, and as more signals are summed together, correlations between the combination and any single source tend to diminish. In a computational setting, these correlations are of course measured through a set of features of the signals, whether they be hand-engineered features like FFT and MFCC, or modern features learned by neural networks.

There are some cases, however, in which we can observe the source signals of a mixture, but it is impractical or impossible to actually compute the features of the signal in question; these are the cases that we investigate in this work. Broadly speaking, there are two primary settings in which we may be unable to observe the features of audio signals:

1. We do not have access to the signals.
2. Computing the features for *all* relevant signals is computationally expensive.

The first setting is commonly encountered in MIR, in which, as with many fields centered around media that may be under copyright or other protections, it is quite common

for researchers to have access to pre-computed features but not to raw data itself. For example, the audio files of the million songs that comprise the Million Song Dataset [5], which serves as a benchmark for many common MIR tasks, cannot be legally distributed. Instead the data contains common audio features like MFCC, chroma, note onsets, and spectral centroids. [1] Though this dataset and others like it are attractive because of their size and scope, they have been of limited use as source material for constructing additional audio mixtures. As semi-supervised and self-supervised approaches to machine learning have become more competitive with fully supervised systems, large datasets even of weakly labeled source material are becoming more useful for research in areas like source separation [15, 24]; estimating the features of mixtures might be one path towards making use of this data in new contexts.

The second setting in which we cannot observe audio features, which is the focus of this work, is the case where the computational cost of calculating an exponential number of audio mixtures is prohibitively high. This computational bottleneck often arises in the aforementioned body of work that attempts to automatically combine signals together during the course of tasks like target-based orchestration. In this context, learning algorithms need to explore a combinatorial space of potential solution sets, making it infeasible to compute the real features of all possible mixtures of signals. Moreover, methods for narrowing down this set of possible solutions, such as reinforcement learning algorithms, are generally iterative, requiring online evaluation of a reward function before the next set of candidates can be explored. Because these methods both have a large solution space and need to be evaluated iteratively, features must be computed *on the fly*, making fast feature computation, or accurate estimation, a necessity.

In this work, we take steps to explore the potential of machine learning models for predicting audio features of a mixture of sounds that we are unable to observe, focusing on the task of target-based assisted orchestration [2, 6, 8]. Concretely, we consider models of the following form: given a feature function f and M individual signals $S_1, \ldots, S_M$, we learn mappings from input features $f(S_1), \ldots, f(S_M)$ to the true feature of the mixture $f(S_1 + \ldots + S_M)$.

In experiments, we examine one standard feature that is known to typically behave linearly when summed (FFT magnitude spectra) and one feature that is less well suited to linear approximation (MFFC coefficients), investigate the ability of different models to predict each feature across a varying number of mixtures ranging from 2 notes to 30, and discuss the implications of our findings for target-based assisted orchestration as well as for the broader range of scenarios in which real audio features cannot be observed.

Code to reproduce our results can be found at `https://github.com/jrgillick/audio-feature-forecasting`.

[1] The full list of fields can be accessed here: `https://labrosa.ee.columbia.edu/millionsong/pages/field-list`

2. TARGET-BASED ASSISTED ORCHESTRATION

Musical orchestration, and in many cases, music production, consists largely of choosing combinations of sounds, instruments, and timbres that support the narrative of a piece of music. Strong orchestration can bring a composition to life by emphasizing, clarifying, or perhaps questioning the elements of the music, and through this process, orchestration can often be a difference-maker to critical or commercial success [20, 21].

Different musical styles and composition environments have different constraints (for example, scores for live performance should only require the sounds of the instruments in the group, whereas the sounds available for use on a recording are only limited by their stylistic relevance), but fundamentally, finding the right set of sounds is important regardless of the context. For composers and producers, employing MIR systems during the orchestration process holds the potential to help spark inspiration, solve challenging problems, or save time.

Though the orchestral setting has been explored extensively in previous work, assisted orchestration methods hold the potential for application in other styles. For example, *layering* drum samples is common practice in music production, and MIR-based tools for drum sample search are beginning to make their way into professional toolkits [2]; existing methods for query-based drum sample retrieval [23] could be extended to consider mixtures of drum samples.

3. RELATED WORK

Existing systems for target-based assisted orchestration compute spectral similarities using standard spectral features [8] or perceptual descriptors [2], along with evolution-based methods for exploring the solution space.

Most relevant to our experimental setting is the implementation of the publicly available state-of-the art Orchidea system [10], which is based in part on a study conducted in [9] on predicting timbral features of combined sounds. This study found that for 3 features (Spectral Centroid, Spectral Spread, and Main Resolved Partials), and for mixtures of up to 4 sounds, predicting the features of the mixture by a linear combination of the source features both achieved a low error and did not vary as a function of the number of mixture components.

Since computing a linear combination has very low computational cost, this finding enables real-time estimation of thousands of candidate mixtures for use in online reinforcement learning, making tools like Orchidea practical for real-world use. The effects on the features of mixing many more components, along with the behavior of higher-dimensional and richer features, however, have not yet been investigated.

[2] https://www.xlnaudio.com/products/xo

4. EXPERIMENTS

4.1 Data

For our experiments, we use the OrchDB dataset of individually recorded musical notes from a variety of orchestral instruments. OrchDB is a streamlined subset of the Studio Online (SOL), dataset that has been optimized for assisted orchestration [30]. Collected as part of the Studio Online project at IRCAM, the full SOL data set contains over 117,000 instrument samples, including extended techniques and contemporary playing styles. OrchDB, which contains a curated subset of these samples, has been used for assisted orchestration since 2008 [7]; the contents of the data are summarized below:

- OrchDB contains about 20,000 notes with lengths ranging from about 1 second to 30 seconds.
- Instruments include bassoon, clarinet, flute, horn, oboe, saxophones, strings, trombone, trumpet, and tuba.
- Approximately 30 different playing styles are represented in OrchDB, such as ordinario, pizzicato, pizzicato Bartók, aeolian, Flatterzung, col legno battuto; brass instrument samples include a variety of different types of mutes.
- Notes across the pitch range are included, along with a range of dynamics from *ppp* to *fff*, including sforzato and crescendo.

4.2 Mixtures of Notes

To train and evaluate models for feature estimation, we partition the dataset into nonoverlapping subsets for training, development, and testing, choose 6 different numbers of mixture components M between 2 and 30, and then for each M, we synthesize a dataset of new audio files by adding together the raw waveforms of M randomly chosen notes. Finally, we divide the summed signals by M to keep the amplitudes of the mixture in the same range as those of the source files.

For each value of M, we synthesize 7500 new audio mixtures for training, 2000 for development, and 2000 for testing, creating these new mixtures after partitioning our data so that no source file that appears in the training set can be chosen as part of a mixture in the test set. After synthesizing the mixtures, we compute and store the real FFT and MFCC features for every mixture for use in training and evaluating our models.

4.3 Predicting Unobserved Features

With this data in hand, we explore several models for predicting the features of a mixture of audio signals given the features of the individual signals. In all experiments, given a feature function f and M individual signals $S_1, \ldots, S_M$, each model is trained to learn a mapping from input features $f(S_1), \ldots, f(S_M)$ to the true feature of the mixture $f(S_1 + \ldots + S_M)$.

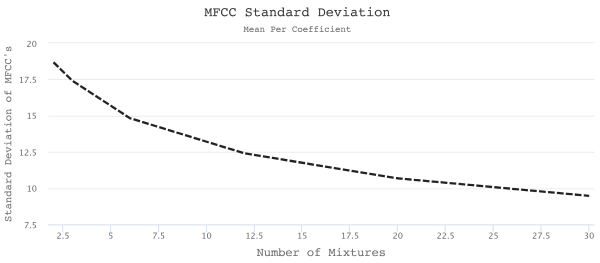


Figure 1. Standard Deviations (averaged across all 19 coefficients) of the real MFCC coefficients of mixtures of audio files. As M increases, the variance in the MFCC coefficients goes down.

5. MODELS

5.1 Features

For our modeling experiments, we choose two standard features: 1024-dimensional FFT magnitude spectra and 19-dimensional MFCC coefficients (we discard the first of 20 MFCC coefficients). Our choice of features is meant to capture the most common MIR settings, so we use the default FFT and MFCC dimensions specified in the Librosa library [22] and compute the features on audio files sampled at 22050 Hz using the default window size (2048 samples) and hop size (512 samples) of the Librosa implementations. We then follow [8] in flattening both the FFT and MFCC features from 2-dimensional time-frequency representations into 1-dimensional feature vectors by taking a linear combination of the features at each frame, weighted by the RMS energy at the corresponding frame.

This method of averaging over time allows us to summarize the spectral characteristics of signals with different lengths using a single feature feature vector, while at the same time ignoring the quieter parts of the signal. In addition, we preserve the interpretability of the FFT and MFCC features through this process, which is particularly useful for inspecting and analyzing our model outputs. Of course, the downside of this preprocessing step is that we discard all time-domain information, so we are unable to predict anything about the envelope or movement of the sound. Depending on the source material and the downstream application, different preprocessing choices might be more appropriate than averaging over time; for example, unpitched percussive sounds require different modeling choices from pitched material. Since our data consists of mostly pitched notes from orchestral instruments, however, we follow the convention of the assisted orchestration literature by focusing on timbre independent of time.

Finally, before training or evaluating models, in order to best align our quantitative results to the expected perceptual results with regards to timbre, we normalize the FFT feature vectors such that the maximum value is 1. Although in the FFT case, relative magnitudes are known to be more correlated with perception of timbre than the raw amplitudes are, magnitudes of MFCC coefficients are important descriptors of timbre, so we do not normalize the

MFCC's, instead predicting the real values.

5.2 Baseline

As a baseline, we compute the element-wise mean of the feature vectors over the entire training set. This vector is computed once for each value of the number of mixture components M. Models that perform better than this baseline can be said to be capturing some useful information about how the features sum together. One important factor to take into account when evaluating results it that as we increase M, mixing more and more notes together, the variance in the features of the mixtures decreases, making the predictive task appear easier. The MFCC features, because they are much lower dimensional than the FFT's, are especially effected by this change in variance; the higher dimensional FFT features exhibit the same trend but to a smaller extent, as they can capture a wider range of combinations of signals. For this reason, in Section 6, we report error metrics as a percentage relative to the error metric of this baseline at the corresponding value of M. Concretely, an error of 0.5 would mean that, averaged over the test set, the sum of squared errors of our predictions was equal to half of the sum of squared errors obtained by always predicting the mean of the data set.

5.3 Linear Combination

The first model we examine is the linear combination of features proposed in [9], which is currently used in state-of-the-art assisted orchestration systems [10]. This model implicitly assumes that for a feature f, the feature of the sum is approximately equal to the sum of the features:

$$w_1 f(S_1) + \ldots + w_M f(S_M) \approx f(w_1 S_1 + \ldots w_M S_M) \quad (1)$$

When features are linear or can be well approximated linearly, this method can be a strong baseline. Especially with high dimensional features like our 1024 FFT magnitudes, subtle details that might be difficult to summarize in an intermediate representation can be easily preserved with a linear model.

For this model, we combine source features in two ways, first by taking the element-wise mean of the M feature vectors as shown in Equation (2) and second by weighting the features by the corresponding RMS energies $a_1 \ldots a_M$ of the component signals as in Equation (3):

$$\bar{f} = \frac{1}{M} \sum_{f_i} \quad (2)$$

$$\tilde{f} = \frac{\sum f_i a_i}{\sum a_i} \quad (3)$$

5.4 MLP

Particularly for nonlinear features, it is reasonable to expect that nonlinear models have the potential to make better estimates. We train multilayer perceptron (MLP) neural networks to predict both FFT and MFCC features. For these models, we use a single hidden layer, and we minimize the mean squared error (MSE) between the predictions and the targets. For the FFT models, in order to constrain the network to output magnitudes between 0 and 1, we use a ReLU activation followed by a L_∞ normalization layer as the last stage in our network. Although we found empirically that sigmoid activations gave similar accuracies, these choices match better with the intuition of normalization performed in preprocessing. We train all our neural network models with Tensorflow [1] and Keras [12], Dropout [27], and the Adam optimizer [19].

Because we are interested in testing our methods on a variable number of audio mixtures M, we train separate MLP models for each value of M. As M increases, the input size and number of parameters in the network increases accordingly; with a feature of dimension D and a hidden layer of size H, the first layer of these networks has $D \times M \times H$ trainable parameters.

5.5 LSTM

As we increase the number of mixtures M, a recurrent network architecture is a natural choice to reduce the number of parameters needed. Intuitively, if a network can learn to estimate the sum of two signals, the same network should be able to process M signals in sequence over M steps by estimating the sum of one signal with the sum of all the signals processed so far.

5.5.1 Ordered Sets

Because the true sum of M signals is independent of the order in which they are combined, we experiment with two approaches inspired by the literature on sequence models for sets. First, even when no true ordering exists, previous results demonstrate that the ordering of inputs to factorized probabilistic models still affects the ability of models to learn [29]. In the case where two semantically valid orderings exist, empirical results from machine translation show that simple changes to the ordering, such as reversing the words in a sentence, can significantly affect model performance [28]. Based on these results, for this variant of the model, we sort the signals by their L_2 norm before passing them to LSTM model, such that the source signal with the highest energy is observed at the final timestep before outputting a final prediction.

5.5.2 Unordered Sets

Although previous work points to the benefits of ordering the signals in a consistent way, fixing an ordering prevents us from implementing a simple but potentially powerful form of data augmentation - randomly shuffling the order of mixtures during training. We empirically test the relative benefits of these two options, reporting results for both ordered and unordered inputs with the same LSTM archicture.

5.5.3 Residual Connections

Finally, we experiment with one more variation of our LSTM model, in which we add a residual connection [17]

between the model inputs at each timestep and the outputs of the LSTM layer, which allows information to pass directly from the input to the final layer without having to be mediated by the nonlinear structure of the LSTM. Intuitively, to the degree to which features are linear, this connection should provide the model with the option to directly sum up the features as part of its computation.

6. RESULTS

We train and evaluate all of the models across 6 different numbers of mixtures M ranging from 2 to 30, summarizing the results in Tables 1 and 2 and displaying the trends across values of M in Figures 2 and 3.

6.1 Predicting FFT Features

As demonstrated in Figure 2, the linear combination outperform the neural methods for values of M between 2 and 12, but the LSTM models make up ground and ultimately begin to overtake the linear combinations at $M = 20$ mixtures. All of the models in the FFT setting trend up in error towards the baseline as the number of mixtures increases; with $M = 30$, all models except for the residual LSTM cross the threshold of the baseline. These results indicate several findings:

- While the ordering in which the mixtures are passed to the LSTM model does not appear to make a significant difference here, the residual LSTM model outperforms the rest of the neural methods at all values of M, demonstrating increasingly large gains as the number of mixtures goes up. This suggests that the residual connection may be enabling the model to exploit the linearity of features when it is advantageous to do so, while maintaining flexibility to make better estimations once the signal from the linearity of the feature fades.
- In confirmation with previous findings [9], these results suggest that linear approximations of FFT features can be quite accurate. As the number of mixtures increases, however, these estimates worsen; by $M = 30$, the linear approximations are no better than random.
- Although estimating a high dimensional feature like the FFT is clearly a challenging task as many streams of audio are mixed together, these results show that neural models do possess the potential to estimate these features to some degree even in settings with many different sources.

6.2 Predicting MFCC Features

Unlike in the case of the FFT features, all of the neural models outperform the linear combination for both small and large numbers of mixtures, and as shown in Figure 3, with more than 6 mixtures, linear combinations of MFCC features no longer contain a useful signal. We detail our findings from the MFCC experiments below:

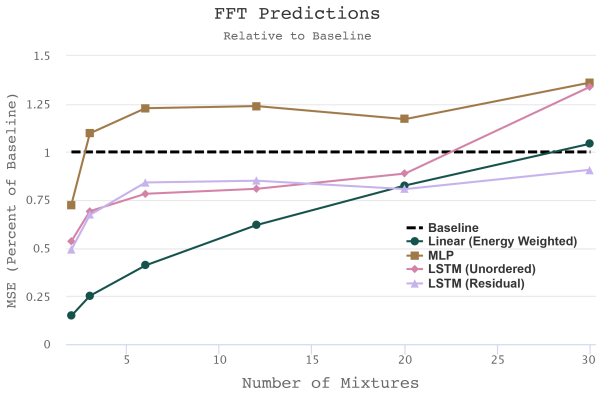


Figure 2. The linear models work well for predicting FFT features of small numbers of mixtures, but at around $M = 20$ mixtures, the best performing LSTM model overtakes the linear combination.

- Because MFCC features are nonlinear, it is not surprising that nonlinear models are able to predict them better than the linear combination. Relative to the baseline, however, we can see that for mixtures of 2, 3, and even 6 different sources, a linear combination of MFFC's can still be reasonably accurate. This suggests that in some cases, MFFC features do behave approximately linearly when summed.
- In contrast to the FFT setting, the residual LSTM does not appear to offer any gains in comparison with the other LSTM models. Perhaps because of the much smaller dimension of the features, the Unordered LSTM model, which we train with data augmentation by randomizing the order in which mixtures of processed, performs best.
- As M continues to increase, the accuracies of the LSTM models flatten out rather than continuing to approach the baseline. This trend suggests that even when dozens of notes are mixed together, we may be still able to estimate certain features of these mixtures based only on the features of the source files.

6.3 Computation Time

While the exact computation time of FFT or MFFC features depends on the implementation, the length of the audio files, and the availability of parallel processing, estimating features with the networks we explore is, in practice, significantly faster than computing the real features. Though it is beyond the scope of this paper to report results on a comprehensive list of hardware and software configurations, as a point of reference, Table 3 displays running times for parallel computation on our research server containing 20 CPU's and one Tesla K40 GPU.

7. CONCLUSIONS

In this work, we experiment with neural models for predicting unobserved audio features based on precomputed

Model	2 Mixtures	3 Mixtures	6 Mixtures	12 Mixtures	20 Mixtures	30 Mixtures
Baseline	1	1	1	1	1	1
Linear (Mean)	0.44	0.60	0.73	0.85	0.99	1.16
Linear (Energy-Weighted)	**0.15**	**0.25**	**0.41**	**0.62**	0.83	1.04
MLP	0.72	1.10	1.23	1.24	1.17	1.36
LSTM (Unordered)	0.54	0.69	0.78	0.81	0.89	1.34
LSTM (Ordered)	0.55	0.73	0.85	0.88	0.86	1.35
LSTM (Residual)	0.49	0.67	0.84	0.85	**0.81**	**0.91**

Table 1. Mean Squared Error for predicting FFT features across different numbers of mixtures.

Model	2 Mixtures	3 Mixtures	6 Mixtures	12 Mixtures	20 Mixtures	30 Mixtures
Baseline	1	1	1	1	1	1
Linear (Mean)	0.43	0.58	0.81	1.03	1.32	1.54
Linear (Energy-Weighted)	0.36	0.59	0.94	1.30	1.69	2.02
MLP	0.42	0.55	0.71	0.79	0.88	0.93
LSTM (Unordered)	**0.30**	**0.46**	**0.57**	**0.63**	**0.71**	**0.70**
LSTM (Ordered)	**0.30**	**0.46**	0.61	0.66	0.73	0.73
LSTM (Residual)	0.32	0.47	0.64	0.71	0.77	0.77

Table 2. Mean Squared Error for predicting MFCC features across different numbers of mixtures.

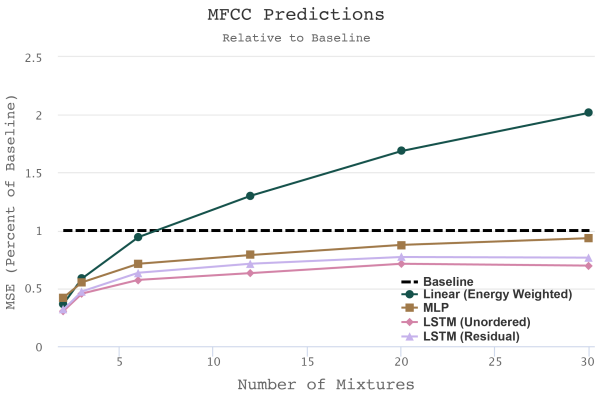


Figure 3. The neural models outperform the linear combinations significantly, widening the gap as M increases.

Feature	**Real** (CPU x 20)	**LSTM** (CPU x 20)	**LSTM** (GPU x 1)
FFT (Mix 2)	14.71	0.32	0.07
FFT (Mix 30)	14.71	4.75	1.10
MFCC (Mix 2)	73.50	0.03	0.01
MFCC (Mix 30)	73.50	0.34	0.15

Table 3. Time in seconds to compute or estimate energy-weighted FFT or MFCC features for the 2000 audio files in the test set using parallel processing. FFT (Mix 30) refers to the FFT feature of a mixture of 30 audio files, which requires 30 autoregressive LSTM steps. LSTM refers to the Residual LSTM model.

features of source files in a mixture, examining the cases of FFT features, which should behave linearly when summed, as well as MFCC's, which are known to be nonlinear. We find that in the case of nonlinear features, LSTM models significantly outperform the methods currently in use for feature estimation, and further, that while the linear predictors perform well for small numbers of mixtures, as we mix more and more signals together, the neural models begin to outperform the linear methods as well.

Our results suggest that we may be able to improve current assisted orchestration systems [10] by replacing feature estimation components with LSTM-based nonlinear predictors. As with any real-world problem that involves perceptual similarity rather than comparisons in a feature space, however, more work is needed to understand how these models may interact with other components of systems they may be embedded in. Deep neural network models can and do adapt to any correlations present in the data, so understanding how these models are making there estimates may be important.

Beyond tasks like assisted orchestration in which we cannot always observe the features of an audio file because of computational limitations, we hope that future work may be able to take advantage of the methods for feature estimation explored here in order to make creative use of data like the Million Song Dataset, for which precomputed features are available but raw data cannot be distributed.

8. ACKNOWLEDGMENTS

The research reported in this article was supported by the Hellman Family Faculty Fund and by resources provided by NVIDIA. We also thank the anonymous reviewers for their valuable feedback.

9. REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Aurelien Antoine and Eduardo Miranda. A perceptually orientated approach for automatic classification of timbre content of orchestral excerpts. *The Journal of the Acoustical Society of America*, 141(5):3723–3723, 2017.

[3] Shoko Araki, Francesco Nesta, Emmanuel Vincent, Zbyněk Koldovskỳ, Guido Nolte, Andreas Ziehe, and Alexis Benichoux. The 2011 signal separation evaluation campaign (sisec2011):-audio source separation. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 414–422. Springer, 2012.

[4] Adel Belouchrani, Karim Abed-Meraim, J-F Cardoso, and Eric Moulines. A blind source separation technique using second-order statistics. *IEEE Transactions on signal processing*, 45(2):434–444, 1997.

[5] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. *In Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 591––596, 2011.

[6] Marcelo Caetano, Asterios Zacharakis, Isabel Barbancho, and Lorenzo J Tardón. Leveraging diversity in computer-aided musical orchestration with an artificial immune system for multi-modal optimization. *Swarm and Evolutionary Computation*, 2019.

[7] Grégoire Carpentier. *Approche computationnelle de l'orchestration musciale-Optimisation multicritère sous contraintes de combinaisons instrumentales dans de grandes banques de sons*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 2008.

[8] Grégoire Carpentier, Gérard Assayag, and Emmanuel Saint-James. Solving the musical orchestration problem using multiobjective constrained optimization with a genetic local search approach. *Journal of Heuristics*, 16(5):681–714, 2010.

[9] Grégoire Carpentier, Damien Tardieu, Jonathan Harvey, Gerard Assayag, and Emmanuel Saint-James. Predicting timbre features of instrument sound combinations: application to automatic orchestration. *Journal of New Music Research*, 39(1):47–61, 2010.

[10] Carmine-Emanuele Cella and Philippe Esling. Open-source modular toolbox for computer-aided orchestration. 2018.

[11] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. *In Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2017.

[12] François Chollet et al. Keras, 2015.

[13] Brecht De Man and Joshua D Reiss. A semantic approach to autonomous mixing. *Journal on the Art of Record Production (JARP)*, 2013.

[14] Antti Eronen and Anssi Klapuri. Musical instrument recognition using cepstral coefficients and temporal features. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*, volume 2, pages II753–II756. IEEE, 2000.

[15] Zhe-Cheng Fan, Yen-Lin Lai, and Jyh-Shing R Jang. Svsgan: Singing voice separation via generative adversarial network. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 726–730. IEEE, 2018.

[16] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. *In Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Eric Humphrey, Simon Durand, and Brian McFee. Openmic-2018: an open dataset for multiple instrument recognition. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] Stephen McAdams. Perspectives on the contribution of timbre to musical structure. *Computer Music Journal*, 23(3):85–102, 1999.

[21] Stephen McAdams. Timbre as a structuring force in music. In *Proceedings of Meetings on Acoustics ICA2013*, volume 19, page 035050. ASA, 2013.

[22] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25, 2015.

[23] Adib Mehrabi, Keunwoo Choi, Simon Dixon, and Mark Sandler. Similarity measures for vocal-based drum sample retrieval using deep convolutional auto-encoders. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 356–360. IEEE, 2018.

[24] Andrew Owens and Alexei A Efros. Audio-visual scene analysis with self-supervised multisensory features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 631–648, 2018.

[25] Joshua Reiss and Øyvind Brandtsegg. Applications of cross-adaptive audio effects: automatic mixing, live performance and everything in between. *Frontiers in Digital Humanities*, 5:17, 2018.

[26] Carl Southall, Ryan Stables, and Jason Hockman. Automatic drum transcription using bi-directional recurrent neural networks. In *Proceedings of the 17th International Society for Music Information Retrieval Conference*, pages 591–597, 2016.

[27] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[28] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[29] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

[30] Rolf Wöhrmann and Guillaume Ballet. Design and architecture of distributed sound processing and database systems for web-based computer music applications. *Computer Music Journal*, 23(3):73–84, 1999.

[31] Yusuf Yaslan and Zehra Cataltepe. Audio music genre classification using different classifiers and feature selection methods. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 2, pages 573–576. IEEE, 2006.

TOWARDS AUTOMATICALLY CORRECTING TAPPED BEAT ANNOTATIONS FOR MUSIC RECORDINGS

Jonathan Driedger[1], **Hendrik Schreiber**[2], **W. Bas de Haas**[1], **and Meinard Müller**[2]
[1] Chordify, [2] International Audio Laboratories Erlangen
jonathan@chordify.net

ABSTRACT

A common method to create beat annotations for music recordings is to let a human annotator tap along with them. However, this method is problematic due to the limited human ability to temporally align taps with audio cues for beats accurately. In order to create accurate beat annotations, it is therefore typically necessary to manually correct the recorded taps in a subsequent step, which is a cumbersome task. In this work we aim to automate this correction step by "snapping" the taps to close-by audio cues—a strategy that is often used by beat tracking algorithms to refine their beat estimates. The main contributions of this paper can be summarized as follows. First, we formalize the automated correction procedure mathematically. Second, we introduce a novel visualization method that serves as a tool to analyze the results of the correction procedure for potential errors. Third, we present a new dataset consisting of beat annotations for 101 music recordings. Fourth, we use this dataset to perform a listening experiment as well as a quantitative study to show the effectiveness of our snapping procedure.

1. INTRODUCTION

Identifying the time positions of beats in music recordings has been a core task in the Music Information Retrieval (MIR) community for a long time. Irrespectively of whether the goal is to evaluate beat tracking algorithms or to train new data-driven models for beat detection, it is necessary to have accurate annotations that describe the temporal locations of beats in music recordings. The beat positions of a music recording are often loosely defined as the time instances where a human would tap along when listening to it [6]. A straightforward approach to create beat annotations is therefore to record these taps—for example by using a specialized audio player software like *Sonic Visualizer* [5], which allows annotators to tap on a key of the keyboard. This method was used, for example, in [20, 21, 25, 26]. However, this procedure is problematic due to the limited ability of humans to accurately align their taps with acoustic cues that are associated with beats such as instrument onsets, percussive sound events, or chord changes [29, 30]. Perception literature indicates that, depending on the complexity of a recording, the onset times of two tones must differ by less than 40 milliseconds such that they may be perceived as being temporally aligned [19]. This means that when sonifying human-made taps with a click track, a click and an audio cue for a beat have to both fall into an interval of at most 40 milliseconds such that the click could be perceived as accurately representing the beat position. [1] This is often not the case as is illustrated in Figure 1. In Figure 1a, we see a short spectrogram excerpt of the song "T'envoler" by Paul Daraiche (item 006 in our dataset, see Table 1 for the YouTube link). One can observe the vertical spectral structures originating from the piano onsets in the song's intro. The human-made taps are visualized in Figure 1b. They roughly coincide with the audio cues, but precede them by about 80 milliseconds most of the time. To obtain more accurate beat annotations, several approaches have been used in the past. One way is to manually correct the taps of a human annotator in a subsequent step [20,21]. However, manual corrections are cumbersome to perform since often every single tap has to be corrected individually—usually in a drag&drop fashion using tools like Sonic Visualizer. Another approach, which has been used in [15, 24], is to compute an initial

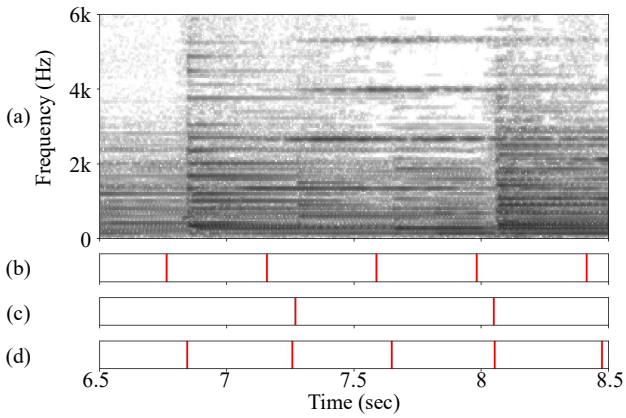


Figure 1. **(a)** Excerpt of the spectrogram for item 006 from our dataset, **(b)** taps by the annotator, **(c)** beat positions as estimated by [4], **(d)** automatically corrected taps.

[1] Note that the 40 milliseconds constitute an upper bound. Depending on the quality of the audio cue this threshold may be significantly lower.

estimate of the beat positions using a beat tracking algorithm. Beat trackers such as [3, 14] actively aim to "snap" potential beat candidates to close-by audio cues in order to create accurate results. In Figure 1c, we see the beat positions as estimated by *madmom*, a Python library featuring a state-of-the-art beat tracker [3, 4]. The estimated beats are well aligned with the audio cues visible in the spectrogram. However, we also see that only every other beat has been captured. Correcting these kinds of errors made by beat tracking algorithms can be just as cumbersome as manually correcting the taps of a human annotator.

In this paper, we propose a new way of creating beat annotations. Our idea is to mostly automate the manual correction of human-made taps by using the concept of snapping beat candidates to audio cues. The intuition is that the taps made by an annotator constitute good beat candidates that are located in close proximity to the actual beat positions. Therefore, snapping them to nearby audio cues should accurately correct the vast majority of taps. This is visualized in Figure 1d, where the automatically corrected taps are aligned with the audio cues in the spectrogram. In this paper, we explore this simple idea in a systematic fashion. First, in Section 2, we model the automatic correction procedure mathematically. Then, in Section 3, we propose a novel visualization that can serve as a tool to reveal rhythmically challenging sections in music recordings as well as potential errors made by the snapping procedure. Section 4 is dedicated to experiments. In Section 4.1, we apply our proposed annotation strategy to create a dataset of beat annotations for 101 music recordings from YouTube. In Section 4.2, we then discuss the results of a listening experiment that shows that human listeners perceive the corrected taps as more accurate than the original taps. In Sections 4.3, we finally conduct a small study that investigates the effect of using either the original or the corrected taps as ground truth for the quantitative evaluation of different beat tracking algorithms. For the purpose of reproducibility, we made our Python implementations (snapping procedure and visualizations) as well as the dataset of beat annotations along with YouTube identifiers publicly available at [13].

2. PROPOSED PROCEDURE

In this section, we formalize our procedure for the automatic correction of human-made taps. We start by explaining the core ideas in Section 2.1 and discuss choices of specific processing steps in Section 2.2.

2.1 Basic Principle

The goal of our proposed procedure is to correct the human-made taps by snapping them to nearby audio cues in the music recording. Our fundamental assumption is that each of the taps indicates the rough position of a beat such that the "real" beat position can be found in close temporal proximity. Given the music recording $x : \mathbb{Z} \to \mathbb{R}$ (Figure 2a), we first derive an *activation curve* $a : \mathbb{Z} \to \mathbb{R}$ (Figure 2b) that is sampled at a sampling rate of $f_s \in \mathbb{R}^+$

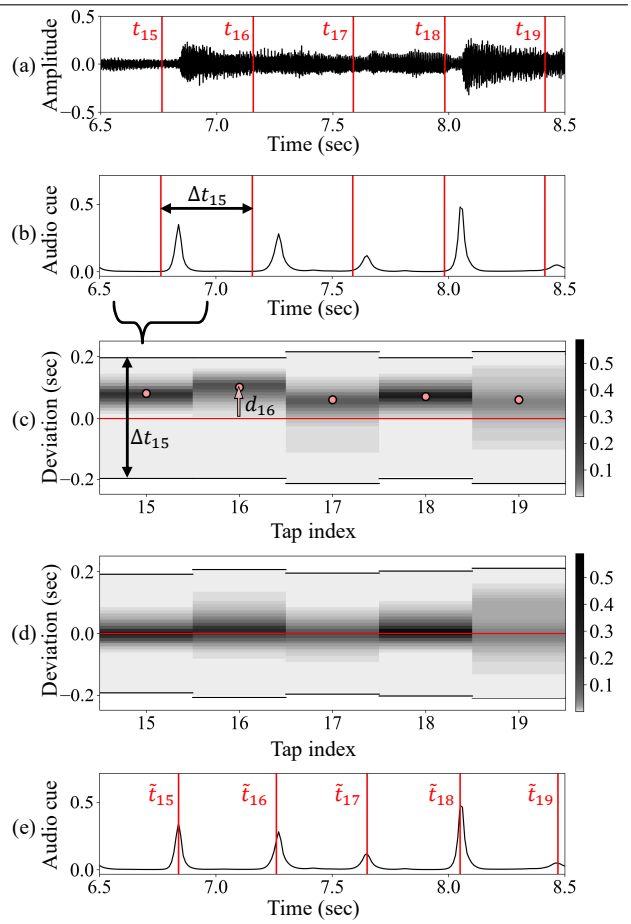


Figure 2. Proposed tap correction procedure. **(a)** Music recording x with taps t. **(b)** Activation curve a. **(c)** Deviation function $\mathcal{D}_t$ with envelope reflecting the inter-tap-intervals, tap positions indicated by red line, and deviation sequence d indicated by light-red dots. **(d)** Deviation function $\mathcal{D}_{\tilde{t}}$ with corrected taps $\tilde{t}$ indicated by red line. **(e)** Activation curve a with corrected taps $\tilde{t}$ indicated in red.

(we use f_s=100 Hertz as suggested in [4, 12, 17]). An activation curve a can be seen as a function whose values $a(n)$ reflect how likely it is that there is a beat present in the music recording x at time n/f_s. We discuss different choices for activation curves in Section 2.2. Along with x and a, we are also given the *sequence of taps* $t = [t_0, \ldots, t_{M-1}]$ with $t_m \in \mathbb{Z}$. Each tap t_m indicates that the human annotator tapped at time t_m/f_s. In our example in Figure 2b one can see that the taps are not well aligned with the peaks in the activation curve a.

With the activation curve and the taps, we now compute the *deviation function* $\mathcal{D}_t : \mathbb{Z} \times [0 : M-1] \to \mathbb{R}$ by

$$\mathcal{D}_t(n, m) := w_m(n)\ a(t_m + n)$$

for $n \in \mathbb{Z}, m \in [0 : M-1]$. Here, $w_m : \mathbb{Z} \to \mathbb{R}$ is a Hann window centered around zero, whose length is defined by the *inter-tap-interval*

$$\Delta t_m := t_{m+1} - t_m$$

for $m \in [0 : M-2]$ and we define $\Delta t_{M-1} := \Delta t_{M-2}$ (such that Δt_m is defined for all taps). The reason for using a

window function is to effectively implement our assumption that the taps are located in close temporal proximity to the actual beat positions. The more temporal distance between a tap and a high activation value, the less likely it is that the activation value reflects the actual beat the tap was meant to represent. In the visualization of $\mathcal{D}_t$ seen in Figure 2c, the length of w_m is indicated with two additional black lines in each column of $\mathcal{D}_t$. This "envelope" of $\mathcal{D}_t$ serves as a visual representation of the individual inter-tap-intervals. In our example, the envelope does not exhibit any significant variation across the shown taps which indicates that the annotator tapped with an almost constant tempo. However, when looking at the individual activation maxima in each column of $\mathcal{D}_t$—which can be found at deviations between 70 and 110 milliseconds—it becomes obvious that the inaccuracy of the human-made taps is not just a constant offset but varies over time.

In the next step we compute the *deviation sequence* $d = [d_0, \ldots, d_{M-1}]$, $d_m \in \mathbb{Z}$, that indicates the individual corrections we will apply to each tap t_m (Figure 2c). There are several options of how to derive d from $\mathcal{D}_t$ which we discuss in Section 2.2. From t and d we finally compute the *automatically corrected taps* $\tilde{t} = [\tilde{t}_0, \ldots, \tilde{t}_{M-1}]$ by

$$\tilde{t}_m := t_m + d_m \ .$$

Based on $\tilde{t}$ we now can also compute the deviation function $\mathcal{D}_{\tilde{t}}$ (Figure 2d). Note that in this visualization, all high activation values are now centered around a deviation of zero. Furthermore, the envelope of $\mathcal{D}_{\tilde{t}}$ does not differ significantly from the envelope of $\mathcal{D}_t$ which indicates that the inter-tap-intervals in $\tilde{t}$ are similar to those in the original tap sequence t. This can be verified when plotting $\tilde{t}$ on top of the activation curve a where the taps now accurately align with the peaks (Figure 2e).

2.2 Technical Realization

In traditional music signal processing, a common choice for activation curves are *novelty curves*, see for example [2, 7, 27]. These functions are designed to reflect sudden temporal changes in a music recording's spectrogram, which are typically caused by percussive sound events such as instrument onsets. As beats often go along with these kinds of sound events it is reasonable to use a novelty curve as activation curve in our tap correction procedure. We denote this activation curve by a^{nov} and the resulting deviation function for taps t by $\mathcal{D}_t^{\text{nov}}$.

Another option is using activation curves based on data-driven models. Recently, models that were trained to transform spectrogram representations of music recordings into activation curves have significantly improved the quality of state-of-the-art beat tracking algorithms [16]. For example in [4], a *deep neural network* (DNN) using a bidirectional long-short-term memory architecture is trained on a large collection of beat-annotated music recordings for that task. It was shown in [4] that the peaks in activation curves derived using this model align well with beats in the underlying music recordings. Using it in our procedure is therefore

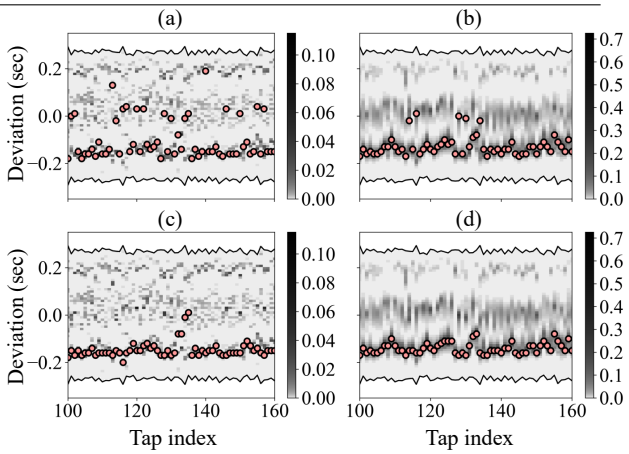


Figure 3. Excerpts of different deviation functions and deviation sequences for item `011`. **(a)** $\mathcal{D}_t^{\text{nov}}$ with $d^{\max}$ derived from it. **(b)** $\mathcal{D}_t^{\text{DNN}}$ with $d^{\max}$ derived from it. **(c)** $\mathcal{D}_t^{\text{nov}}$ with d^{con} derived from it. **(d)** $\mathcal{D}_t^{\text{DNN}}$ with d^{con} derived from it.

reasonable as well (we use the implementation freely available in [3]). We denote the resulting activation curves by a^{DNN} and the deviation function by $\mathcal{D}_t^{\text{DNN}}$, respectively.

We also have several choices concerning the derivation of the deviation sequence d from $\mathcal{D}_t$. A straight-forward way is to simply pick the deviation that yields the highest activation value in each column of $\mathcal{D}_t$ as

$$d_m^{\max} := \operatorname*{argmax}_n \mathcal{D}_t(n, m) \ .$$

While this method considers every tap individually, it is also possible to incorporate some contextual information into the derivation by defining

$$d^{\text{con}} := \operatorname*{argmax}_{[d_0,\ldots,d_{M-1}]} \mathcal{D}_t(d_0, 0) \prod_{m=1}^{M-1} \mathcal{D}_t(d_m, m)\, \mathcal{T}(d_{m-1}, d_m),$$

with $\mathcal{T} : \mathbb{Z} \times \mathbb{Z}$ being a transition function defined by

$$\mathcal{T}(i, j) := e^{-\lambda|i-j|}$$

with $i, j \in \mathbb{Z}$. The sequence d^{con} can be derived using dynamic programming. The idea, inspired by [14, 22], is that subsequent human taps are unlikely to drastically vary in their deviation from the actual beat. To reflect this, the use of the transition function $\mathcal{T}$ makes it unlikely to have large deviation jumps from one tap to the next (we use λ=0.1 in our experiments). Furthermore, this method also allows us to correct *non-event beats* [18], meaning taps for which no cue in the music recording exists. In this case, the activation curve shows no salient values around the tap and $\mathcal{T}$ favors a constant deviation until there are clear cues in the activation curve again.

Figure 3 shows the two types of deviation functions $\mathcal{D}_t^{\text{nov}}$ and $\mathcal{D}_t^{\text{DNN}}$ in combination with the two methods for deriving the deviation sequences $d^{\max}$ and d^{con}. We use item `011` in our dataset as an example (see Table 1). It is a rather old recording featuring singing voice, acoustic guitar, mouth-organ, and piano. Comparing $\mathcal{D}_t^{\text{nov}}$ in Figure 3a/c to $\mathcal{D}_t^{\text{DNN}}$ as seen in Figure 3b/d, we can observe

item	YouTubeID	artist	song title
006	I3gHugP6bPE	Paul Daraiche	T'envoler
009	hRFLv29K__o	La Sonora Dinamita	Escandalo
011	M7u5SdjDSQQ	The Lovin' Spoonful	Daydream
025	8jsFGdeWNPo	Nicky Jam	Juegos Prohibidos
040	1zrxnqejwCk	Los Tigres Del Norte	Mañanitas Tapatias
046	ebZZpVFUQDY	Green Valley	Relaja
048	B_e7QbWc5mI	Orthodox Celts	Rocky Road To Dublin

Table 1. List of dataset items used throughout this paper. The YouTube videos can be found by using the URL `www.youtube.com/watch?v=[YouTubeID]`.

that $\mathcal{D}_t^{\text{nov}}$ is noisier while the structures seen in $\mathcal{D}_t^{\text{DNN}}$ are smoother and more salient. This becomes obvious when comparing the two $d^{\max}$ in Figure 3a and b. While the $d^{\max}$ based on $\mathcal{D}_t^{\text{nov}}$ jumps back and forth between deviations of about -0.2 and $+0.2$ seconds, $d^{\max}$ for $\mathcal{D}_t^{\text{DNN}}$ is more stable, showing only a few jumps between tap indices 115 and 135. The strength of using contextual information in the computation of the deviation sequence is visible in Figure 3c, where we see d^{con} based on $\mathcal{D}_t^{\text{nov}}$. Here, similar as in Figure 3b, most deviation values cluster around -0.2 seconds, except for a short passage of deviation 0 around tap index 135. This is caused by a single very strong activation value which does not coincide with a beat position in the recording. In the deviation function $\mathcal{D}_t^{\text{DNN}}$ this spurious activation is not present and the deviation sequence d^{con} in Figure 3d does not show prominent jumps. When listening to the sonified automatically corrected taps based on this deviation sequence, one can hear that they are in fact very accurate. Overall, we made similar observations for the vast majority of songs in our dataset. For this reason, we chose the combination of $\mathcal{D}_t^{\text{DNN}}$ with d^{con} in our subsequent experiments and also refer to them by just $\mathcal{D}_t$ and d for the sake of simplicity.

3. ANALYSIS OF AUTOMATED CORRECTIONS

Although the method introduced in the previous section is capable of automatically correcting the vast majority of taps, there is still potential for error. To find these errors efficiently, visualizing the deviation functions $\mathcal{D}_t$ and $\mathcal{D}_{\tilde{t}}$ can give helpful insights into the automatic correction process, point to problematic sections in music recordings, and reveal anomalies in the human-made taps. In Figure 4, we show several examples.

Figure 4a depicts the deviation functions of the original and automatically corrected taps for item `009`. This latin american song features strong and steady rhythmic pulses, which is reflected in the activation values visible in $\mathcal{D}_t$. In each column of $\mathcal{D}_t$, there is basically only one high activation value. The deviation sequence d nicely captures this train of high activations, which leads to a very clean deviation function $\mathcal{D}_{\tilde{t}}$. Note that the inter-tap-intervals in $\mathcal{D}_{\tilde{t}}$ are more regular than in $\mathcal{D}_t$, which can be seen by the envelope of $\mathcal{D}_{\tilde{t}}$ being less noisy than the one of $\mathcal{D}_t$. Based on this visualization, it is rather safe to assume that the corrected taps are accurate with virtually no errors. One thing noteworthy about this example is that d has a fairly constant offset of about -90 milliseconds, meaning that almost all original taps were about 90 milliseconds behind the beat. This was caused by technical problems in the process of recording the taps of the human annotator, which caused delays between the physical taps and the registered times.

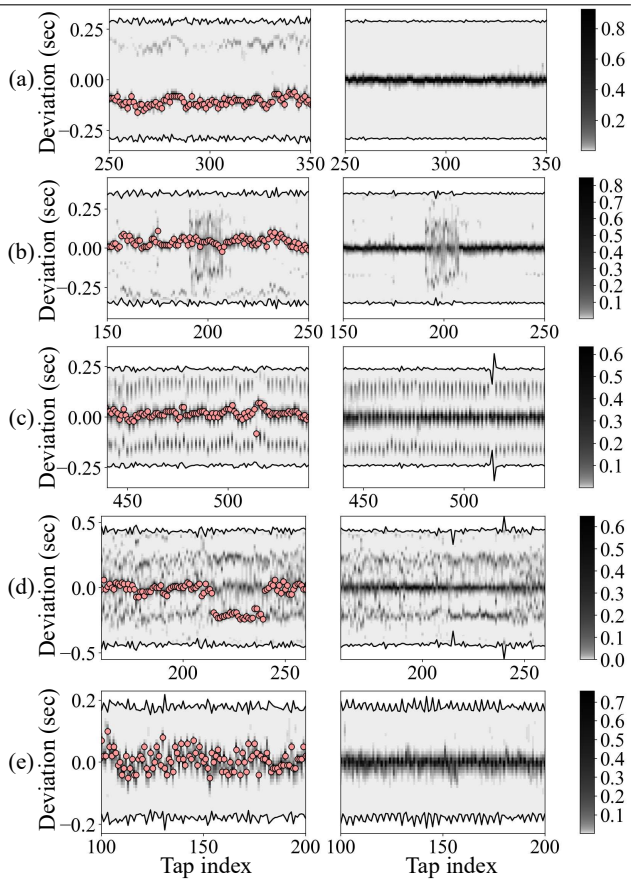


Figure 4. Deviation functions $\mathcal{D}_t$ (left) and $\mathcal{D}_{\tilde{t}}$ (right) for excerpts of different items from our proposed dataset. **(a)** Item `009` (2:25 to 3:23), **(b)** item `025` (2:12 to 3:22), **(c)** item `048` (3:30 to 4:18), **(d)** item `046` (2:21 to 3:49), **(d)** item `040` (0:37 to 1:13).

In Figure 4b, we see the deviation functions for item `025`. This hip hop song again has a very clear and prominent beat, which is reflected in the activations in $\mathcal{D}_t$. However, around the 200$^{\text{th}}$ beat, the song has a short part without any percussions. This manifests in $\mathcal{D}_t$ as a blurry section, which is caused by low and diffuse activation values. Since this indicates that there are fewer audio cues that the procedure can utilize to correct the original taps, such sections should be manually inspected after the automatic correction step. In this particular example the inspection showed that no manual corrections were necessary.

As a third example, we see the deviation functions for item `048` in Figure 4c. In the last part of this Irish folk song the bass drum plays a swing-like rhythm. This pattern causes the activation structures as seen in $\mathcal{D}_t$ with strong activations around deviation zero and weaker ones at about ± 200 milliseconds for every other tap. In the computation of the deviation sequence d, this lead to an error at tap in-

dex 510, where the tap was incorrectly snapped to one of the activations caused by the rhythmic ornaments. This single tap, misplaced by the correction procedure, can be easily detected in our visualization, since it caused a distinct structure in the envelope of $\mathcal{D}_{\tilde{t}}$, where it manifested in a "lightning" pattern.

A similar error can be seen in Figure 4d, which shows the deviation functions of an excerpt from item 046. This acoustic song featuring vocals, guitar, and keyboard is pretty challenging for beat tracking due to the lack of strong audio cues for beats, as can be seen by the rather noisy activations in $\mathcal{D}_t$. As in the previous example, the used rhythmic pattern, this time played by the guitar, causes strong activations at non-beat positions. In the computation of the deviation sequence d, these lead to a section of about 30 taps that were incorrectly snapped to these off-beat guitar accents rather than the actual beats. This can be seen by d being shifted to a deviation of about -0.2 seconds from tap index 210 to 240. The visualization of d allowed us to easily locate and understand this problematic passage, which could then be corrected manually in a post-processing step.

As a final example, Figure 4e shows the deviation functions of an excerpt from item 040. This Polka-like song has a $3/4$ time signature, but the third beat in each bar is consistently played late. [2] The human annotator tapped through the song in a rather straight fashion, not explicitly reflecting this rhythmic pattern. The automatic correction procedure then aligned each tap with the closest instrument onset, resulting in unevenly spaced corrected taps. This is visible as regular spike pattern in the envelope of $\mathcal{D}_{\tilde{t}}$. Whether the corrected taps reflect the "true" beat positions is dependent on whether one sees the delayed note onsets as part of the rhythm or mere ornamentation. Either way, this interesting example was easily revealed by our visualization of the deviation functions.

4. EXPERIMENTS

In this section, we evaluate our proposed procedure. We first introduce in Section 4.1 a dataset of beat annotations which we created using our correction procedure. Then, in Section 4.2, we discuss the results of a listening experiment to show that the corrected taps are actually perceived as being more accurate than the original taps. Finally, in Section 4.3, we show how the choice of annotation used as ground truth influences the evaluation of beat tracking algorithms.

4.1 The Dataset

In order to show the usefulness of our proposed tap correction procedure in a real-world annotation scenario, we applied it to create a new dataset of beat annotations. To this end, we selected 101 different music recordings available on YouTube. [3] This collection, which consists of about 7.25 hours of music in total, comprises a variety of different genres, recording conditions and instrumentations. Similar to [1, 23], we decided to use music recordings from YouTube to ensure reproducibility of our results. For each of these recordings, a musically experienced annotator tapped along to create the tap sequences t using Sonic Visualizer, adding up to 41.011 individual taps. The sequences of automatically corrected taps $\tilde{t}$ were then computed for each recording using the method described in Section 2. Each sequence $\tilde{t}$ was then manually inspected by the first author using Sonic Visualizer. In this step, a total of 331 incorrect taps were identified across 54 of the 101 sequences and corrected manually. We denote the resulting *fully corrected tap sequences* by $\tilde{t}'$.

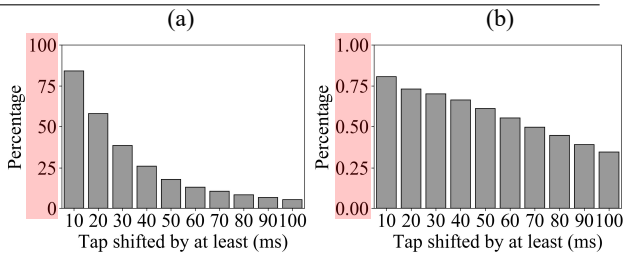


Figure 5. Percentage of the 41.011 original taps that were shifted in the two consecutive automatic and manual correction steps. Note the different scales of the vertical axes. **(a)** Creating $\tilde{t}$ from t, **(b)** creating $\tilde{t}'$ from $\tilde{t}$.

Figure 5 summarizes the distribution of shifts applied to the individual taps in the two consecutive correction steps. In Figure 5a we can see that about 25% of the original taps were shifted by 40 milliseconds or more by the automatic correction procedure. This means that in case these corrections would have been done manually, about one in four taps, would have been shifted—even when assuming that smaller inaccuracies where taps were misaligned with audio cues by less than 40 milliseconds would not have been considered. To create the fully corrected taps on the other hand, less than 1% of the 41.011 original taps were shifted at all, see Figure 5b. The shifts applied in the manual correction step were equally distributed between very small and larger shifts which can be seen by the rather constant slope of the graph.

The dataset containing all three annotations t, $\tilde{t}$, and $\tilde{t}'$ for each recording as well as metadata and the respective YouTube links is available at [13].

4.2 Listening Experiment

With the creation of the fully corrected taps $\tilde{t}'$, the main question is whether the applied corrections actually make the beat annotations perceptually more accurate—and therefore better. To answer this, we conducted a listening experiment. For each of the 101 recordings in our dataset, we created two new recordings by sonifying the taps t as well as the fully corrected taps $\tilde{t}'$ as a click track and superimposed each of them on the original mu-

[2] This observed rhythmic pattern is rather unusual for the style of the song but commonly found in Viennese Waltz.

[3] The dataset is part of a Chordify project that assesses the quality of automatically generated annotations in a large scale industry setting. The selected recordings reflect a random sample of songs used on Chordify, weighted by their popularity on the service.

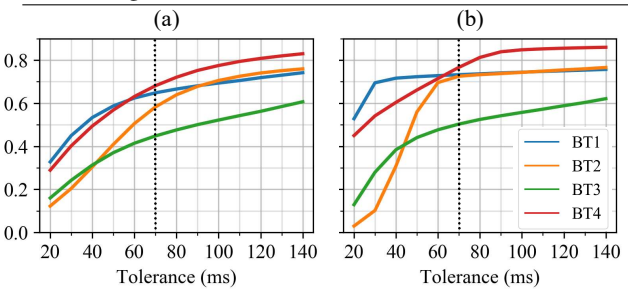


Figure 6. Average beat F-measures on our dataset for four different beat tracking algorithms when using different ground truth annotations. **(a)** Original taps t, **(b)** fully corrected taps $\tilde{t}'$.

sic recording. Note that we chose to use the fully corrected taps $\tilde{t}'$ rather than the automatically corrected taps $\tilde{t}$ because the difference between $\tilde{t}$ and $\tilde{t}'$ is very small and we did not want the participants to focus on the few noticeable mistakes made by the automatic correction procedure. Three musically trained people took part in the experiment, none of them being one of the authors. Each participant was presented with the 101 pairs of recordings and asked to pick the recording with the more accurate click track from each pair. The order in which the two recordings of a pair were presented was random and the participants did not know whether the clicks they heard were the original taps t or the fully corrected taps $\tilde{t}'$. They were able to listen to each recording for as long and as often as they liked before making their decision. Additionally, they had the opportunity to give comments. The complete set of answers and comments can be found at [13]. Looking at all 303 given answers individually (three participants times 101 pairs), 89% of the time the participants found the fully corrected taps $\tilde{t}'$ to be more accurate than the original taps t. For 72% of the 101 pairs, all three participants even consistently perceived $\tilde{t}'$ to be more accurate than t. Having a closer look at the participants' answers and comments, it turned out that in many instances where a participant chose the original taps to be more accurate than the fully corrected taps, the two click tracks were perceived as being very similar (*"Both are about the same quality, IMO."*). Furthermore, some of the comments also indicate that sometimes there was also a degree of personal preference involved in the decision (*"...[the click track] lags like a proper gospel drummer."*). Overall, the results show that the fully corrected taps $\tilde{t}'$ are commonly perceived as more accurate than the original taps t.

4.3 Quantitative Study

As a final experiment, we were interested in the effects of using either the original taps t or the fully corrected taps $\tilde{t}'$ as ground truth for a quantitative evaluation of beat tracking algorithms. We investigated four different algorithms: The *Queen Mary beat tracker* (BT1) [10], the *librosa beat tracker* (BT2) [14], the *Aubio beat tracker* (BT3) [8, 9], and our own implementation of [4] (BT4). Note that the madmom beat tracker [3,4], which would have an intrinsic advantage since our tap correction procedure is built upon the same activation curve, is not among them. For each of the four beat tracking algorithms, we computed the *beat F-measure* [11], a popular beat tracking evaluation measure, for all recordings in our dataset. We did this two times, once taking the original taps t as ground truth and once the fully corrected taps $\tilde{t}'$, see Figure 6a and 6b, respectively. An important parameter in the computation of the beat F-measure is the tolerance, which determines the maximal temporal distance between an estimated beat and a ground truth beat such that the estimate can be considered correct. In Figure 6, we see that for a very large tolerance (140 milliseconds and above) it basically makes no difference whether we use t or $\tilde{t}'$ as ground truth. This makes sense when recalling that most of the corrections applied were smaller than 100 milliseconds (see Section 4.1). However, this changes when considering a smaller, and hence more realistic tolerance. The default tolerance as implemented in *mir_eval* [28], the Python library we used for this evaluation, is 70 milliseconds, indicated by dotted black lines in Figure 6. At this tolerance, the computed beat F-measures differ noticeably depending on the ground truth. For example, BT4 achieves an average beat F-measure of 0.68 when the original taps t are used as ground truth, but 0.77 when the fully corrected taps $\tilde{t}'$ are used. The difference is even more prominent when comparing BT1 and BT2. Here, BT1 scores much better (0.65) than BT2 (0.58) when comparing the two algorithms based on the original taps. However, when using the fully corrected taps as ground truth, the algorithms perform nearly identically (both 0.73). For smaller tolerances, the differences between the individual algorithms become even more salient. For example, at a tolerance of 30 milliseconds and using the original taps as ground truth, BT1 and BT2 differ in beat F-measure by 0.25 (scoring 0.45 and 0.20, respectively), while differing by 0.59 (scoring 0.69 and 0.10, respectively) on the fully corrected taps. This shows that it can make a substantial difference for quantitative evaluations of beat tracking algorithms whether the underlying ground truth annotations are "only" human-made taps or corrected ones.

5. CONCLUSIONS

In this paper we proposed and formalized a simple procedure for correcting tapped beat annotations by automatically "snapping" the tap positions to cues in the underlying music recording. Furthermore, we proposed a visualization that can help identifying errors made by the correction procedure as well as rhythmically interesting passages in music recordings. Finally, we created a new dataset for which we showed that beat annotations corrected with our procedure are perceived as being more accurate and that using them for the quantitative evaluation of beat tracking algorithms may significantly impact the evaluation results. The last observation motivates us to apply our proposed correction procedure to beat annotated datasets commonly used in the MIR community. We believe that our proposed visualization could help in identifying incorrectly annotated recordings and therefore getting a more realistic view on the performance of beat tracking algorithms.

6. ACKNOWLEDGMENTS

We would like to thank Leigh Smith and Sebastian Böck for our discussions at ISMIR 2018 that sparked the ideas for this contribution. Furthermore, we thank Jeroen Bransen for the discussions about the algorithmic details of the procedure, Jeffrey van Rossum for creating the original taps for the recordings in the dataset, as well our listening experiment participants Tijmen Ruizendaal, Matúš Tejiščák, and Bart Laan. Hendrik Schreiber and Meinard Müller are supported by the German Research Foundation (DFG MU 2686/10-1). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

7. REFERENCES

[1] S. Balke, C. Dittmar, J. Abeßer, K. Frieler, M. Pfleiderer, and M. Müller. Bridging the gap: Enriching YouTube videos with jazz music annotations. *Frontiers in Digital Humanities*, 2018.

[2] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1047, 2005.

[3] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer. madmom: A new python audio and music signal processing library. In *Proceedings of the ACM Conference on Multimedia Conference*, pages 1174–1178, Amsterdam, The Netherlands, 2016.

[4] S. Böck, F. Krebs, and G. Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *Proceedings of the 17th International Conference on Music Information Retrieval (ISMIR)*, pages 255–261, New York City, United States, 2016.

[5] C. Cannam, C. Landone, and M. Sandler. Sonic Visualiser: An open source application for viewing, analysing, and annotating music audio files. In *Proceedings of the ACM Multimedia 2010 International Conference*, pages 1467–1468, Firenze, Italy, October 2010.

[6] E. Clarke. Rhythm and timing in music. *The Psychology of Music*, 2:473–530, 12 1999.

[7] N. Collins. A comparison of sound onset detection algorithms with emphasis on psychoacoustically motivated detection functions. In *AES Convention 118*, Barcelona, Spain, 2005.

[8] M. E. P. Davies, P. Brossier, and M. D. Plumbley. Beat tracking towards automatic musical accompaniment. In *Proceedings of the Audio Engineering Society 118th Convention*, Barcelona, Spain, May 2005.

[9] M. E. P. Davies and M. D. Plumbley. Causal tempo tracking of audio. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, Barcelona, Spain, October 2004.

[10] M. E. P. Davies and M. D. Plumbley. Context-dependent beat tracking of musical audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(3):1009–1020, 2007.

[11] S. Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30:39–58, 2001.

[12] S. Dixon. Evaluation of the audio beat tracking system BeatRoot. *Journal of New Music Research*, 36:39–50, 2007.

[13] J. Driedger, H. Schreiber, W. B. de Haas, and M. Müller. Accompanying material: Towards automatically correcting tapped beat annotations for music recordings. `https://github.com/chordify/tapcorrect`.

[14] D. P. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[15] V. Eremenko, E. Demirel, B. Bozkurt, and X. Serra. Audio-aligned jazz harmony dataset for automatic chord transcription and corpus-based research. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 483–490, Paris, France, September 2018.

[16] M. Fuentes, B. McFee, H. C. Crayencour, S. Essid, and J. P. Bello. Analysis of common design choices in deep learning systems for downbeat tracking. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 106–112, Paris, France, September 2018.

[17] A. Gkiokas and V. Katsouros. Convolutional neural networks for real-time beat tracking: A dancing robot application. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 286–293, Suzhou, China, October 2017.

[18] P. Grosche, M. Müller, and C. S. Sapp. What makes beat tracking difficult? A case study on Chopin Mazurkas. In *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, pages 649–654, Utrecht, The Netherlands, 2010.

[19] S. Handel. *Listening: An Introduction to the Perception of Auditory Events*. A Bradford book. MIT Press, 1993.

[20] A. Holzapfel, M. E. P. Davies, J. R. Zapata, J. Oliveira, and F. Gouyon. Selective sampling for beat tracking evaluation. *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, 20(9), 2012.

[21] A. Holzapfel, F. Krebs, and A. Srinivasamurthy. Tracking the "odd": meter inference in a culturally diverse music corpus. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, pages 425–430, Taipei, Taiwan, October 2014.

[22] F. Krebs, S. Böck, and G. Widmer. An efficient state-space model for joint tempo and meter tracking. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 72–78, Málaga, Spain, 2015.

[23] P. López-Serrano, C. Dittmar, and M. Müller. Finding drum breaks in digital music recordings. In *Proceedings of the International Symposium on Computer Music Modeling and Retrieval (CMMR)*, Porto, Portugal, 2017.

[24] U. Marchand and G. Peeters. Swing ratio estimation. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Trondheim, Norway, 2015.

[25] M. Mauch, C. Cannam, M. E. P. Davies, S. Dixon, C. Harte, S. Kolozali, D. Tidhar, and M. Sandler. OMRAS2 metadata project 2009. In *Late-breaking session at the 10th International Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan, 2009.

[26] M. F. McKinney and D. Moelants. Ambiguity in tempo perception: What draws listeners to different metrical levels? *Music Perception*, 24(2):155–166, 2006.

[27] M. Müller. *Fundamentals of Music Processing*. Springer Verlag, 2015.

[28] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis. MIR_EVAL: A transparent implementation of common MIR metrics. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*, pages 367–372, Taipei, Taiwan, October 2014.

[29] B. H. Repp. Sensorimotor synchronization: A review of the tapping literature. *Psychonomic Bulletin & Review*, 12(6):969–992, 2005.

[30] C. Weiß, V. Arifi-Müller, T. Prätzlich, R. Kleinertz, and M. Müller. Analyzing measure annotations for western classical music recordings. In *Proceedings of the 17th International Conference on Music Information Retrieval (ISMIR)*, pages 517–523, New York, USA, 2016.

ALGORITHMIC ABILITY TO PREDICT THE MUSICAL FUTURE: DATASETS AND EVALUATION

Berit Janssen[1] **Tom Collins**[2,3] **Iris Yuping Ren**[4]
[1] Digital Humanities Lab, Department of Humanities, Utrecht University, The Netherlands
[2] Music, Science and Technology Research Cluster, Department of Music, University of York, UK
[3] Music Artificial Intelligence Algorithms, Inc., Davis, CA, USA
[4] Department of Information and Computing Sciences, Utrecht University, The Netherlands
{berit.janssen@gmail.com, tomthecollins@gmail.com, y.ren@uu.nl}

ABSTRACT

Music prediction and generation have been of recurring interest in the field of music informatics: many models that emulate listeners' musical expectancies, or that produce novel musical content have been introduced over the past few decades. So far, these models have mostly been evaluated in isolation, following diverse evaluation strategies. Our paper provides an overview of the new MIREX task *Patterns for Prediction*. We introduce a dataset, which contains monophonic and polyphonic data, both in symbolic and audio representations. We suggest a standardized evaluation procedure to compare algorithmic musical predictions. We compare two neural network models to a baseline model and show that algorithmic approaches can correctly predict about a third of a monophonic segment, and around half of a polyphonic segment, with one of the neural network models achieving best results. However, other approaches to algorithmic music prediction are needed to achieve a more rounded picture of the potential of state-of-the-art methods of music prediction.

1. INTRODUCTION

Prediction of future events is fundamental to human and artificial intelligence, and has therefore been discussed as a core research interest bridging cognitive psychology and machine learning [5]. Music, with its complex event sequences extending over time, provides an excellent setting for the study of prediction.

In music cognition, human prediction of future events, or *expectation*, is studied from theoretic, behavioral, imaging, and modeling perspectives. Some theoretical work [2, 15] distinguishes between *veridical*, *schematic*, and *dynamic* expectations: veridical expectations occur due to familiarity with a musical piece, schematic expectations are elicited by familiarity with a genre, and dynamic expectations manifest in-the-moment predictions, when, consciously or subconsciously, a listener becomes attuned to a pattern in a novel piece. It has been claimed that the pleasure we derive from music resides in the tension between these three forms of expectation [15].

While the full complexity of expectation in music may still be hard to capture in computational models, the goal of this paper is to give an overview of how computational models may emulate human expectations through prediction of future musical events, and how we should evaluate such models.

Our main contributions are as follows: first, we review different approaches to modelling expectation in music. Second, we introduce a dataset on which such models can be trained and evaluated. Third, we propose two evaluation tasks and associated evaluation measures. [1] Fourth, we provide the results of a baseline and two more complex models on the tasks. Finally, we discuss findings and recommendations for future model development and evaluation.

2. RELATED WORK

2.1 Approaches to music prediction

2.1.1 Markov models

Markov models have been influential in music prediction: statistics on transitions between musical events may be used to generate predictions for unseen musical events. Musical events may be represented in various ways, such as pitch, duration, onset, metric weight, and so on. Therefore, it has been suggested to build distinct models for different combinations of music representations [11]. Markov models trained on music corpora may very well serve to model schematic expectation, such as a leading tone to be followed by an octave. Dynamic (i.e., work-specific) expectations may also be modelled through a Markovian approach through training on a musical piece itself, where the model is incrementally updated as the piece progresses. The question of how to combine models of various music representations, or how to combine models trained on corpora ("long-term models") and models trained incremen-

[1] The corresponding MIREX task description, datasets, and evaluation code can be found at https://tinyurl.com/y455cf97

tally on a piece ("short-term models") has been experimentally investigated [26], but on specific styles, which might not generalize to other musical genres.

Even though some models can theoretically extend over very long contexts, the question remains whether Markovian models, which are by nature "forgetful", will capture longer structure that may facilitate precise predictions, such as repetition of themes in a Classical piece of music, or the return of the theme in a jazz performance. [36]

2.1.2 Neural networks

Over the last two decades, interest in neural network models for music prediction has been increasing. The first attempts in this direction made use of recurrent neural networks (RNNs) [23], with an input, hidden and output layer, which predict future states of sequential input data. Variants of RNNs, such as long short-term memory models (LSTM), have also been applied to music [13]. Various extensions of such models have been presented since [4, 14].

Another class of neural networks, convolutional neural networks (CNN), is usually used for image data. A musical composition may also be thought of as an image rendered in time-pitch or time-amplitude space. Some authors therefore applied CNNs to piano roll representations of symbolic music [12], or to audio [34] for music prediction.

While these and other neural network architectures have resulted in generation and prediction of music, the output of the models in itself is often not easy to predict. One of the challenges for neural network models, as for Markovian models, is the degree to which they can capture large-scale structure in a musical piece, and recreate dynamic expectations that may arise within a piece in itself.

2.1.3 Pattern discovery

Yet another approach to predicting the musical future is to search for repeating patterns within the piece. This approach emulates dynamic expectations of a listener (patterns occur in earlier parts of a piece, leading to predictions based on later, partially complete occurrences of the same patterns [6, 26]), but less so schematic expectations.

Various algorithms have been proposed to discover repeated patterns within a piece [7, 18, 22, 28, 30, 35], which differ in the kinds of patterns they aim to discover, in the way music is represented, and in the algorithms used to find repetitions [16]. These algorithms have been tested on benchmarks of annotated patterns, while evaluation by prediction is suggested but yet-to-be implemented [21].

2.2 Evaluating music prediction

To ascertain how models compare to human expectations, various approaches have been used: some of them fall in the domain of music generation, while others fall in the domain of information theoretic measures.

2.2.1 Information-theoretic measures

In order to investigate musical predictions of a model with information theoretic measures, the model is trained on a corpus or a corpus subset, then exposed to a novel musical piece. For each musical event, the likelihood of that event according to the model is measured. Alternatively, the uncertainty of the model after each note in predicting the following melody note may be recorded. Ratings of likelihood or uncertainty may then be compared to human ratings from experimental research.

To compare likelihood as rated by a model to human ratings, priming experiments may serve as an evaluation ground: in such experiments, participants had to give an indication as to how well, given a melodic context, a note fitted their expectations [17, 31]. There were also experiments on uncertainty, in which participants were asked to indicate how uncertain they were of what might follow each note in a Bach chorale [19]. As phrase boundaries often coincide with points of greater uncertainty, human segmentations have also been occasionally used as a ground truth for evaluating model predictions [27].

2.2.2 Music generation

A very common way to evaluate predictions from a model is the demonstration of music generated by a given model (e.g., [11]). While this is informative, it is not self-evident how to judge the quality of such an output. Music practitioners of a given genre may be asked as judges [32], but aesthetic judgments alone may not reveal much about a model's shortcomings [1].

Another approach to evaluating music generation is through comparing generated music with human-composed melodies. Human compositions can then be used as the touchstone of how well a model captures structure and style [20]. We choose this approach by providing models with a short piece of music, or prime, and instruct the models to generate a continuation, which we evaluate against the true continuation. Moreover, we test how successfully a model distinguishes between the true continuation and an artificial, foil continuation.

3. A DATASET FOR MUSIC PREDICTIONS

3.1 Dataset construction

We provide small, medium, and large development datasets (100, 1,000, and 10,000 pieces, respectively). This caters to different approaches to designing models for the task, some of which are more data-intensive than others. Each dataset has audio/symbolic and monophonic/polyphonic variants.

Pieces were selected at random from the Lakh MIDI Dataset (LMD) [29] with the aim of creating primes lasting $\approx 35\,\mathrm{sec}$ according to tempo information. True continuations were selected – and foil continuations generated – such that onsets (note start times) occurred in a 10 quarter-note-beat window.

We also constructed a test dataset from another corpus of MIDI files, which is similar in nature to LMD. The test dataset also contains audio/symbolic and monophonic/polyphonic variants, and provides primes, true and foil continuations. In keeping with the MIREX guidelines,

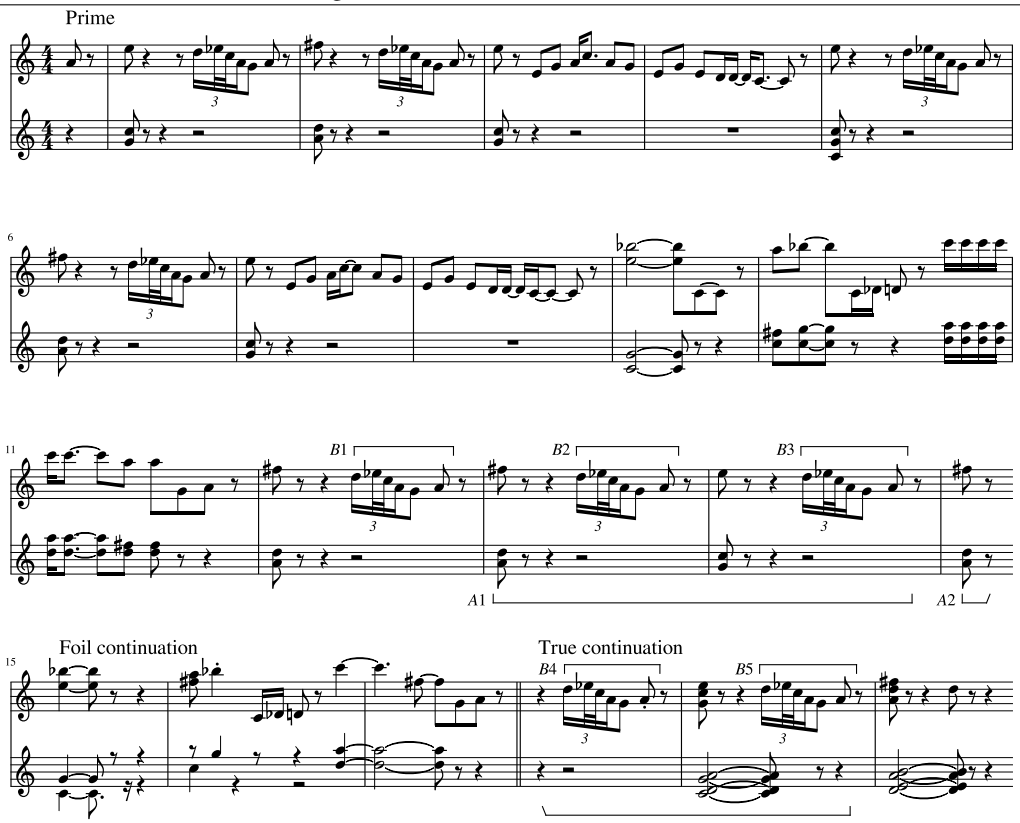


Figure 1. Item from the large, polyphonic variant of the development dataset. Musical provenance unknown.

we will not reveal details about the provenance of the test dataset. What we can say is that the source for the test dataset contains approximately $30,000$ files, and that both this source and LMD are gathered from sites that represent musical interests and tastes, broadly construed. The number of items in the test source tagged as "pop", for instance, is $9,165$. Other items have similar tags, however, such as "latin pop". To our knowledge, no such analysis of genres exists for the LMD, so remarks about the overlap of genres and content between our training and test datasets are necessarily speculative.

MIDI files were selected at random, imported using midi-convert, [2] quantized, and then the following criteria were applied when generating monophonic primes and continuations:

1. A prime had to contain at least 20 notes;
2. The maximum inter-onset interval in a prime could not exceed 8 quarter-note beats;
3. A continuation had to contain at least 10 notes;
4. The channel from which material was selected had to be suitably monophonic prior to *skylining* (see below), meaning at least 80% of minimal segments [25] had to be single notes or rests.

Skylining means to select the highest-sounding notes at each onset and return only those notes, perhaps with modified durations, so that the output is truly monophonic. The rationale for only skylining material that was already 80% or more monophonic is that skylining inherently polyphonic material often results in odd-sounding or implied-polyphonic output. For polyphonic dataset generation, criteria (1)-(3) were the same, but a replacement for (4) was needed because parsed MIDI files sometimes contain erroneously long notes. In place of criterion (4), polyphonic dataset generation involved clipping any notes longer than 8 quarter-note beats.

If a prime or continuation did not satisfy one or more of the above criteria, generation proceeded to the next randomly selected piece (rather than, say, selecting a different excerpt from the same piece). Approximately $1/6$ random selections passed the criteria, meaning we had to process $6N$ pieces to produce a dataset of size N.

Our baseline for generating foil continuations is the Racchman-Jun2015 model [9] described in section 4. Since previous work has emphasized the need to progress from Markovian approaches to modeling music [36], this seemed to be the most appropriate baseline. Examples of a prime, true and foil continuations are shown in Figure 1. This excerpt came from an LMD song called "Dirtyluv". We were unable to identify further title or artist information – an issue when working with this source.

Returning to the discussion of pattern discovery for prediction, the example is annotated with pattern occurrences $A_1, A_2, B_1, B_2, \ldots, B_5$ to indicate how such an approach would be fruitful in this case. The annotating lines are placed above and below the stave for clarity, but encompass notes from both staves that begin in the indicated time spans. The prime ends by repeating the first 3 notes of A_1.

[2] `https://www.npmjs.com/package/@pioug/MidiConvert`

Therefore, one reasonable prediction for the continuation is that it will proceed to restate the remainder of A_1. Comparing such a prediction to the true continuation, we see that it would be quite successful – some extra notes in the left hand in measure 16 are the only difference between A_1 and A_2. In an analogous fashion, the regularity of occurrences of B_1, B_2, B_3 could be used to make a prediction about B_4 and B_5 appearing in the true continuation.

3.2 Dataset characteristics

Figure 2 contains three violin plots showing basic distributional characteristics of the symbolic, monophonic, medium-size development ("dev") dataset and the symbolic, monophonic test dataset. Inspection of these plots suggests that the development and test datasets share similar characteristics. A slightly more marked peak can be seen around inter-onset 0.5 in the primes and true continuations of the test compared to the development datasets (Figure 2A), and the test dataset has a slightly lower mean MIDI note number than that of the development dataset (Figure 2C). While the test dataset is separate from the development dataset (LMD) and it would be inappropriate to report the extent to which they overlap in terms of content, evidently their distributional characteristics are similar. It is worth noting that there is healthy representation of "bass lines" in monophonic – but not polyphonic – variants of the datasets (see the modal concentration around MIDI note 35 in Figure 2C), as a result of the selection criteria outlined above.

4. COMPARED MODELS FOR MUSIC PREDICTION

We compare the output of foil continuations by the first-order Markov model (see section 3.1), in the following referred to as *baseline* model, with two recurrent neural network models, *BachProb* [10] and *Seq2SeqP4P* [24].

BachProb is a deep-gated, recurrent neural network with three consecutive layers. Notes are represented as triplets of pitch, duration and inter-onset interval with respect to the previous note. Durations and inter-onset intervals are rounded to durations commonly found in musical scores. *BachProb* is trained on the development dataset using truncated back propagation, with separate models for the monophonic and the polyphonic parts of the dataset.

Seq2SeqP4P is a long short-term memory network with two layers. Music was represented as the MIDI commands *note-on*, *note-off*, which define when a given pitch starts or ends, and time shifts between those commands, quantized to 12 subdivisions per beat. Such a sequence of MIDI commands and time shifts was used as the input to training the model on the development dataset. By virtue of design, Seq2SeqP4P was trained only on the monophonic part of the dataset.

The baseline model consists of a first-order Markovian generator nested in other processes intended to ensure the output has long-term repetitive and phrasal structure [9]. The state space consists of beat of the mea-

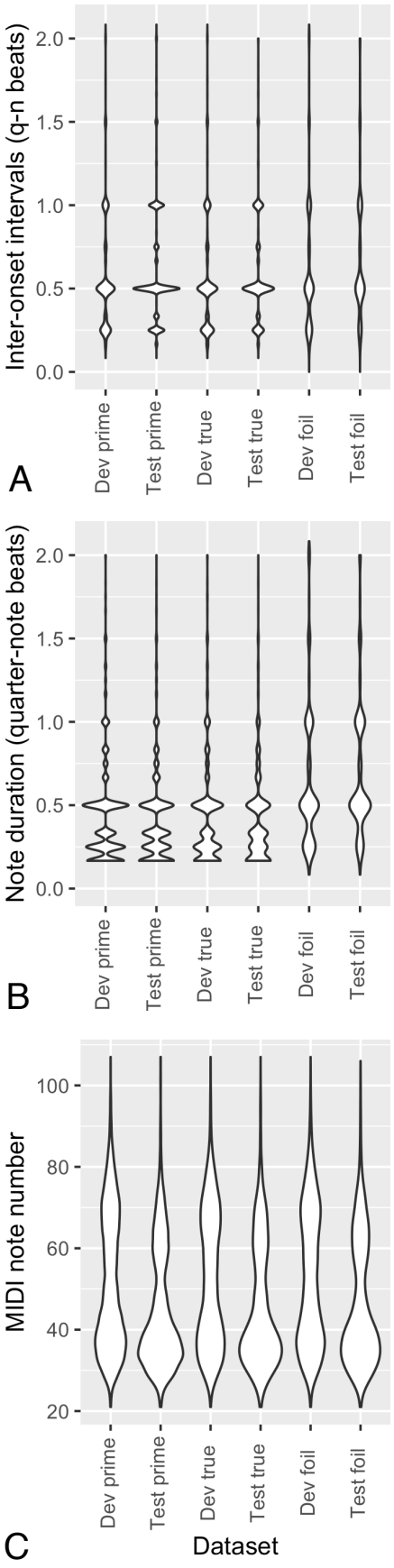


Figure 2. Characteristics of the symbolic, monophonic, medium-size dataset: (A) Inter-onset interval distributions of development and test datasets; (B) Duration distributions; (C) MIDI note number distributions.

sure on which notes occur, and MIDI note numbers relative to tonal center. The Markov generator alone, called Racchman-Jun2015, is a useful benchmark, because any longer structures that emerge here do so by chance.

5. EVALUATION

We evaluate music prediction in two ways:

- Explicit task. Models are provided with a prime, from which they generate continuations. The output of the models is then judged according to how many events of the true continuation they correctly predicted (metric definitions below).
- Implicit task. Models are provided with correct and foil continuations after a prime, from which they have to select the correct continuation.

5.1 Explicit task

For the explicit task, the evaluation proceeds as follows: within a time interval of ten quarter notes, we step through the time line by small time increments. We choose a time increment of $t = 0.5$ quarter notes, i.e., an eighth note.

We represent each note in the true and algorithmic continuation as a point in a two-dimensional space of onset and pitch, giving the point-set $\mathbf{P}$ for the true continuation, and $\mathbf{Q}$ for the algorithmic continuation. We calculate differences between all points p_i in $\mathbf{P}$ and q_j in $\mathbf{Q}$, which represent the translation vectors $\mathbf{T}$ to transform a given algorithmically generated note into a note from the true continuation [8, 33].

We then search for the largest set match achievable through translation with any vector, leading us to the number of correctly predicted notes cp:

$$\text{cp}(\mathbf{P}, \mathbf{Q}) = \max_{\mathbf{T}} |\{q_j | q_j \in \mathbf{Q} \wedge q_j + \mathbf{T} \in \mathbf{P}\}| \quad (1)$$

We define recall as the number of correctly predicted notes, divided by the cardinality of the true continuation point set $\mathbf{P}$. Since there exists at least one point in $\mathbf{Q}$ which can be translated by any vector to a point in $\mathbf{P}$, we subtract 1 from numerator and denominator to scale to $[0, 1]$.

$$\text{Rec} = (\text{cp}(\mathbf{P}, \mathbf{Q}) - 1)/(|\mathbf{P}| - 1) \quad (2)$$

Precision is the number of correctly predicted notes, divided by the cardinality of the point set of the algorithmic continuation $\mathbf{Q}$, scaled in the same way:

$$\text{Prec} = (\text{cp}(\mathbf{P}, \mathbf{Q}) - 1)/(|\mathbf{Q}| - 1) \quad (3)$$

The F_1-score is the harmonic mean of precision and recall. As the measures we propose are not defined for cases in which either the true or the algorithmic continuation contain fewer than two events, we start evaluation from onset 2.0, i.e, two quarter notes after the end of the prime, which ensures long enough sequences.

5.2 Implicit task

In the implicit task, a prediction model has to judge which of two continuations after a given prime is the true continuation. The foil is generated by the baseline model (see Section 4).

To evaluate the implicit task, we measure the success rate, i.e., the number of cases in which the model correctly picks the true continuations, divided by the total amount of decisions undertaken by the model.

6. RESULTS

Our evaluation and results focus on the symbolic variants of our test dataset. For the monophonic part of the dataset, we compare all three models, whereas for the polyphonic part, we only compare *BachProb* against the baseline, as *Seq2SeqP4P* has not been trained on polyphonic data yet.

6.1 Explicit task

For the monophonic dataset, the various models predict around a third of the events correctly, with *BachProb* outperforming the baseline and *Seq2SeqP4P* slightly, especially at the start of the predicted continuation (see Figure 3B).

Seq2SeqP4P predicts more of the notes in the true continuation correctly than the baseline, but at the cost of precision (Figure 3A), which means that its F_1 score is also lower overall than the baseline (Figure 3C).

For the polyphonic dataset, *BachProb* achieves a much higher recall than the baseline Markov model (Figure 3E). In precision, it performs close to the baseline, which results in very similar F1-scores, too (Figure 3D, F).

In general, the recall, precision, and F_1 score of the models decrease as the onset of the generated continuation increases, even though the baseline model has fairly stable performance over the evaluated time interval for the monophonic dataset, and *Seq2SeqP4P* increases in performance at the start of the continuation.

6.2 Implicit task

BachProb achieves a success rate of 0.85 for the monophonic continuations, i.e., 85% of the true continuations were identified correctly. For the polyphonic continuations, *BachProb* scores a success rate of 0.90. According to the binomial distribution, a success rate of 0.54 or higher constitutes above-chance performance on this task. At present, *Seq2SeqP4P* has not been implemented for the implicit task, so there are no results for it at this stage.

7. DISCUSSION

How events in the recent or more distant past may be appraised – consciously or otherwise – so as to be better adapted for what lies ahead is a phenomenon that has intrigued researchers from diverse disciplines such as cognitive psychology, philosophy, computer science, and music. In this paper, we focussed on music as a vehicle for studying the ability of computational models to predict continuations of given primes, and described datasets, evaluation procedures, and results to this end.

BachProb, utilizing a gated recurring neural network, outperforms the other two models. *Seq2SeqP4P*, based

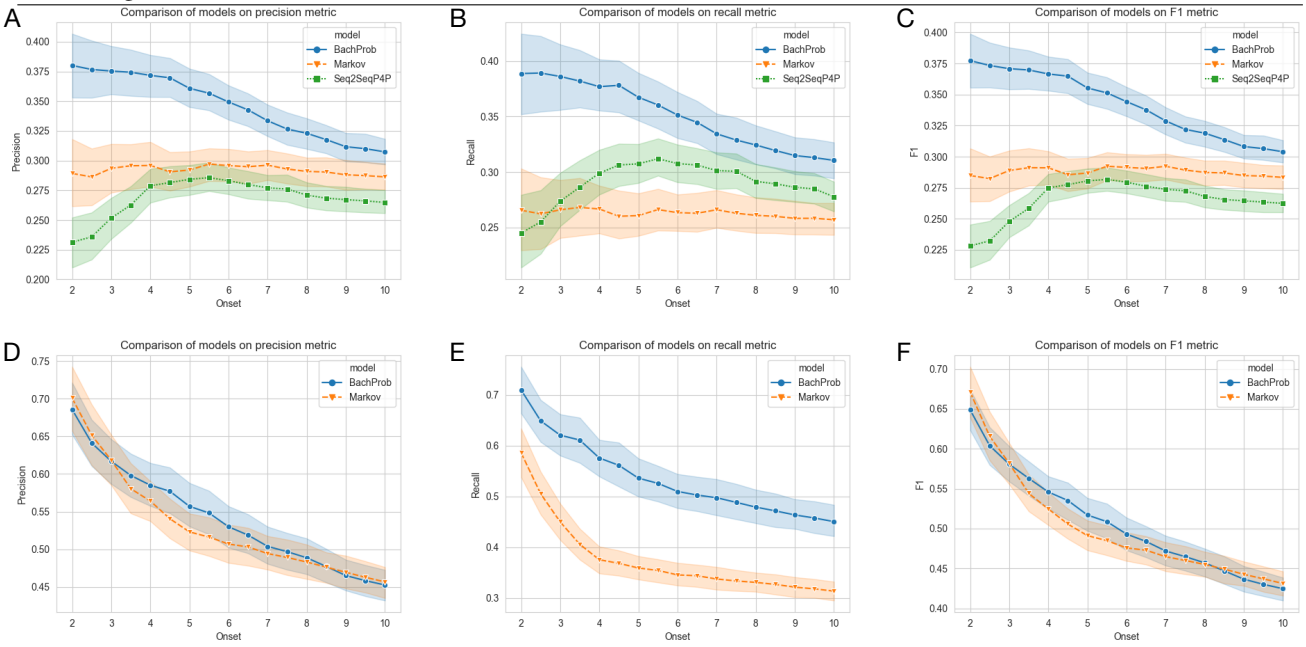


Figure 3. (A) Precision, (B) recall, and (C) F_1-measure of the three monophonic prediction models; (D) precision, (E) recall, and (F) F_1-measure of the two polyphonic prediction models. Evaluation measures are not defined for very short sequences, so evaluation starts at onset 2.0. Shading around the curves indicates one standard error from the mean.

on a long-short term memory model, predicts less musical material correctly. Arguably, music representation may be a larger factor in this than network architecture: as the model reportedly produces repeated pitches in some cases, and tends to always assign the same durations between the note-on and note-off events [24], the sequences of decoupled pitch and duration related information may not be suitable for the model to learn musical structure. Repeated short durations in the output may also explain the high recall and low precision of the model.

The overall higher recall, precision and F1-score of *BachProb* and the baseline on the polyphonic dataset, as compared to the monophonic results, is surprising. A possible explanation may be that repetitive chords, for instance in a piano or guitar part, are frequently present in the dataset and may be relatively easy to predict. The comparatively lower precision of *BachProb* suggests that while many notes from the true continuation are generated, there are also many spuriously generated notes.

For the implicit task, it is remarkable that *BachProb* distinguishes between true and foil continuations highly above chance. While the explicit task shows that the Markovian continuation reproduces a modest percentage of notes in the true continuations, the implicit task shows that *BachProb* learned details of the musical structure which could not be emulated by the Markovian foil.

We hope to evaluate more models for music prediction in the future, which might give us more insights into what constitutes successful prediction. As such, our proposal of a dataset and evaluation measures opens up the ground to discussion of how comparison of music prediction models may be improved.

First, we need to consider improved, or additional evaluation measures for the explicit task: our current approach to evaluating the explicit task entails that algorithmic continuations will be evaluated as correct continuations even if they are shifted in onset or pitch. The proposed measures may also penalize deviations from a true continuation that might be almost imperceptible to a human listener, such as an added chord note, or the reordering of chord tones.

Second, the evaluation of the implicit task also needs to be reconsidered: it depends heavily on the quality of the foil continuation. Perhaps the Markov baseline generates material which is too easily distinguishable from the true continuation. Moreover, we measure success rate, which has the advantage of easy interpretation, but does not take into account a model's confidence in its distinction between the true and foil continuation. Alternative foil continuations, or more fine-grained measures of the models' distinctions, would certainly give additional insights on model performance.

Third, additional tasks and datasets may be needed. We envision bringing together outcomes of music prediction models with evidence on human expectations in music. The continued systematic comparison of various models for music prediction can teach us much about the successes and shortcomings of prediction models in relation to each other, as well as about the influence of music representation and model parameters. Studies which measure human responses on their levels of surprise when hearing the continuation of a musical prime [19], or studies which ask humans to improvise a continuation [3] may inform improved tasks and evaluation strategies, and underpin models to predict the musical future.

8. ACKNOWLEDGMENTS

We thank Anja Volk and James Owers for sharing their insights and comments throughout our research.

9. REFERENCES

[1] Christopher Ariza. The Interrogator as Critic: The Turing Test and the Evaluation of Generative Music Systems. *Computer Music Journal*, 33(2):48–70, 2009.

[2] Jamshed J. Bharucha. Tonality and Expectation. In Rita Aiello and John A. Sloboda, editors, *Musical Perceptions*, pages 213–239. Oxford University Press, 1994.

[3] James C. Carlsen, Pierre I. Divenyi, and Jack A. Taylor. A Preliminary Study of Perceptual Expectancy in Melodic Configurations. *Bulletin of the Council for Research in Music Education*, pages 4–12, 1970.

[4] Srikanth Cherla, Tillman Weyde, Artur Garcez, and Marcus Pearce. A Distributed Model For Multiple-Viewpoint Melodic Prediction. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 15–20, 2013.

[5] Andy Clark. Whatever Next? Predictive Brains, Situated Agents, and the Future of Cognitive Science. *The Behavioral and Brain Sciences*, 36:181–204, 2013.

[6] John G. Cleary and Ian H. Witten. Data Compression Using Adaptive Coding and Partial String Matching. *IEEE Transactions on Communcations*, COM-32(4):396–402, 1984.

[7] Tom Collins, Andreas Arzt, Sebastian Flossmann, and Gerhard Widmer. SIARCT-CFP: Improving Precision and the Discovery of Inexact Musical Patterns in Point-Set Representations. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 549–554, 2013.

[8] Tom Collins, Sebastian Böck, Florian Krebs, and Gerhard Widmer. Bridging the Audio-Symbolic Gap: The Discovery of Repeated Note Content Directly From Polyphonic Music Audio. In *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*, pages 1–12, 2014.

[9] Tom Collins and Robin Laney. Computer-generated Stylistic Compositions with Long-term Repetitive and Phrasal Structure. *Journal of Creative Music Systems*, 1(2), 2017.

[10] Florian Colombo. Generating and Discriminating Symbolic Music Continuations with Bach-Prob. `https://www.music-ir.org/mirex/wiki/2018:Patterns_for_Prediction_Results`, 2018. Accessed: 2019-04-09.

[11] Darrell Conklin and Ian H. Witten. Multiple Viewpoint Systems for Music Prediction. *Journal of New Music Research*, 24(1):51–73, 1995.

[12] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. MuseGAN: Demonstration of a Convolutional GAN Based Model for Generating Multi-Track Piano-Rolls. In *Late Breaking/Demos, 18th International Society for Music Information Retrieval Conference*, 2017.

[13] Douglas Eck and Juergen Schmidhuber. Finding Temporal Structure in Music: Blues Improvisation with LSTM Recurrent Networks. In *Proceedings of the 12th IEEE workshop on neural networks for signal processing*, pages 747–756. IEEE, 2002.

[14] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music Transformer, 2018.

[15] David Huron. *Sweet Anticipation. Music and the Psychology of Expectation*. MIT Press, Cambridge, Massachusetts, 2007.

[16] Berit Janssen, W. Bas de Haas, Anja Volk, and Peter van Kranenburg. Finding Repeated Patterns in Music: State of Knowledge, Challenges, Perspectives. In M. Aramaki, editor, *10th International Symposium, CMMR 2013, Revised Selected Papers*, number 8905, pages 277–297. 2014.

[17] Carol L. Krumhansl and Edward J. Kessler. Tracing the Dynamic Changes in Perceived Tonal Organization in a Spatial Representation of Musical Keys. *Psychological Review*, 89(4):334–368, 1982.

[18] Olivier Lartillot. In-depth Motivic Analysis Based on Multiparametric Closed Pattern and Cyclic Sequence Mining. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 361–366, 2014.

[19] Leonard C. Manzara, Ian H. Witten, and Mark James. On the Entropy of Music: An Experiment with Bach Chorale Melodies. *Leonardo Music Journal*, 2(1):81–88, 1992.

[20] Gabriele Medeot, Srikanth Cherla, Katerina Kosta, Matt McVicar, Samer Abdallah, Marco Selvi, Ed Newton-Rex, and Kevin Webster. StructureNet: Inducing Structure in Generated Melodies. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 725–731, 2018.

[21] David Meredith. COSIATEC and SIATECompress: Pattern Discovery by Geometric Compression. `https://www.music-ir.org/mirex/wiki/2013:Discovery_of_Repeated_Themes_\%26_Sections_Results`, 2013. Accessed: 2018-02-09.

[22] David Meredith, Kjell Lemström, and Geraint A. Wiggins. Algorithms for Discovering Repeated Patterns in

Multidimensional Representations of Polyphonic Music. *Journal of New Music Research*, 31(4):321–345, 2002.

[23] Michael C. Mozer. Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multi-Scale Processing. *Connection Science*, 6(2-3):247–280, 1994.

[24] Eric Nichols. Seq2SeqP4P: A Sequence-to-Sequence model for Monophonic Music Continuation. `https://www.music-ir.org/mirex/wiki/2018:Patterns_for_Prediction_Results`, 2018. Accessed: 2019-04-09.

[25] Bryan Pardo and William P. Birmingham. Algorithms for chordal analysis. *Computer Music Journal*, pages 27–49, 2002.

[26] Marcus Pearce and Geraint A. Wiggins. Improved Methods for Statistical Modelling of Monophonic Music. *Journal of New Music Research*, 33(4):367–385, dec 2004.

[27] Marcus T. Pearce, Daniel Müllensiefen, and Geraint A. Wiggins. The Role of Expectation and Probabilistic Learning in Auditory Boundary Perception: A Model Comparison. *Perception*, 39(10):1367–1391, 2010.

[28] Matevž Pesek, Aleš Leonardis, and Matija Marolt. A Compositional Hierarchical Model for Music Information Retrieval. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 131–136, 2014.

[29] Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-Midi Alignment and Matching*. PhD thesis, Columbia University, 2016.

[30] Iris Yuping Ren. Closed Patterns in Folk Music and Other Genres. In *Proceedings of the 6th International Workshop on Folk Music Analysis*, 2016.

[31] E. Glenn Schellenberg. Expectancy in Melody: Tests of the Implication-Realization Model. *Cognition*, 58:75–125, 1996.

[32] Bob L. Sturm and Oded Ben-Tal. Taking the Models back to Music Practice: Evaluating Generative Transcription Models Built Using Deep Learning. *Journal of Creative Music Systems*, 2, 2017.

[33] Esko Ukkonen, Kjell Lemström, and Veli Mäkinen. Geometric Algorithms for Transposition Invariant Content-Based Music Retrieval. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 193–199, 2003.

[34] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. In *9th ISCA Speech Synthesis Workshop*, pages 125–125, 2016.

[35] Gissel Velarde, Tillman Weyde, and David Meredith. An Approach to Melodic Segmentation and Classification Based on Filtering with the Haar-Wavelet. *Journal of New Music Research*, 42(4):325–345, 2013.

[36] Gerhard Widmer. Getting Closer to the Essence of Music: The Con Espressione Manifesto. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):19, 2017.

LEARNING SOFT-ATTENTION MODELS FOR TEMPO-INVARIANT AUDIO-SHEET MUSIC RETRIEVAL

Stefan Balke[1] **Matthias Dorfer**[1] **Luis Carvalho**[1] **Andreas Arzt**[1] **Gerhard Widmer**[1,2]
[1] Institute of Computational Perception, Johannes Kepler University Linz, Austria
[2] The Austrian Research Institute for Artificial Intelligence (OFAI), Austria
stefan.balke@jku.at

ABSTRACT

Connecting large libraries of digitized audio recordings to their corresponding sheet music images has long been a motivation for researchers to develop new cross-modal retrieval systems. In recent years, retrieval systems based on embedding space learning with deep neural networks got a step closer to fulfilling this vision. However, global and local tempo deviations in the music recordings still require careful tuning of the amount of temporal context given to the system. In this paper, we address this problem by introducing an additional soft-attention mechanism on the audio input. Quantitative and qualitative results on synthesized piano data indicate that this attention increases the robustness of the retrieval system by focusing on different parts of the input representation based on the tempo of the audio. Encouraged by these results, we argue for the potential of attention models as a very general tool for many MIR tasks.

1. INTRODUCTION

Algorithms for content-based search and retrieval play an important role in many applications that are based on large collections of music data. In this paper, we re-visit a challenging cross-modal retrieval problem, namely audio–sheet music retrieval: given a short audio excerpt, we are trying to retrieve the corresponding score from a database of sheet music (stored as images).

Traditional methods for audio–sheet retrieval are based on common mid-level representations that allow for an easy comparison of time points in the audio and positions in the sheet music, see for instance [2, 14, 17, 21]. Typical examples are spectral features like pitch class profiles (chroma-features) or symbolic representations. Unfortunately, deriving such mid-level representations is an error-prone process, as this may involve preprocessing steps such as music transcription [6, 9, 16, 18, 19, 26], Optical Music Recognition [7, 15, 24, 25, 28], and sometimes both. For a recent overview of different cross-modal retrieval approaches, see [22].

In [11], an alternative approach has been proposed that circumvents these problematic preprocessing steps by learning embeddings directly from the multi-modal data (see Figure 1 for a schematic overview). Given short snippets of audio and their respective excerpts of sheet music images, a cross-modal neural network is trained to learn an embedding space in which both modalities are represented as 32-dimensional vectors (Figure 1a). The vectors can then be easily compared in the embedding space by means of a distance function, e. g., the cosine distance (Figure 1b). The retrieval results are then selected by sorting the candidates by the obtained distances (Figure 1c). A conceptually similar retrieval approach for videos was presented in [1].

In essence, the neural network replaces the step of extracting mid-level features by directly learning a transformation from the audio and from the sheet music image data to a common vector space. A limitation of this approach is that the temporal context (or field of view) of the data in both modalities is of fixed size. For audio data, this implies that the actual musical content of the window depends on the tempo a piece is played in. If it is played in a fast tempo, the audio excerpt will contain a larger amount of musical content (i. e., more notes) than if the same piece is played slowly. Obviously, this can lead to large discrepancies between what the model has seen during training, and the data it will see during test time.

A possible solution to this problem is to let the network decide the appropriate temporal context for a given audio query by using a separate attention mechanism [3,8,23,27]. In a recent workshop paper [12], the concept of using a soft-attention mechanism in a cross-modal retrieval scenario was presented. This allowed the network to deal with a much larger temporal context at the input than without attention. In this paper, we substantially extend this idea with systematic and quantitative experiments with respect to tempo robustness, as well as giving further details about the used architecture and the training data. The remainder of this paper is structured as follows. In Section 2, we introduce the retrieval task and explain the necessary steps to approach it with end-to-end cross-modal embedding learning. Section 3 reports on systematic experiments that show the benefits of using a dedicated attention network on audio spectrograms, to improve retrieval results compared to

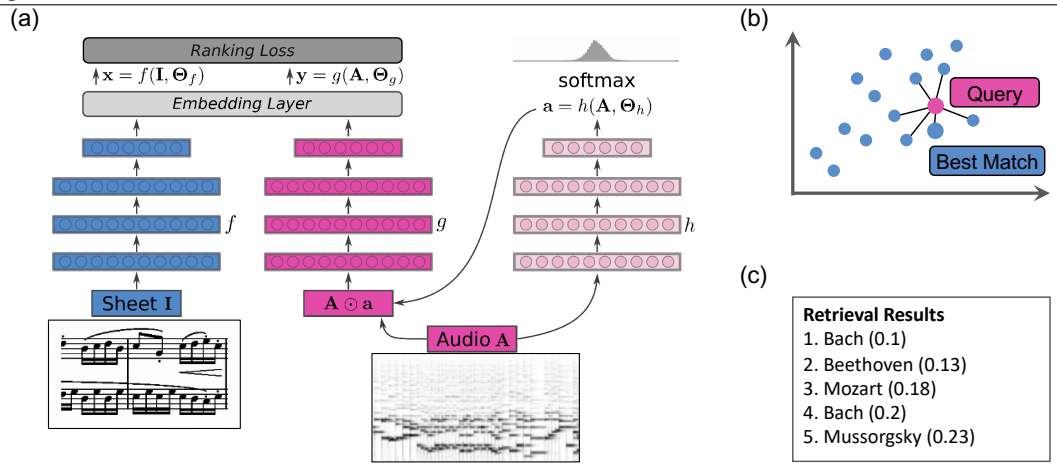


Figure 1: Illustration of the cross-modal retrieval application. (a) Audio snippet serves as input query to the embedding network. The attention network selects the "relevant" region of the audio spectrogram. (b) 2-dimensional visualization of the embedding space. The nearest neighbors are selected as candidates. (c) Ranked list of candidates sorted by their distances to the query embedding.

the current state of the art. Furthermore, we present qualitative examples to highlight the intuitive behavior of the attention network and conclude the paper in Section 4.

2. AUDIO–SHEET RETRIEVAL

We consider a cross-modal retrieval scenario (Figure 1): given an audio excerpt as a search query, we wish to retrieve the corresponding snippet of sheet music of the respective piece. We approach this retrieval problem by learning a low-dimensional multimodal embedding space (32 dimensions) for both snippets of sheet music and excerpts of music audio (Figure 1a). We desire for each modality a projection into a shared space where semantically similar items of the two modalities are projected close together, and dissimilar items far apart (Figure 1b). Once the inputs are embedded in such a space, cross-modal retrieval is performed using simple distance measures and nearest-neighbor search. Finally, the retrieval results are obtained by means of a ranked list through sorting the distances in an ascending order (Figure 1c).

The embedding space is trained with convolutional neural networks (CNN). Figure 1a sketches the network architecture. The baseline model (without attention) consists of two convolutional pathways: one is responsible for embedding the sheet music, and the other for embedding the audio excerpt. The key part of the network is the canonically correlated (CCA) embedding layer [13]. This layer forces the two pathways to learn representations that can be projected into a shared space. The desired properties of this multimodal embedding space are enforced by training with pairwise ranking loss (also known as contrastive loss) [20]. This is the basic structure of the model recently described and evaluated in [11]. This attention-less model serves as a baseline in our experiments, i. e., the input audio (or sheet music) has to be sliced into excerpts of a given size (e. g., 2 s). However, when processing performed music, the temporal context captured by the fixed-size input excerpts varies, based on the current tempo of the piece.

Audio (Spectrogram) $92 \times \{42, 84, 168\}$	**Sheet-Image** 160×200
2x Conv(3, pad-1)-24 - BN MaxPooling(2) 2x Conv(3, pad-1)-48 - BN MaxPooling(2) 2x Conv(3, pad-1)-96 - BN MaxPooling(2) 2x Conv(3, pad-1)-96 - BN MaxPooling(2) Conv(1, pad-0)-32 - Linear GlobalAvgPooling + Softmax Mult(Spectrogram, Attention)	Attention Network
2x Conv(3, pad-1)-24 - BN MaxPooling(2) 2x Conv(3, pad-1)-48 - BN MaxPooling(2) 2x Conv(3, pad-1)-96 - BN MaxPooling(2) 2x Conv(3, pad-1)-96 - BN MaxPooling(2) Conv(1, pad-0)-32 - Linear - BN Dense(32) + Linear	2x Conv(3, pad-1)-24 - BN MaxPooling(2) 2x Conv(3, pad-1)-48 - BN MaxPooling(2) 2x Conv(3, pad-1)-96 - BN MaxPooling(2) 2x Conv(3, pad-1)-96 - BN MaxPooling(2) Conv(1, pad-0)-32 - Linear - BN Dense(32) + Linear
Embedding Layer + Ranking Loss	

Table 1: Overview of the network architecture. The upper part describes the attention network, the lower part the embedding part. Conv(3, pad-1)-24: 3×3 convolution, 24 feature maps and zero-padding of 1. We use ELU activation functions on all layers if not stated otherwise [10].

This attention-less model trains and operates on fixed-size input windows from both modalities. In other words, the musical content provided to the CNN may contain significantly less or more information—especially note onsets—than excerpts it has been trained on. One may of course compensate this with data augmentation, but a more general solution is to simply let the model decide how much information is needed from the audio.

For this purpose, we explore a soft-attention mecha-

nism. First, we substantially increase the audio field of view (number of spectrogram frames), up to a factor of four. Next, we add to the model the attention pathway h, which should learn to restrict the audio input again by focusing only at those parts that appear relevant for an efficient search query. As detailed in Table 1, this attention mechanism is implemented as a separate CNN. The output of this CNN is a probability density function a which has the same number of frames as the audio spectrogram. Before feeding the spectrogram to the audio embedding network g, we multiply each frame with its attention weight. This enables the model to cancel out irrelevant parts of the query and focus on the important information. In the following experiments, we show that adding this attention network in combination with the longer temporal context, substantially improves the results in the considered audio–sheet retrieval task.

3. EXPERIMENTS

This section reports on the conducted retrieval experiments. We start by describing the data used for training and testing the models, and the data augmentation steps applied during training. Afterwards, we present the results for the two main experiments, both dealing with audio–sheet retrieval: given an audio excerpt, retrieve the corresponding snippet of sheet music of the respective piece. Finally, we look at the attention-layer's behavior for five examples and discuss benefits and limitations of the approach.

3.1 Data Description and Training

We use a dataset of synthesized classical piano music, called *MSMD* [11]. In version 1.1, MSMD comprises 467 pieces by 53 composers, including Bach, Mozart, Beethoven and Chopin, totalling in over a thousand pages of sheet music and 15+ hours of audio, with fine-grained cross-modal alignment between note onsets and noteheads. The main changes from version 1.0 (as used in [11]) to version 1.1 are that we cleaned the test set from broken pieces and set all note velocities to a value of 64. The scores and audio are both synthesized using the score engraving software LilyPond [1] and FluidSynth. [2]

During training, we augment the sheet music by resizing and shifting the image. For augmenting the audio, we vary the tempo between 95 and 110 % and sample from a pool of three different piano soundfonts. For details of these augmentation steps, we kindly refer the reader to the explanation given in [11]. After training convergence, we refine the used CCA embedding layer on the whole training set. The reason for this is that during training, the covariance estimates used in the CCA projection are only based on the number of samples contained in the mini-batch.

For testing, we select 10,000 audio–sheet music pairs from an unseen testset, where the synthesized audio is rendered with a separate, hold-out piano soundfont. The rendering procedure for the sheet music remains the same.

[1] http://www.lilypond.org
[2] http://www.fluidsynth.org

Model	R@1	R@5	R@25	MRR	MR
BL1-SC [11]	13.67	34.62	57.44	0.24	15
BL2-SC	34.25	54.68	70.62	0.44	4
BL2-MC	36.59	60.52	77.21	0.48	3
BL2-LC	27.66	51.79	70.34	0.39	5
BL2-SC + AT	38.41	59.95	74.36	0.48	3
BL2-MC + AT	46.54	68.45	81.10	0.57	2
BL2-LC + AT	53.16	74.42	85.35	0.63	1

(a) Un-refined CCA layer.

Model	R@1	R@5	R@25	MRR	MR
BL1-SC [11]	19.12	44.16	66.63	0.31	8
BL2-SC	48.91	67.22	78.27	0.57	2
BL2-MC	47.08	68.19	80.82	0.57	2
BL2-LC	43.46	68.38	82.84	0.55	2
BL2-SC + AT	55.43	72.64	81.05	0.63	1
BL2-MC + AT	58.14	76.50	84.60	0.67	1
BL2-LC + AT	66.71	84.43	91.19	0.75	1

(b) Refined CCA layer.

Table 2: Overview of the experimental results. (a) Lists the results without and (b) with a refinement of the CCA layer. All experiments used 10,000 candidates and were conducted on MSMD-v1.1 (R@k = Recall@k, MRR = Mean Reciprocal Rank, MR = Median Rank).

The above described augmentation steps are disabled during testing. In terms of performance, the refinement step of the CCA layer yields an increase in performance of around 0.07 up to 0.15 in terms of mean reciprocal rank (MRR) on the test set (see Table 2 for details). The used dataset , [3] as well as the implementation along with trained models , [4] are publicly available.

3.2 Experiment 1: Attention

In the first experiment, we investigate the influence of the additional attention network. We systematically increase the temporal context of the audio representation from a short context (SC, 42 frames = 2.1 s), over a medium (MC, 84 frames = 4.2 s), to a long context (LC, 168 frames = 8.4 s). The results are summarized in Table 2. As evaluation metrics we use different Recalls (R@1, R@5, R@25 $\in [0, 100]$, higher is better), the mean reciprocal rank (MRR $\in [0, 1]$, higher is better), as well as the median rank (MR $\in [1, 10000]$, lower is better). In the following discussion, we focus on the results of the refined CCA layer, as given in Table 2b.

As a baseline (BL1-SC), we use the network architecture as described in [11], which uses the short context (SC). For this model, approximately 20% of the queries are on rank 1 (R@1 = 19.12) and in almost 70% of the audio

[3] https://github.com/CPJKU/msmd/tree/v1.1
[4] https://github.com/CPJKU/audio_sheet_retrieval/tree/ismir-2019

Model	$\rho = 0.5$	$\rho = 0.66$	$\rho = 1$	$\rho = 1.33$	$\rho = 2$
BL1-SC [11]	0.20	0.27	0.31	0.30	0.22
BL2-SC	0.44	0.52	0.57	0.56	0.46
BL2-MC	0.46	0.53	0.57	0.55	0.43
BL2-LC	0.44	0.50	0.55	0.51	0.35
BL2-SC + AT	0.55	0.63	0.63	0.64	0.56
BL2-MC + AT	0.54	0.61	0.67	0.67	0.62
BL2-LC + AT	0.64	0.69	0.75	0.73	0.64

Table 3: Mean Reciprocal Rank (MRR) for different models and different tempo ratios $\rho \in \{0.5, 0.66, 1, 1.33, 2\}$. For example, $\rho = 0.5$ stands for half of the original tempo and $\rho = 2$ for doubling the tempo. The listed models correspond to the refined CCA models as listed in Table 2b with the same test set size of 10,000 candidates.

queries, the relevant sheet image is within the first 25 ranks (R@25 = 66.63). As a second baseline (BL2-SC), we slightly adapt the original architecture by exchanging the global average pooling layer (before the embedding layer) by a dense layer for each modality (see the non-attention part of Table 1 for details). With this adaptation, the results improve significantly to R@1 = 48.91, R@25 = 78.27, and a median rank MR = 2, instead of MR = 8 for BL1.

By increasing the temporal context on BL2 to medium sized context (BL2-MC), mean reciprocal rank (MRR = 0.57) and median rank (MR = 2) stay unchanged. When increasing the temporal context to the long context (BL2-LC), the model degrades in performance, e. g., R@1 drops from 48.91% for SC to 43.46% for LC and the MRR from 0.57 to 0.55. Adding the attention network to the audio input (BL2-SC + AT) improves the results by 7, 5, and 3% for the recalls, as well as 0.05 for the MRR compared to BL2-SC. The more context is given to the network, the better the performance metrics get, e. g., R@1 improves from 58.14 (BL2-MC + AT) to 66.71 (BL2-LC + AT). The MRR, improves from 0.63 (BL2-SC + AT), over 0.67 (BL2-MC + AT), up to 0.75 (BL2-LC + AT).

We derive two main observations from these results. First, optimizing the network architecture is important; dropping the global average pooling in favour of a fully-connected dense layer lifted the results to another level. The reason could be that the fully-connected layer better retains the structure of the input spectrogram than the average pooling and in addition can be more selective on relevant input parts, e. g., by setting weights to zero. Second, the attention network enables the network to deal with larger temporal context sizes. From a signal processing perspective, one would expect that more context (and thus longer queries) would always help since it increases the specificity of the query. However, since we squash this information into a 32-dimensional embedding vector, it seems that too much information (e. g., too many onsets in the audio), actually harms the retrieval quality when not using attention.

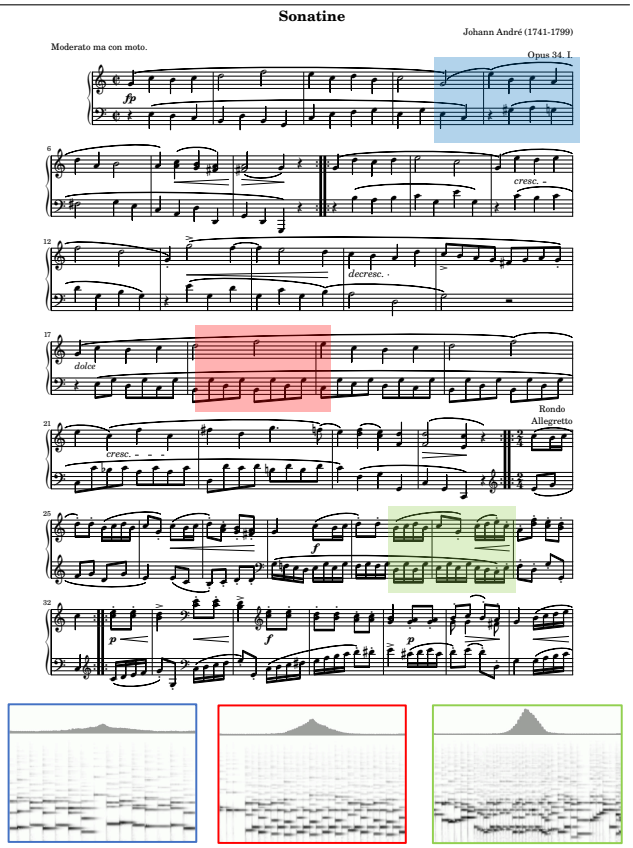


Figure 2: Example of tempo variations within a piece by Johann André - Sonatine (Op. 34, I.). The three boxes below the sheet music show the attention output, and the corresponding audio spectrogram for the respective excerpt.

3.3 Experiment 2: Tempo Robustness

In a second experiment, we explicitly test the system's robustness to global tempo changes. For this purpose, we re-rendered the MSMD test dataset with various tempo ratios $\rho \in \{0.5, 0.66, 1, 1.33, 2\}$, where $\rho = 0.5$ stands for halving and $\rho = 2$ for doubling the original tempo. The results are given in Table 3. For the sake of brevity, we only show the mean reciprocal rank (MRR) for the refined CCA models.

In general, we observe a similar trend as in the first experiment. While the original baseline approach (BL1-SC) performs rather poorly (0.20 to 0.31), exchanging the global average pooling layer (BL2-SC) helps to improve the performance (0.44 to 0.57). The best attention model (BL2-LC + AT) yields values ranging from 0.64 to 0.75. In this experiment, we would have expected that the improvement holds true for all testsets rendered at different global tempi. However, the numbers tell a slightly different story. To understand this, we had to go back to the generation of the MSMD dataset. Recall that the dataset is generated from LilyPond files obtained from the Mutopia Project. [5] These files contain tempo specifications which are retained in the synthesis pipeline. The specified tempi vary in a range between 30 and 182 bpm (mean tempo = 106 bpm,

[5] https://www.mutopiaproject.org

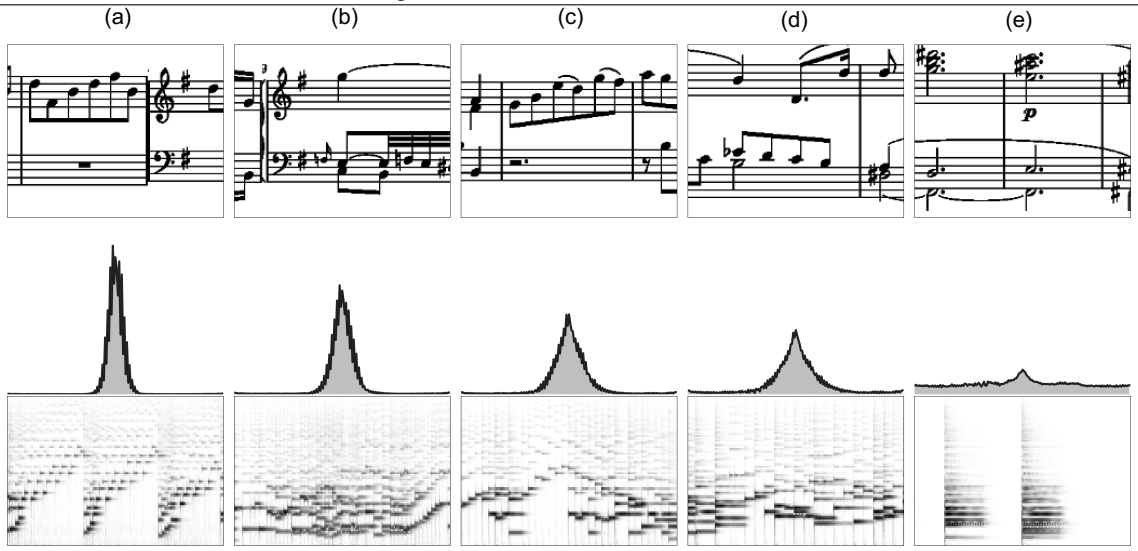


Figure 3: Illustration of the sheet music, input attention (normalized), and spectrogram for five examples from the MSMD test set. (a) L. v. Beethoven - Piano Sonata (Op. 79, 1st Mvt.), (b) J. S. Bach - Goldberg Variations: Variatio 12 (BWV 988), (c) J. S. Bach - French Suite VI: Menuet (BWV 817), (d) R. Schumann - Album for the Youth: Untitled (Op. 68, Nr. 26), and (e) M. Mussorgsky - Pictures at an Exhibition VIII: Catacombae.

std. dev. = 25 bpm). This distribution of tempi implies that all experiments we perform on the original MSMD testset ($\rho = 1$) already test for a variety of tempi. Synthesizing this dataset with different tempo factors—as done in our second experiment—shifts and stretches this distribution. In the edge cases ($\rho = 0.5$ and $\rho = 2$), this leads to unrealistic tempi, e. g., 15 bpm or 364 bpm, producing input audio windows with absurdly low or high onset density.

In summary, all of the tested models have in common that they work best for the original testset tempo ($\rho = 1$) and have similar relative performance drops for the tempo variations. However, the attention model is able to keep the retrieval results at a much higher level than all the other models. In the following section, we take a closer look at some examples from the MSMD test set to get a better intuition of the soft-attention mechanism and how it reacts to tempo deviations.

3.4 Examples

In the experiments above, we have seen that the attention models improve the retrieval results by a considerable margin. Another positive aspect of the soft-attention mechanism is that its behavior is directly interpretable—a rare case in the deep learning landscape. For all the presented examples, we provide further videos with audio on an accompanying website, along with detailed instructions for reproduction. [6]

Figure 2 shows the sheet music for Johann André's Sonatine, Op.34. The piece starts with a calm part with mainly legato quarter notes at 103 bpm. The first (blue) box shows the corresponding attention output and the audio spectrogram for measure five. The attention output is relatively flat with a small peak in the middle of the audio excerpt. In bar 17, the note density changes, with eighth notes in the left hand entering the scene. As shown by the second (red) box, the attention layer reacts to this by building up a more distinct focus around the center, placing less weight on the outer parts of the spectrogram. In the third part beginning with measure 24, the "tempo" (more precisely: the perceived "speed", in terms of events per time unit) essentially doubles, which is reflected in a high peak and narrow distribution in the attention output (green right box). From these three examples, it also becomes clear that tempo and note density are essentially two sides of the same coin for the attention network.

In Figure 3, we show examples from the testset with different global performance tempi, along with the sheet music excerpt, the attention weights, and the spectrogram. The fastest piece, with around 250 bpm, is shown in Figure 3a. The corresponding attention output is very focused on the middle part of the audio spectrogram, trying to concentrate the attention to the notes that actually appear in the sheet music snippet. The second example (b) is rather slow with 95 bpm. However, through the use of 16th and 32th notes, the main melody gets a double-time feel, thus the actual tempo is perceived at around 190 bpm. In Figure 3c, the tempo is at 115 bpm. The attention output starts to widen up, allowing more temporal context to reach the actual embedding network. This trend goes in Figure 3d when the tempo 78 bpm. Figure 3e shows an extreme example where the piece mainly consists of chords with long, dotted half notes. Here, the attention has to use the complete temporal context of the audio spectrogram to match the score information with the audio.

The examples demonstrate that depending on the spec-

[6] http://www.cp.jku.at/resources/2019_ASR-TempoInv_ISMIR

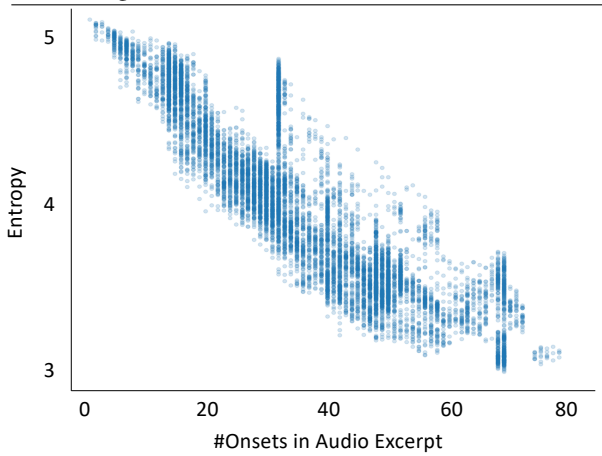


Figure 4: Entropy of the input attention vs. the number of onsets in the respective audio frame.

trogram content, the model indeed attends to whatever it believes is a representative counterpart of the target sheet music snippet. Since the fixed-size sheet snippets contain roughly similar numbers of notes, as the density of note heads on the printed page tends to be independent of the tempo of the piece, attention is sharply peaked when the density of onsets in the audio is high, and conversely is distributed more evenly when there are fewer notes in the audio excerpt.

4. SUMMARY

In this paper, we have described a soft-attention mechanism that helps to overcome the fixed window sizes uses in Convolutional Neural Networks. In our end-to-end audio–sheet music retrieval application, the results improved substantially compared to the state of the art. By looking at a number of examples from the retrieval results, the soft-attention mechanism showed an intuitive and interpretable behavior.

This appealing and intuitive behavior is summarized in Figure 4, which shows the entropy of the attention distribution in relation to the number of onsets contained in the audio excerpt, for all 10,000 test samples. The entropy is a measure of flatness of the attention output: a flat function gives high entropy, a very narrow function low entropy. The downward trend in the figure confirms our observations from above: the more onsets in the respective audio spectrogram, the narrower the attention distribution.

Given the improved retrieval performance and the intuitive behavior of the attention model, we think this is a promising line of research for reducing the sensitivity of cross-modal music retrieval models to the audio input window size. To this end, our experiments were conducted on synthesized piano recordings. However, results in [11] indicate that the embedding models trained on this data generalize to real scores and performances. A possible next step would be to investigate whether the attention mechanism reacts to local tempo changes as occuring frequently in real performances (e. g., ritardandi and accelerandi). Furthermore, it would be interesting to leave the piano music domain and extend the model to cope with differences in timbre, (e. g., orchestral music) as done in [4] for the challenging Barlow–Morgenstern scenario [5].

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement number 670035, project CON ESPRESSIONE, and the Marie Skłodowsa-Curie grant agreement number 765068, MIP-Frontiers).

5. REFERENCES

[1] Relja Arandjelovic and Andrew Zisserman. Look, listen and learn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 609–617, Venice, Italy, 2017.

[2] Andreas Arzt, Sebastian Böck, and Gerhard Widmer. Fast identification of piece and score position via symbolic fingerprinting. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 433–438, Porto, Portugal, 2012.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, pages 1–15, San Diego, CA, USA, 2015.

[4] Stefan Balke, Vlora Arifi-Müller, Lukas Lamprecht, and Meinard Müller. Retrieving audio recordings using musical themes. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 281–285, Shanghai, China, 2016.

[5] Harold Barlow and Sam Morgenstern. *A Dictionary of Musical Themes*. Crown Publishers, Inc., revised edition edition, 1975.

[6] Sebastian Böck and Markus Schedl. Polyphonic piano note transcription with recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 121–124, Kyoto, Japan, March 2012.

[7] Donald Byrd and Jakob G. Simonsen. Towards a standard testbed for optical music recognition: Definitions, metrics, and page images. *Journal of New Music Research*, 44(3):169–195, 2015.

[8] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964, Shanghai, China, 2016.

[9] Tian Cheng, Matthias Mauch, Emmanouil Benetos, and Simon Dixon. An attack/decay model for piano transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 584–590, New York City, United States, 2016.

[10] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *Proceedings of the International Conference on Learning Representations (ICLR) (arXiv:1511.07289)*, 2016.

[11] Matthias Dorfer, Jan Hajič jr., Andreas Arzt, Harald Frostel, and Gerhard Widmer. Learning audio–sheet music correspondences for cross-modal retrieval and piece identification. *Transactions of the International Society for Music Information Retrieval*, 1(1), 2018.

[12] Matthias Dorfer, Jan Hajič jr., and Gerhard Widmer. Attention as a perspective for learning tempo-invariant audio queries. In *Proceedings of the ICML Joint Workshop on Machine Learning for Music*, Stockholm, Sweden, 2018.

[13] Matthias Dorfer, Jan Schlüter, Andreu Vall, Filip Korzeniowski, and Gerhard Widmer. End-to-end cross-modality retrieval with CCA projections and pairwise ranking loss. *International Journal of Multimedia Information Retrieval*, 7(2):117–128, Jun 2018.

[14] Christian Fremerey, Michael Clausen, Sebastian Ewert, and Meinard Müller. Sheet music-audio identification. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 645–650, Kobe, Japan, October 2009.

[15] Jan Hajič jr., Jiri Novotný, Pavel Pecina, and Jaroslav Pokorný. Further steps towards a standard testbed for Optical Music Recognition. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 157–163, New York City, United States, 2016.

[16] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 50–57, Paris, France, 2018.

[17] Özgür Izmirli and Gyanendra Sharma. Bridging printed music and audio through alignment using a mid-level score representation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 61–66, Porto, Portugal, 2012.

[18] Rainer Kelz, Sebastian Böck, and Gerhard Widmer. Deep polyphonic ADSR piano note transcription. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, United Kingdom, May 2019.

[19] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 475–481, New York City, USA, 2016.

[20] Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint (arXiv:1411.2539)*, 2014.

[21] Frank Kurth, Meinard Müller, Christian Fremerey, Yoon ha Chang, and Michael Clausen. Automated synchronization of scanned sheet music with audio recordings. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 261–266, Vienna, Austria, September 2007.

[22] Meinard Müller, Andreas Arzt, Stefan Balke, Matthias Dorfer, and Gerhard Widmer. Cross-modal music retrieval and applications: An overview of key methodologies. *IEEE Signal Processing Magazine*, 36(1):52–62, 2019.

[23] Chris Olah and Shan Carter. Attention and augmented recurrent neural networks. *Distill*, 2016.

[24] Christopher Raphael and Jingya Wang. New approaches to optical music recognition. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 305–310, Miami, Florida, USA, 2011.

[25] Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre R. S. Marcal, Carlos Guedes, and Jaime S. Cardoso. Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, 1(3):173–190, 2012.

[26] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):927–939, 2016.

[27] Carl Southall, Ryan Stables, and Jason Hockman. Automatic drum transcription for polyphonic recordings using soft attention mechanisms and convolutional neural networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 606–612, Suzhou, China, 2017.

[28] Cuihong Wen, Ana Rebelo, Jing Zhang, and Jaime S. Cardoso. A new Optical Music Recognition system based on combined neural network. *Pattern Recognition Letters*, 58:1–7, 2015.

CONTRIBUTING TO NEW MUSICOLOGICAL THEORIES WITH COMPUTATIONAL METHODS: THE CASE OF CENTONIZATION IN ARAB-ANDALUSIAN MUSIC

Thomas Nuttall **Miguel García Casado** **Víctor Núñez Tarifa**
Rafael Caro Repetto **Xavier Serra**
Music Technology Group, Universitat Pompeu Fabra, Spain
thomas.nuttall01@estudiant.upf.edu, miguel.garciac@upf.edu

ABSTRACT

Arab-Andalusian music was formed in the medieval Islamic territories of the Iberian Peninsula, drawing on local traditions and assuming Arabic influences. The expert performer and researcher of the Moroccan tradition of this music, Amin Chaachoo, is developing a theory whose last formulation was recently published in *La Musique Hispano-Arabe, al-Ala* (2016), which argues that centonization, a melodic composition technique used in Gregorian chant, was also utilized for the creation of this repertoire.

In this paper we aim to contribute to Chaachoo's theory by means of tf-idf analysis. A high-order n-gram model is applied to a corpus of 149 prescriptive transcriptions of heterophonic recordings, representing each as an unordered multiset of patterns. Computing the tf-idf statistic of each pattern in this corpus provides a means by which we can rank and compare motivic content across *nawabāt*, distinct musical forms of the tradition. For each *nawba*, an empirical comparison is made between patterns identified as significant via our approach and those proposed by Chaachoo. Ultimately we observe considerable agreement between the two pattern sets and go further in proposing new, unique and as yet undocumented patterns that occur at least as frequently and with at least as much importance as those in Chaachoo's proposals.

1. INTRODUCTION

The rich culture developed in the medieval Islamic territories of the Iberian Peninsula known as Al-Andalus gave birth to a refined musical and literary tradition that combines local musical practices with Middle Eastern Arabic poetry and sensibilities. The core of this tradition is the singing of *ṣanā'i'* (plural of *ṣan'a*) or poems either by a choir accompanied by an instrumental ensemble or by a soloist. These *ṣanā'i'* are performed in suites known as *nawabāt* (plural of *nawba*), which also include orchestral pieces and both instrumental and vocal solo improvisations [6, 13]. The migration of Andalusian population to North Africa brought this tradition to this region, were it survived to this date after the disappearance of Al-Andalus in the 15th century. Nowadays, it is considered the classical musical repertoire in countries such as Morocco, Algeria, and Tunisia, in each of which it developed local characteristics, and is commonly known (among other names [22]) as Arab-Andalusian music. In this paper, we focus on the Moroccan repertoire of this tradition, which is known as *al-Āla* [6].

1.1 Music Theory and Centonization

Nawba is the essential form of Arab-Andalusian music. All the *ṣanā'i'* and other pieces in one *nawba* are composed in one single mode, known in this tradition as *ṭāb'* (plural *ṭūbu'*). In the specific case of the Moroccan *al-Āla* repertoire, pieces from certain *nawabāt* were lost during the process of oral transmission, so that the surviving ones where attached to other *nawabāt* according to modal similarity. In the 18th century, the scholar al-Haiek fixed the number of *nawabāt* in the *al-Āla* tradition to eleven (Table 1) [6].

The scholar and expert performer of *al-Āla* Amin Chaachoo is researching and developing a theoretical framework for this tradition. Chaachoo argues that regarding its musical aspect, Arab-Andalusian music heavily draws on local Iberian practices, and especially on plainchant in terms of compositional principles. A main argument for this proposal is the nature of Arab-Andalusian *ṭūbu'*, which lack the microtonalities and nuances of Arabic *maqam*. Chaachoo, in his publication *La música andalusí al-Ála* [5] characterizes each *ṭāb'* with a particular ascending and descending scale, a fundamental degree similar to the *finalis* of Gregorian modes, and one or two dominant degrees. In his more recent *La Musique Hispano-Arabe, al-Ala* [6], he also proposes the concept of a "persistent degree," inspired in the reciting tone from plainchant.

One of the most original proposals by Chaachoo is the use of centonization as the basic technique for melodic creation in Arab-Andalusian music. Centonization, from latin *cento* meaning patchwork, is defined by Paolo Ferretti as

melodic composition by synthesis of pre-existing musical units known as *centos* [11], and is generally associated to melodic creation in Gregorian chant [1, 7]. Chaachoo argues that the melodic material in Arab-Andalusian music is also created by combination of *centos*, thus strengthening the connection between this tradition and Iberian local practices. The author associates these melodic units or *centos* to specific *ṭūbu'* (See Fig.1), so that one particular *ṭāb'* is characterized by a set of *centos*. To the best of our knowledge, Chaachoo's is the first attempt to explain melodic creation of Arab-Andalusian music through the concept of centonization.

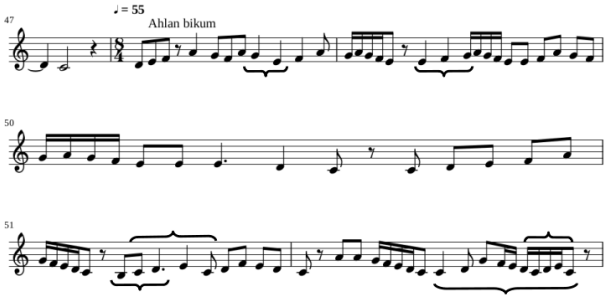


Figure 1: Score example with the characteristic *centos* of the piece *Btayhi Rasd Dayl* of Nawba 4

1.2 Motivation

The use of centonization as composition technique in Arab-Andalusian music is still being developed by Chaachoo. Already published in two recent works [5,6], the list of *centos* per *ṭāb'* slightly varies from its first formulation to the second, as the theory is being consolidated. Chaachoo draws on his lifetime experience as performer and instructor of this music, as well as in his analytic work as a musicologist, for the definition of each *ṭāb'* list of characteristic *centos* [1] . Therefore, our main goal is to contribute to the development of Chaachoo's theoretical work with findings from computational analysis of melodic patterns in a dataset of machine readable music scores.

Preliminary experiments with this dataset shows that, if we were to try and identify the *nawba* to which a specific score belongs by simply counting the occurrences of *centos* unique to the *ṭūbu'* that constitute it as specified by Chaachoo, we would be unsuccessful. In fact, attempting to do so on the corpus used in this study results in a misclassification rate of 80%. This is due to certain *nawabāt* relying heavily on very general melodic sequences located around the tonic and the dominant of the modes, these general sequences can be found in many of the *centos* documented by Chaachoo as being specific to a certain *ṭāb'* (and therefore *nawba*).

Our aim is to quantify the importance of particular melodic patterns with respect to *nawabāt* and provide an empirical ranking of melodic content for each. We must therefore rely on an approach that considers more than just frequency of occurrence, one that normalises for generality, putting forward new proposals, as to the melodic constitution of each *nawba* [6].

[1] Personal communication

2. RELATED WORK

There exists many studies into melodic pattern recognition, summaries of which have been made by Jansen et al. [15] and more recently Ren et al. [24]. Lack of agreement on the current state-of-the-art stems from the difficulty in evaluating approaches, with expertly annotated ground truth often required for performance measurement, more often than not on a study-by-study basis.

In no deviation from this trend we draw on the work of Chaachoo [6] for validation of our results and go further in suggesting/supplementing his studies with unique insights of our own. Accordingly, we aim to approach this investigation with interpretability in mind, seeking to build upon the string-based, frequency approaches found in [8,14,16], where patterns are represented by counting the number of instances of re-occurring sub-sections of notes in a musical sequence and their significance computed by comparing these counts, ignoring potential *interaction* between non-consecutive notes. The appeal of this method is that the theory is intuitive to a non-specialist and aligns with what a musician might consider important when characterising a musical piece melodically, an important consideration when wishing to contribute to and communicate with Chaachoo and his works.

The choice to consider only consecutive notes as belonging to the same pattern is supported by the nature of the music scores in our dataset (see section 3.1). These scores are manual transcriptions by Chaachoo himself from a collection of representative recordings. Due to the heterophonic nature of this music tradition, and the analytical purpose of the music scores, they contain only the common melodic line underlying the actual rendition of the choir and instruments in the orchestra. This rather prescriptive character of the transcriptions results in a representation of the music more theoretical than fine detailed, thus permitting the assumption that *centos* are literally repeated in the score, without the modifications of the ornamentation in actual performance.

As mentioned in the previous section however, simple frequency alone is not sufficiently powerful in characterising *nawba*. A more desirable attitude would be probabilistic, as found with Conklin in [9]. Conklin puts forth a novel method of computing pattern significance by comparing the probability of occurrence in a corpus with the probability of occurrence in an anti-corpus, patterns overrepresented in the former are said to be distinct to that. Our tf-idf approach is very similar, the probability of occurrence of each pattern in a score is normalised by how likely it is to occur across all other scores.

Nawba number	*Nawba* transliterated name	Number of Scores
1	*raml al-māya*	19
2	*al-isbahān*	13
3	*al-māya*	13
4	*rasd al-ḏayl*	18
5	*al-istihlāl*	24
6	*al-rasd*	10
7	*garībat al-ḥusayn*	13
8	*al-ḥiǰāz al-kabīr*	10
9	*al-ḥiǰāz al-māšriqī*	15
10	*ʻirāq al-ʻaǰam*	7
11	*al-ʻušš*āq	7

Table 1: Distribution of Scores across *nawabāt*

3. METHODOLOGY

Our analysis is implemented in Python, the code for which is available on Github [2] should the reader wish to reproduce this work. We use the music21 library [10] for processing scores and adopt the same convention for accidental notes in our reporting as in this toolkit, that is '#' for a sharp and '-' for a flat.

It is worth noting that we do not take into account any note/rest duration in this analysis, this decision is based on Chaachoo's theory which proposes the set of centos with no variation in their durations. Furthermore, we omit all octave information from our data, observing that melodic lines in our corpus very rarely jump between octaves and that same omission is made by Chaachoo in his work.

3.1 Dataset

Our dataset is a subset of 149 scores across all 11 Moroccan *nawabāt* from the CompMusic Arab-Andalusian corpus [25] of Dunya [23]. It has been selected such that each score is accompanied with the relevant *ṭāb'* metadata (and hence allows us to identify the *nawba* to which they belong). Each score is represented as an ordered list of consecutive notes (all scores are monophonic) with rests included. Table 1 shows the number of scores for each *nawba*.

3.2 Pre-processing

In analogy to a bag-of-words representation of a document, we represent each score as a bag-of-patterns. That is, we extract from each every possible n-gram up to a specified length, N. Any n-gram (or pattern as they will from here on be referred to as) that contains a rest, R, is discarded. For example, the bag-of-patterns for a score of [G, E, F, F, R, E, G, E] is:

$N = 2$: [GE, EF, FF, EG, GE];
$N = 3$: [GE, GEF, EF, EFF, FF, EG, EGE, GE];
$N \geq 4$: [GE, GEF, GEFF, EF, EFF, FF, EG, EGE, GE]

This method of motivic representation - where every possible sub-pattern is included alongside its parent patterns down to minimum length - is said to satisfy the *Sub-motif Existence Axiom* (SEA), first proposed in [18] and realised by Buteau in [2–4, 12].

3.3 TF-IDF Algorithm

Pattern importance is computed using the tf-idf statistic. Tf-idf, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus [26]. With each of our 149 scores in bag-of-patterns form, we compute the tf-idf for each pattern on a score by score basis, using sub-linear term frequency so as to down-weight very popular or very rare terms.

Averaging our tf-idf values for each pattern for each *nawba* provides a metric for pattern importance per *nawba*.

3.4 Post-processing

A common problem in melodic pattern recognition by computational means is in dealing with the large quantity of results. As is typical of such methods, a set of selection rules is required. After applying the tf-idf algorithm to identify significant patterns, they are filtered according to the following:

1. We consider a pattern having occurred less than 50 times per score per nawba as being insufficiently frequent given our data size, this choice is informed by inspecting the frequency of patterns identified by Chaachoo as characteristic.

2. We do not consider strings that contain less than three notes or more than ten notes a pattern, these are discarded. This decision is informed by referring to existing literature on melodic composition of Arab-Andalusian music [6, 20, 25] and is somewhat justifiable intuitively, the same lower bound can be found in [9, 21] for example.

3. Any pattern that is a substring of another pattern in our selection of significant patterns (subject to previous selection rules) is discarded in favour of the longer pattern. A similar approach is adopted in [9, 14, 17, 19].

4. RESULTS AND EVALUATION

For each *nawba* we rank its pattern content by our averaged tf-idf, cross-referencing our findings with the patterns identified as characteristic of the *nawba* from Chaachoo.

4.1 Visualising Patterns Importance

Fig.2 and Fig.3 show the top ranked patterns for *nawba* 1 and 2 respectively. Each bar is coloured such that patterns identified as characteristic of the *nawba* by Chaachoo are black, patterns that themselves contain patterns identified

[2] https://github.com/MTG/quantifyingcentones

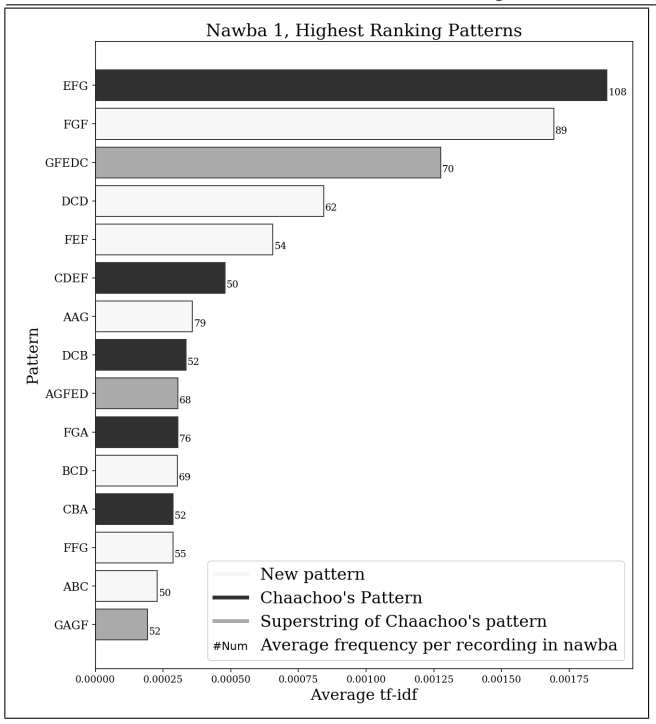


Figure 2: Ranked pattern importance for Nawba 1

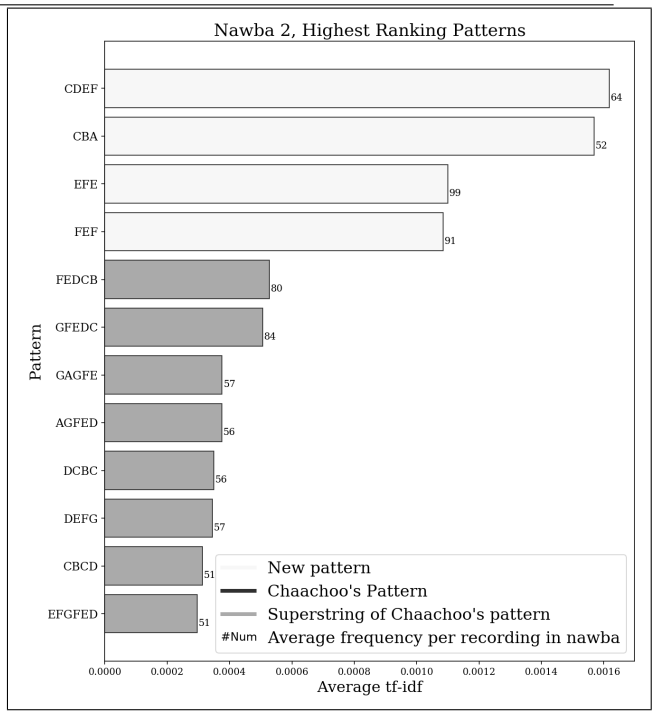


Figure 3: Ranked pattern importance for Nawba 2

by Chaachoo as characteristic of the *nawba* are grey and hitherto undocumented patterns white. Each bar is annotated with the average frequency of occurrence of that pattern in that *nawba*; that is the total number of occurrences in the corpus as a proportion of how many scores we have for that *nawba* (Table 1). These two examples are representative cases in which the most relevant patterns are a mix of Chachoo's patterns and new ones. Fig.4 displays the same score excerpt as in Fig.1 but with the new recognized relevant patterns for the *nawba*.

Table 3 presents the top 10 (where 10 exist) patterns determined by our analysis to be most characteristic of each *nawba*. The patterns identified by Chaachoo as characteristic of the *nawba* are bolded, those that themselves contain patterns identified by Chaachoo as characteristic of the *nawba* are italicized.

4.2 Evaluation

As a method of evaluating which patterns are most characteristic of each *nawba*, we compare the classification power of the patterns identified in this study to those documented by Chaachoo in [6] through a simple classifier.

4.2.1 Simple Classifier

An l2 regularised logistic regression model is applied to our corpus of 149 scores, we use a 60/40 train/test split and the frequency of occurrence of each pattern as features. In total, there are 184 patterns from Chaachoo's literature and 182 from our own.

We bootstrap our accuracy score on test 100 times. The results are displayed in Table 2.

Pattern Set	μ	σ
Chaachoo	70.8	4.9
Ours (Table 3)	72.2	5.0

Table 2: Bootstrapped accuracy when classifying nawba using two pattern sets (n=100)

5. DISCUSSION

The patterns discovered by this analysis are found to be as predictive as those of Chaachoo for classifying *nawabāt* and 44% of our top 109 patterns match the *centos* outlined in his works. Furthermore, we have demonstrated that in Arab-Andalusian music of the Moroccan tradition there exists at least 61 unique melodic patterns that occur as frequently (and in many cases more frequently) as those identified and documented to date and with at least as much significance as measured via our normalised frequency approach.

From a musical perspective it is clear that our patterns share some similarities; they constitute very simple cells, based on the relevant degrees of the *ṭāb'* (tonic and dominant). Since Arab-Andalusian music has uniform melodic contour and intervals rarely larger than a third, some of these new patterns could probably be clustered into other more generic groups. This could mean that we are not discovering new important material rather than variations of the known *centos* proposed previously by Chaachoo.

6. CONCLUSIONS

Our methodology is not without its limitations. The variability of our results is conditioned by factors such as whether we include rests in our patterns (and if so to what

Nawba	Most Characteristic Patterns
1	**EFG**, FGF, *GFEDC*, DCD, FEF, **CDEF**, AAG, **DCB**, *AGFED*, **FGA**
2	CDEF, CBA, EFE, FEF, *FEDCB*, *GFEDC*, *GAGFE*, *AGFED*, *DCBC*, *DEFG*
3	**B-AG**, EEF, FGEF, FGA, DCB, *AGFEDC*, *GAGFE*, EFEDC, CEE, *DEFG*
4	**CDEC**, ECD, *DEFG*, GAGFE, EDCDE, CDEF, AGFEDC, FAG, CDC, CCD
5	**ABC**, DCBA, **FAG**, *AGFEDC*, *EDCB*, *GAGFE*, *DEFG*, BCD, EDE, CBAG
6	**EDC**, *AGED*, *ABAGE*, *EDECD*, *EF#GA*, *F#GAG*, GF#G, GAB, AGG, GGA
7	EDE, *GAGFE*, *DEFG*, **CDE**, FGA, *AGFED*, AAG, *GFEDC*, CCD, DCD
8	E-DC, GAGF#E-, **AGF#E-D**, DEF#, BAGF#, **EF#GAG**, **CBAG**, DCB, BCB
9	EFG, FEF, FGF, **FGA**, FFG, *AGFED*, *GFEDC*, GAG, DEF, EFE
10	BAGF#E, **GF#ED**, *F#GAG*, CBAG, *EF#GA*, EDC, GAB, GGA, DCBA, ABC
11	GAG, GFED, DED, **GAB**, **FEDC**, *ABC*, EDCB, GGA, EFE, **CBAG**

Table 3: Top 10 most characteristic patterns per *nawba*. Those bolded exactly match patterns identified by Chaachoo. Those italicized are superstrings of one of those identified by Chaachoo

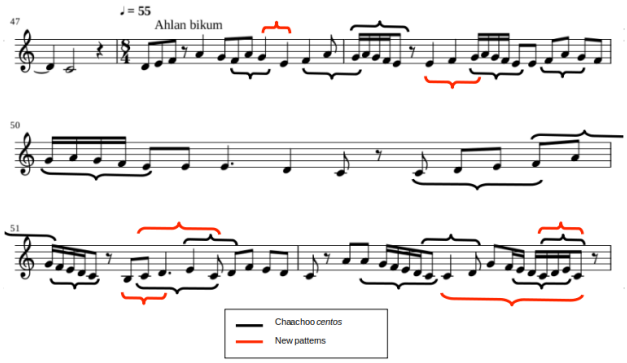


Figure 4: Score example with the new patterns discovered of the piece *Btayhi Rasd Dayl* of Nawba 4

length), the minimum frequency of occurrence for a pattern to be considered significant and maximum/minimum pattern length. The alteration of these and others like them would likely have marked consequences for our results. The justification for our decision was driven by necessity and intuition but would indeed benefit from further collaboration with experts of the tradition.

A further limitation exists in our inability to compare the significances computed here with other methods given that the tf-idf statistic is proportional to corpus size and hence only comparable within our training corpus.

However, as one of the first computational analysis on Arab- Andalusian music, we hope to have contributed to the musicological theory around centonization and hope that our approach may serve as a first reference for pattern recognition in Arab-Andalusian music, establishing the principles and basis for future and helpful study for musicological theories that can contribute to a better understanding and preservation of the musical tradition.

7. ACKNOWLEDGMENTS

The authors would like to acknowledge Mr. Amin Chaachoo, expert musicologist and experienced musician of this tradition, main curator of the Arab-Andalusian corpus and author of the music transcriptions and for his help with musicological explanations. This work was carried out with the support of the Musical Bridges project, funded by ObraSocialLaCaixa.

8. REFERENCES

[1] W. Apel. *Gregorian Chant*. A Midland book. Indiana University Press, 1958.

[2] C. Buteau. Melodic clustering within topological spaces of schumann's träumerei. *International Computer Music Conference, ICMC 2006*, 01 2006.

[3] C. Buteau and G. Mazzola. Motivic analysis according to Rudolph Réti: formalization by a topological model. volume 2, pages 117–134. Taylor & Francis, 2008.

[4] C. Buteau and J. Vipperman. Melodic clustering within motivic spaces: Visualization in openmusic and application to schumann's träumerei. volume 37, pages 59–66, 01 2009.

[5] A. Chaachoo. *La música andalusí Al-Ála: Historia, conceptos y teoría musical*. Editorial Almuzara, Córdoba, 2011.

[6] A. Chaachoo. *La musique hispano-arabe, al-Ala*. Univers musical. Editions L'Harmattan, 2016.

[7] G. Chewand and J. W. McKinnon. Centonization. *Oxford Music Online*, 2001.

[8] D. Conklin. Discovery of distinctive patterns in music. In *Intell. Data Anal., Vol 14*, pages 547–554, 2010.

[9] D. Conklin and C. Anagnostopoulou. Comparative pattern analysis of cretan folk songs. In *Proceedings of 3rd International Workshop on Machine Learning and Music*, MML '10, pages 33–36, New York, NY, USA, 2010. ACM.

[10] M. Cuthbert, C. Scott, and C. Ariza. Music21: A toolkit for computer-aided musicology and symbolic music data. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 637–642. International Society for Music Information Retrieval, 2010.

[11] P. Ferretti and A. Agaësse. *Esthétique grégorienne ou Traité des formes musicales du chant grégorien. Volume I. Traduit de l'italien par Dom A. Agaësse*. Desclée, 1938.

[12] H. Fripertinger and L. Reich. Topological motive spaces, and mappings of scores' motivic evolution trees. *Grazer mathematische Berichte*, (347):35, 2005.

[13] M. Guettat. La musique arabo-andalouse, l'empreinte du maghreb. page 560, 2000.

[14] J. Hsu, C. Liu, and A. L. P. Chen. Discovering nontrivial repeating patterns in music data. In *IEEE Transactions Multimedia*, pages 311–325, 2001.

[15] B. Janssen, W. De Haas, A. Volk, and P. Van Kranenburg. Discovering repeated patterns in music: state of knowledge, challenges, perspectives. In *Proc. of the 10th International Symposium on Computer Music Multidisciplinary Research, Marseille, France*, volume 20, page 74, 2013.

[16] I. Karydis, A. Nanopoulos, and Y. Manolopoulos. Finding maximum-length repeating patterns in music databases. In *Multimedia Tools and Applications*, volume 32, pages 49–71, 2006.

[17] I. Karydis, A. Nanopoulos, and Y. Manolopoulos. Closed patterns in folk music and other genres. In *6th International Workshop on Folk Music Analysis*, pages 15–17, 2016.

[18] G. Mazzola. *The Topos of Music: Geometric Logic of Concepts, Theory, and Performance*. Birkhäuser, 2011.

[19] D. Meredith, K. Lemström, and G. A. Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, 31(4):321–345, 2002.

[20] M.Sordo, A. Chaachoo, and X.Serra. Creating corpora for computational research in arab-andalusian music. In *Proceedings of the 1st International Workshop on Digital Libraries for Musicology (DLfM '14). ACM, New York, NY, USA, 1-3. DOI: https://doi.org/10.1145/2660168.2660182*, 2014.

[21] O. Nieto and M. M. Farbood. Perceptual evaluation of automatically extracted musical motives. In *Proceedings of the 12th International Conference on Music Perception and Cognition*, pages 723–727, 2012.

[22] C. Poché. *La música arábigo-andaluza (con CD)*. Músicas del mundo. Ediciones Akal, 1997.

[23] A. Porter, M. Sordo, and X. Serra. Dunya: A system for browsing audio music collections exploiting cultural context. In *14th International Society for Music Information Retrieval Conference (ISMIR 2013)*, pages 101–106, Curitiba, Brazil, 04/11/2013 2013.

[24] I. Y. Ren, H. V. Renand Koop, A. Volk, and W. Swierstra. In search of the consensus among musical pattern discovery algorithms. In *Proceedings of the 18th International Society for Music Information Retrieval Conference*, pages 671–678, 2017.

[25] R. Caro Repetto, N. Pretto, A. Chaachoo, B. Bozkurt, and X. Serra. An open corpus for the computational research of arab-andalusian music. In *Proceedings of 5th International Conference on Digital Libraries for Musicology (DLfM 2018)*, pages 78–86, 2018.

[26] H. Wu, R. W. Luk, K. F. Wong, and K. L. Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Trans. Inf. Syst.*, 26(3):13:1–13:37, June 2008.

TEMPORAL CONVOLUTIONAL NETWORKS FOR SPEECH AND MUSIC DETECTION IN RADIO BROADCAST

Quentin Lemaire
KTH Royal Institute of Technology
qle@kth.se

Andre Holzapfel
KTH Royal Institute of Technology
holzap@kth.se

ABSTRACT

The task of speech and music detection aims at the automatic annotation of potentially overlapping speech and music segments in audio recordings. This metadata extraction process finds important applications in royalty collection for broadcast audio. This study focuses on deep neural network architectures made to process sequential data, and a series of recent architectures that have not yet been applied for this task are evaluated, extended and compared with a state-of-the-art architecture. Moreover, different training strategies are evaluated, and we demonstrate the advantages of a pre-training procedure with low-quality data that facilitates the combination of heterogeneous datasets. The study shows that Temporal Convolution Network (TCN) architectures can outperform state-of-the-art architectures. In specific, the novel non-causal TCN extension introduced in this paper leads to a significant improvement of the accuracy.

1. INTRODUCTION

The location of speech and music segments in large amounts of audio recordings is an important metadata information especially in the context of royalty collection in broadcasting. The task of speech and music detection is a multi-label problem, each frame can be labeled as music, speech, both, or neither of both. Assuming potential overlaps between the classes makes speech and music detection a complex and unsolved task, as opposed to a simple discrimination of segments into either speech or music [12]. Apart from royalty collection, a speech and music detection system can be useful to extract the relevant parts of the audio on which to apply other meta-data extraction such as speech-to-text, genre classification or music fingerprinting.

The first approaches to speech/music detection – discussed in the work of Carrey et al. [5] – focused on manually designed features and subsequent classification. More recently, features are automatically learned from spectrogram images using deep neural networks for various audio tasks [4, 7, 14, 22, 27, 33, 37]. End-to-end learning systems with waveform-audio input have been compared with spectrogram-based learning, with better results of the latter approach [16]. This may however be due to the lack of sufficient data in the particular case, as discussed by Pons et al. [24].

Recently, the Temporal Convolutional Network (TCN) [2] showed promising results on tasks involving sequential data. No study to date has compared the TCN architecture with other deep learning architectures such as Recurrent Neural Networks (RNN) for the task of speech and music detection. A key contribution of this paper is the investigation of a novel non-causal TCN architecture. This is of special interest to various applications where real-time analysis is not a constraint. The results of this study demonstrate that TCN architectures can outperform RNN. The final system is compared with the state of the art on the MIREX dataset [1] with results that further document the high performance of the non-causal TCN.

Another contribution of this paper is the use of an efficient pre-training procedure with low-quality data that facilitates the combination of heterogeneous datasets. We compiled the most extensive data resource available until now for the task of speech and music detection, and trained and tested networks using this heterogeneous data resource. Our results document the advantage of the pre-training procedure, and we provide the code of this study as a toolbox for the systematic comparison of system architectures for the speech and music detection task.

The following section is a review of existing work with neural networks on speech and music detection and tasks that employ similar methods. Section 3 explains the method that will be applied in this study. Section 4 presents the results that are discussed in Section 5. Finally, Section 6 draws various conclusions about this work.

2. BACKGROUND

The speech and music detection problem is a sound event detection problem. It applies to an audio stream containing temporal segments of speech and/music in arbitrary positions. Segments of speech and audio may overlap. Furthermore, some parts of the audio might contain neither music nor speech, but task-irrelevant content such as environmental sounds, footsteps or keyboard typing sounds.

Attribution: Quentin Lemaire, Andre Holzapfel. "Temporal Convolutional Networks for Speech and Music Detection in Radio Broadcast", 20th International Society for Music Information Retrieval Conference, Delft, The Netherlands, 2019.

[1] https://www.music-ir.org/mirex/wiki/2018:Music_and/or_Speech_Detection

Figure 1 shows an overview of the processing chain typically applied in speech/music detection and other related tasks, such as acoustic event detection. The following subsections provide an overview of examples of these steps from the literature.

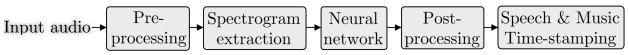


Figure 1: Overview of the processing chain

2.1 Pre-Processing & Spectrogram Extraction

The most commonly used features in the literature are the Mel-scaled log-magnitude spectrograms [14,18,22,27,33]. To obtain them, a Short-Time Fourier Transform (STFT) is computed from the audio sample containing the discrete-time signal. Only the magnitudes are kept and a Mel-scaled Filterbank is applied. Finally, the obtained coefficients are put on a log magnitude scale and normalized.

Data augmentation is a pre-processing solution that artificially creates new data based on the available ones by applying various manipulations either in the time-domain [27] or on the spectrograms [29].

2.2 Network Architectures

In the context of deep neural networks, there are two different ways described in the literature to handle the speech and music detection task (or related tasks). The first possibility is to treat the audio as non-sequential data by working on small excerpts independently which is mainly used for classification tasks. The state-of-the-art architectures are based on the Convolutional Neural Networks (CNN) [14, 19, 27, 33].

The second possibility is to use a sequence model for the audio. An audio sample is injected into the network and each frame will be classified as music, speech, both music and speech or neither of both. The state-of-the-art architectures are mainly based on the Bidirectional Long Short-Term Memory (B-LSTM) [18, 22], which are a type of RNN.

Since the systems have been evaluated on differing test sets, no conclusion regarding the best performing architecture can be obtained from the above cited work. Moreover, music and speech have different temporal and spectral properties than environmental sounds, and it is therefore not clear if results from other tasks apply to speech and music detection.

2.2.1 Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks (CLDNN)

Information from the long-term context in past and future can be very relevant for the classification and especially for borderline cases. It motivates the use of sequential models for the detection tasks. However, CNNs have obtained state-of-the-art results in image and audio feature extraction, which motivated the combination of RNN and CNN in the CLDNN architecture [26] to use the strength of both. CLDNN are divided into three sections: first, the input goes through several convolutional layers. Then, the result goes through a classic LSTM network and finally, several fully connected (FC) layers are applied. This architecture was shown to outperform CNN, RNN or DNN based approaches on various datasets [26]. This architecture has also been reused in several subsequent audio and music related studies [4, 7, 10], but has so far not been applied to speech and music detection.

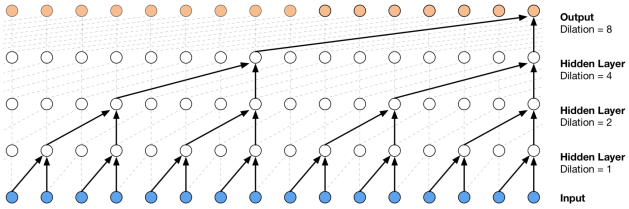


Figure 2: Visualization of a stack of dilated causal convolutional layers. Source: [35]

2.2.2 Temporal Convolutional Network (TCN)

Recently, the Temporal Convolution Network (TCN) was introduced [2] as a simple and flexible architecture using CNN for sequential learning. This architecture is based on the previous work trying to use only CNNs for sequential learning [13, 35] but it remains much simpler. A TCN cell is based on causal and dilated convolutions, and on residual blocks [15]. Figure 2 illustrates a dilated causal convolutional layer. Dilated convolutions permit to retrieve information from far in the past without extensive computation.

TCN architectures have not been used for speech and music detection yet. Therefore, within this paper, we compare four architectures: the standard TCN, a novel non-causal extension of the TCN, the BLSTM and the CLDNN architectures. This non-causal TCN architecture is designed to combine the strengths of both BLSTM and CNN architectures: the bidirectional long-term memory of the BLSTM architecture and the performance and parallelizability of the CNN architecture.

2.3 Post-processing and Evaluation

The network output vector contains two coefficients between 0 and 1, one for speech and one for music, and commonly 0.5 is used as a threshold for the labeling decision [18, 30]. A probabilistic model has been used by [11] to smooth the output and reduce the noise.

Two evaluation methods for speech and music detection – segment-level evaluation and event-level evaluation [21] – are applied within MIREX [1] and in most related studies. Segment-level evaluation compares the labels of segments of fixed size given by the algorithm with the ground-truth labels. Event-based evaluation takes the whole ground-truth events into account and a time-window tolerance is allowed. This evaluation is either made on the on-set of the events or on both the on-set and the off-set.

3. METHODOLOGY

3.1 Datasets

A larger amount of previously compiled datasets have been obtained and combined for this study (*MUSAN Corpus* [31] (MUSAN), *GTZAN Speech and Music Dataset* [34] (GTZAN), *Scheirer & Slaney Music Speech Corpus* [28] (SSMSC), *MuSpeak Speech and Music Detection Dataset* [2] (MuSpeak), *OFAI Speech and Music Detection Dataset* [30] (OFAI), *ESC-50: Dataset for Environmental Sound Classification* [23] (ESC)). In addition, a newly compiled dataset (Sveriges Radio dataset) was included as well, which is composed of songs from different genres that are played on the radio (42h 57mn) and of speech files (17h 24mn) that are extracted from radio programs.

Table 1: Categorization of the datasets according to the precision of their labels.

Low-quality datasets (LQ)	High-quality datasets (HQ)
MUSAN, GTZAN	OFAI
SSMSC, ESC	MuSpeak
Sveriges Radio dataset	

The datasets can be separated into two families regarding the precision of their labels, "low-quality datasets" (LQ) and "high-quality datasets" (HQ) as presented in Table 1. The HQ datasets are labeled at the frame-level, which precisely corresponds to the goal of the speech and music detection task to classify each frame of the audio. On the other hand, the LQ datasets are labeled at the file-level, which assumes that all the frames in the audio recording have the same label. It corresponds to a low-precision version of the targeted system (*e.g.* pauses in speech are not annotated).

The datasets were split into training, validation, and test set (if not already done in the source material). HQ datasets were split into 70% training set, 20% validation set and 10% test set, and LQ datasets into 80% training set and 20% validation set, because labels are not precise enough to be used for testing. The resulting data collection contains about 78h15mn / 86h54mn / 15mn / 8h49mn of speech, music, both speech and music, and task-irrelevant data in the LQ data, and 47h12mn of audio combining all the previous categories in the HQ data, making it the most extended data resource compiled so far for the task of speech and music detection.

3.2 Architectures

Three previous sequential architectures and one novel extension will be compared in this paper. All four architectures are followed by a dense layer that reduces the number of coefficients to 2, one for music and one for speech, and by a sigmoid activation to have coefficients between 0 and 1. The range of allowed values for hyper-parameter were chosen to cover the ranges previously presented in literature, and are listed in the following tables for each architecture. In each case, a dropout randomly selected from 0.05 and 0.5 was added.

[2] `http://mirg.city.ac.uk` - visited 09/11/2018

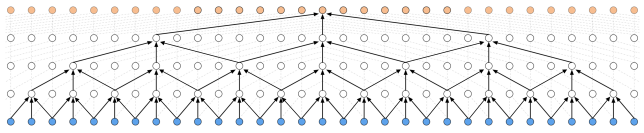


Figure 3: Visualization of a stack of dilated non-causal convolutional layers. The architecture presented in Figure 2 was made non-causal to take both past and future into account for the prediction. The kernel size had to be increased by 1.

The TCN architecture as introduced in [2] is causal. In this paper, we propose a novel extension of the TCN that takes future data into account, the non-causal TCN (ncTCN). To this end, the dilated convolutions were made non-causal, as shown in Figure 3. The use of non-causal dilated convolutions was previously shown to be successful for image processing with the Dilated Temporal Fully-Convolutional Neural Network (DTFCN) architecture [6].

Table 2: Hyperparameters for the four architectures.

Architecture 1: B-LSTM	
Num. of layers	1, 2, 3, 4
Units by layer	25, 50, 75, ... 250
Architecture 2: CLDNN	
Num. of layers	1, 2, 3
Kernel size (conv. layers)	3, 5 or 9
Number of LSTM layers	1, 2, 3
Units per LSTM layer	25, 50, 75, ... 150
Num. of fully-connected layers	1, 2, 3
Units per fully-connected layer	25, 50, 75, ... 150
Architectures 3 & 4: TCN & non-causal TCN (ncTCN)	
Num. of layers	1, 2, 3, 4
Num. of stacks	3, 4, 5, ... 10
Kernel size	3, 5, 7, ... 19
Skip some connections	true/false
Dilatations	$[2^0, 2^1, ..., 2^{N_D}]$ $N_D = 3, 4, ..., 8$
Num. of filters by layer	8, 16, 32

3.3 Comparison methodology

The study in this paper is composed of two phases: The first phase conducts a comparison of four sequential neural network architectures (B-LSTM, CLDNN, TCN, ncTCN) for the speech and music detection task. In order to facilitate the comparison in a reasonable time, two assumptions were made. First, the neural network architecture achieving the best performances on a sub-training set is likely to achieve the best performances on the total training set. And, second, the neural network architecture achieving the best performance with a restricted number of pa-

rameters is likely to perform best without this restriction. Therefore, the comparison was done with a limited number of parameters and on a sub-dataset. A Bayesian hyper-parameter optimization with a Tree of Parzen Estimators (TPE) surrogate [3] was performed for each architecture on the *OFAI* and the *MuSpeak* datasets, with the number of hyper-parameters restricted to 1 million. The best set of hyper-parameters was then used to train each architecture over more epochs to get a final validation loss, and the architecture achieving the lowest validation loss was kept for the second phase.

In the second phase, the best-performing architecture from the first phase is further optimized without limiting the number of hyper-parameters. This hyper-parameter optimization was done on *OFAI*, *MuSpeak*, and *ESC* to have a more balanced dataset between music, speech, and task-irrelevant content. Then the architecture was trained on more training data and evaluated on the test set. However, due to the heterogeneity in the quality of the labels, explained in Section 3.1, only the high-quality datasets represent the target of the network. Therefore, four strategies to take advantage of both HQ and LQ data were compared. The four strategies were to (1) train the network on the high-quality datasets, (2) to train the network on the low-quality datasets, (3) to train the network on both the low and the high-quality datasets at the same time, and (4) to pre-train the network on the low-quality datasets and then train on the high-quality datasets to fine-tune the parameters.

The final system was evaluated on two different test sets. The first test set is the in-house test set described in subsection 3.1 and it shows the generalization of the system on similar data. In order to compare the system with several state-of-the-art algorithms, and to assess the generalization of the system on data different than the one used for the training, the dataset number 2 of the 2018 MIREX Competition [1] [11] was used as a second test set.

The evaluation methods and parameters from MIREX 2018 were applied for comparisons, using the implementation of sed_eval [21]. The segment-level evaluation was conducted with segments of 10ms. The event-level evaluation was performed with a tolerance time-window size of 500ms on on-set only and both on-set and off-set. Precision, Recall, and F-measure were computed for segment level (P_s, R_s, F_s), and event-level (P_e, R_e, F_e).

3.4 Pre-processing

Before the training, the audio samples were re-sampled to 22.05 kHz mono audio samples split into files of 90 s. Then a Short Time Fourier Transform (STFT) with a Hann window, a frame length of 1024 and a hop size of 512 samples was computed. Only the squared magnitude (the power spectrum) was kept and saved for the training. During the training, data augmentation was applied to the saved spectrograms and then, a Mel-filterbank with 80 triangular filters between 27.5 Hz and 8 kHz was applied. Finally, the data were put on a logarithmic scale and normalized to a zero mean and unit variance over the training set.

The data augmentation pipeline applied to each spectrogram used the implementation and paramatrisation of [29], applying time stretching, pitch shifting, Gaussian filtering, loudness manipulation, and block mixing. Data from the HQ datasets and LQ data labeled as task-irrelevant content was augmented without block mixing. Otherwise, couples of speech and of music spectrograms were created, passed individually in the data augmentation pipeline and then each couple was mixed together with random overlap. It helps to have a more balanced dataset by artificially creating overlaps between speech and music.

Broadcast audio is characterized by overlaps between speech and music, for instance in jingles, commercial ads, and transitions between pieces of music. Data augmentation helps to obtain larger overlaps that resemble this characteristics of broadcast audio.

3.5 Training

To allow parallel computation and to speed up the backward pass, mini-batches are used with a sequence length of 270 and a batch size of 32. Binary cross-entropy was minimized during training, and stochastic gradient descent with momentum $m = 0.9$ [25] was used. When the validation loss did not improve after three epochs, the learning rate was divided by 10. Dropout [32] was used and the training was stopped whenever the validation loss had not improved in 5 consecutive epochs.

3.6 Post-processing

A threshold of 0.5 was applied to the output of the networks. A simple strategy was applied to smooth the output and delete spurious breaks or events. To this end, thresholds for the minimal duration of speech (Dur_{sp}) and music (Dur_{mus}) events were defined, respectively. Furthermore, thresholds for the minimal duration of a break in speech (Brk_{sp}) and music (Brk_{mus}) were defined. In order to specify values for these four duration thresholds, the training set was analyzed to obtain statistics on the lengths of the events and the breaks. To choose the best values between the relevant values found with the analysis, each set of values was evaluated on the validation set and the set achieving the best performances was selected. The effect of post-processing will be analyzed separately in the results.

3.7 Implementation

The implementation is done with *Keras* [9] using TensorFlow as a backend [1] and the library *keras-tcn* [3] is used for the TCN implementation. The code is provided as a new framework for speech and music detection that allows comparison between configurations with different architectures, datasets and hyper-parameters. The implementation has been made available on GitHub. [4]

[3] https://github.com/philipperemy/keras-tcn
[4] https://bit.ly/2XcuzsJ

4. RESULTS

4.1 Comparison

After each hyper-parameter optimization, the best configuration has been trained until the early-stopping. Figure 4 shows the micro-averaged [36] ROC curve of the 4 different architectures. The architecture that achieved the best performances under the constraints of the experiment is the non-causal TCN. All three architectures that have not yet been applied to this task (ncTCN, TCN, CLDNN) outperform the BLSTM architecture.

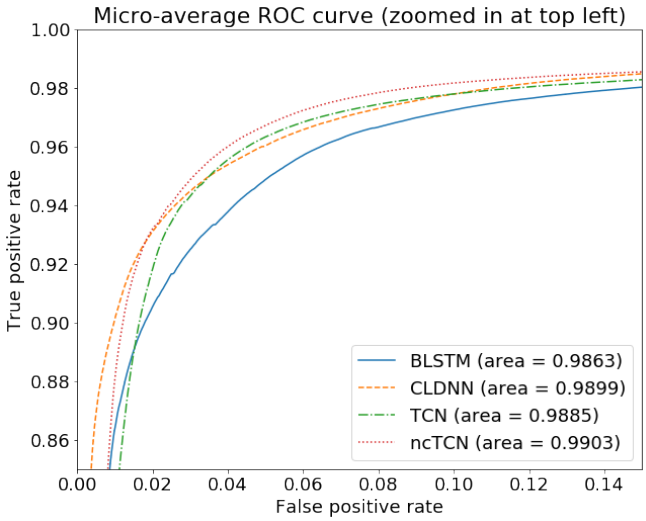


Figure 4: ROC curve micro-averaged over speech and music.

4.2 Dataset strategies

A new hyper-parameter optimization allowing for a bigger range of hyper-parameters was performed for the best-performing architecture (ncTCN). The batch size was reduced to 16 to allow the GPU to work with bigger architectures. The resulting architecture was trained with the four different strategies explained in Section 3.3. The strategy that achieved the lowest validation loss (Table 3) on the targeted high-quality dataset (HQ loss) is pre-training on the low-quality dataset, and subsequent training on the high-quality dataset (Strategy 4).

Table 3: Validation loss of the different strategies to use on high and low quality datasets.

Strategy	LQ loss	LQ/HQ loss	HQ loss
(1) LQ	**0.097**	0.125	0.232
(2) HQ	0.365	0.323	0.096
(3) LQ/HQ	0.098	**0.101**	0.136
(4) Pre-train	0.222	0.181	**0.070**

4.3 Post-processing (PP)

The high-quality training set was analyzed to set the four duration thresholds for the post-processing (see Section 3.6). The 1st, 5th and 10th percentiles were considered to obtain threshold values. For instance, in the case of the 5th percentile for the music event duration, it means that 95 % of the music events in the training set have a length exceeding the threshold. Table 4 presents the evaluation of the system with several post-processing methods based on the values from the different percentiles.

Table 4: F-measures for segment-level evaluation (F_s) and event-level evaluation (F_e) on the high-quality validation set by percentile. Underlined values denote statistically significant differences to the value one row above (paired-sample t-test, $p < 0.05$).

PP	F_s (segment)			F_e (event)		
	All	Mus.	Sp.	All	Mus.	Sp.
None	**0.973**	**0.982**	**0.951**	0.182	0.247	0.129
1st	**0.973**	**0.982**	0.950	0.510	0.589	0.417
5th	**0.973**	**0.982**	0.950	0.544	0.615	0.454
10th	0.971	0.981	0.949	**0.547**	**0.617**	**0.459**

The different impact of the post-processing on the segment and on the event-level evaluation is caused by the fact that a small modification at the frame level has a limited impact on the segment-level evaluation, but it can have a significant impact on the event-level evaluation. The set of values from the 5th percentile was selected for the rest of the evaluation since it represents a good compromise between a decrease in the segment-level evaluation and an increase in the event-level evaluation.

4.4 Evaluation of the ncTCN

Table 5: Segment-level evaluation of the final system

	All	Music	Speech
F_s	0.968	0.971	0.957
P_s	0.963	0.969	0.944
R_s	0.973	0.973	0.971

The results of the segment-level and event-level evaluations on the test set 1 are presented in Table 5 and Table 6, respectively. For the segment-level evaluation, the proposed non-causal TCN system obtains an F-measure of 0.971 for the music and of 0.957 for the speech. Due to the marginal effect of post-processing on the segment level, only results with post-processing are depicted. For the event-level evaluation, the system obtains clearly superior results with the post-processing. For example, the overall F-measure on both onset and offset increases from 0.151 without post-processing to 0.417 with post-processing.

Finally, the non-causal TCN was evaluated on the dataset 2 of the 2018 MIREX competition. The results are presented in Table 7 and Table 8 and the results of the other systems are taken from the MIREX website [1]. For the segment-level evaluation, the system obtains the best F-measure (0.946) on speech. On music, the system obtains an F-measure of 0.879 and the best system of the

Table 6: Event-level evaluation of the final system. Underlined values denote statistically significant difference to the value without pre-processing (paired-sample t-test, $p < 0.05$).

		Ons.			On/Offs.		
PP		All	Mus.	Sp.	All	Mus.	Sp.
No	F_e	0.248	0.248	0.248	0.151	0.177	0.115
	P_e	0.153	0.146	0.165	0.930	0.104	0.760
	R_e	**0.650**	**0.828**	**0.499**	**0.394**	0.587	**0.231**
Yes	F_e	**0.653**	**0.782**	**0.519**	**0.417**	**0.590**	**0.236**
	P_e	**0.733**	**0.778**	**0.671**	**0.447**	**0.587**	**0.305**
	R_e	0.589	0.787	0.422	0.376	**0.593**	0.192

Table 8: F-measure comparison with other algorithms on the test set 2 of MIREX 2018 for the event-level evaluation and a tolerance time-window of 500ms.

Algorithm	Music		Speech	
	Ons.	On/Offs.	Ons.	On/Offs.
[8, a]	0.087	0.023	**0.223**	**0.077**
[8, b]	0.073	0.020	0.192	0.051
[8, c]	0.068	0.015	0.206	0.052
[20, a]	0.141	0.016	0.063	0.002
[20, b]	0.154	**0.031**	0.116	0.021
[20, c]	0.152	0.022	0.080	0.015
ncTCN	**0.162**	0.0169	0.216	0.070

competition obtains 0.923. For the event-level evaluation, the system obtains the best F-measure of 0.162 for the onset evaluation of the music. For the other cases, the system does not come first but achieves good results and comes second in 2 of the 3 remaining cases.

Table 7: Comparison with other algorithms on test set 2 of MIREX 2018 for segment-level evaluation. Architectures [8](a, b, c) use a Multi-layer Perceptron to classify both Mel-Frequency Cepstral Coefficient and features extracted by a SampleCNN [17] architecture. Architectures [20](a) are based on a logistic regression classifier and architectures [20](b, c) are based on a Deep Residual Network [15].

	Music			Speech		
Algo.	F_s	P_s	R_s	F_s	P_s	R_s
[8, a]	0.786	0.813	0.760	0.846	0.967	0.751
[8, b]	0.759	0.768	0.750	0.789	**0.975**	0.663
[8, c]	0.797	0.797	0.797	0.823	0.964	0.718
[20, a]	**0.923**	0.977	0.875	0.933	0.913	0.953
[20, b]	0.916	0.925	0.907	0.914	0.933	0.896
[20, c]	0.879	**0.979**	0.797	0.897	0.829	**0.978**
ncTCN	0.879	0.790	**0.990**	**0.946**	0.949	0.943

5. DISCUSSION

Based on the curves from Figure 4, our experiments suggest that the TCN-based architectures are outperforming the RNN-based architectures. Moreover, the TCN-based architectures train faster than the RNN-based architectures at similar sizes (around 80s/epoch for the TCN-based architectures and 340s/epoch for the RNN-bases architectures), results that corroborate conclusions [2] in different task contexts. The proposed non-causal TCN achieves better results than the causal TCN.

We also demonstrated that pre-training the neural network on a low-quality dataset prior to training it on the high-quality dataset improves the validation loss. Table 3 highlights the difference between the different strategies and it shows that the HQ loss function goes down from 0.096 to 0.070 by pre-training the network on low-quality datasets, it represents a relative improvement of 27 %. The analysis of the training set provides relevant thresholds for the post-processing method. Those values allow an improvement for the event-level evaluation on the validation set without harming the segment-level evaluation. Results on the test set (Table 5 and 6) confirm this analysis made on the validation set. Instead of applying some thresholds, more sophisticated probabilistic models may further improve the post-processing.

For the event-level evaluation, the overall results of all algorithms are much lower compared to the segment-level results. It shows one of the limits of the event-level evaluation: it is difficult to standardize precise boundaries for the events and especially the speech events. The results of the event-level evaluation highly depend on the rules that have been chosen during the annotation of the training set. Therefore, the segment-level evaluation might be more relevant to compare different algorithms on a test set with unknown annotation rules.

End-to-end learning may be considered an important area for future research. Pre-training an end-to-end learning solution on low-quality datasets and fine-tuning it on high-quality datasets might be a viable solution to overcome the expensive price of labeling data. This method was shown to be successful in this study and it might be suitable for other tasks.

6. CONCLUSION

In this paper, various architectures have been compared in a speech and music detection task. The findings are consistent with previous studies, demonstrating that convolutional architectures can yield better performance and are faster to train than RNN-based architectures for sequence modeling. Furthermore, the novel non-causal TCN can improve performance when real-time computation is not a constraint. Low-quality data was successfully used to improve the system performance on high-quality data. It provided a better starting point for the learning phase and it converged faster towards a lower minimum. Through the MIREX evaluation, the final system has demonstrated to perform well in relation to the state of the art. This encourages further exploration of TCN and non-causal TCN architectures for sequence modeling tasks.

7. ACKNOWLEDGEMENTS

This work was supported by the NordForsk's Nordic University Hub "Nordic Sound and Music Computing Network - NordicSMC" (proj. nr. 86892), and conducted in collaboration with Swedish Radio.

Special thanks to Christofer Bustad, Paul Nygren, and Ragnar Schön from Sveriges Radio for their help and advice during this project.

8. REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA, 2016. USENIX Association.

[2] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018.

[3] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pages 2546–2554, USA, 2011. Curran Associates Inc.

[4] Emre Cakir, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, Tuomas Virtanen, Emre Cakir, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 25(6):1291–1303, June 2017.

[5] Michael J Carey, Eluned S Parris, and Harvey Lloyd-Thomas. A comparison of features for speech, music discrimination. In *IEEE International Conference on Acoustics, Speech, and Signal Processing.*, volume 1, pages 149–152, 1999.

[6] Noshaba Cheema, Somayeh Hosseini, Janis Sprenger, Erik Herrmann, Han Du, Klaus Fischer, and Philipp Slusallek. Dilated temporal fully-convolutional network for semantic segmentation of motion capture data. *CoRR*, abs/1806.09174, 2018.

[7] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2392–2396. IEEE, 2017.

[8] Minsuk Choi, Jongpil Lee, and Juhan Nam. Hybrid features for music and speech detection. *Music Information Retrieval Evaluation eXchange (MIREX)*, 2018.

[9] François Chollet et al. Keras. `https://keras.io`, 2015.

[10] Heinrich Dinkel, Yanmin Qian, and Kai Yu. Investigating raw wave deep neural networks for end-to-end speaker spoofing detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(11):2002–2014, 2018.

[11] David Doukhan, Eliott Lechapt, Marc Evrard, and Jean Carrive. Ina's MIREX 2018 music and speech detection system. In *Music Information Retrieval Evaluation eXchange (MIREX 2018)*, 2018.

[12] Lars Ericsson. *Automatic Speech/music Discrimination in Audio Files*. M.Sc. thesis, KTH Royal Institute of Technology, 2010.

[13] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017.

[14] T. Grill and J. Schlüter. Two convolutional neural networks for bird detection in audio signals. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1764–1768, Aug 2017.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[16] Lars Hertel, Huy Phan, and Alfred Mertins. Comparing time and frequency domain for audio event recognition using deep learning. In *Neural Networks (IJCNN), 2016 International Joint Conference*, pages 3407–3411. IEEE, 2016.

[17] Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam. Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms. *CoRR*, abs/1703.01789, 2017.

[18] S. Leglaive, R. Hennequin, and R. Badeau. Singing voice detection with deep recurrent neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 121–125, April 2015.

[19] Thomas Lidy and Alexander Schindler. CQT-based convolutional neural networks for audio scene classification. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, pages 60–64, September 2016.

[20] Matija Marolt. Music/speech classification and detection submission for MIREX 2018. *Music Information Retrieval Evaluation eXchange (MIREX)*, 2018.

[21] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. Metrics for polyphonic sound event detection. *Applied Sciences*, 6(6):162, 2016.

[22] Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. Recurrent neural networks for polyphonic sound event detection in real life recordings. *CoRR*, abs/1604.00861, 2016.

[23] Karol J. Piczak. ESC: Dataset for Environmental Sound Classification. In *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pages 1015–1018. ACM Press.

[24] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik M Schmidt, Andreas F Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR 2018)*, 2018.

[25] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[26] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference*, pages 4580–4584. IEEE, 2015.

[27] J. Salamon and J. P. Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, March 2017.

[28] E. Scheirer and M. Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 1331–1334 vol.2, April 1997.

[29] Jan Schlüter and Thomas Grill. Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, Malaga, Spain, 2015.

[30] Jan Schlüter and Reinhard Sonnleitner. Unsupervised feature learning for speech and music detection in radio broadcasts. In *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx)*, pages 369–376, York, UK, September 2012.

[31] David Snyder, Guoguo Chen, and Daniel Povey. MUSAN: A Music, Speech, and Noise Corpus, 2015. arXiv:1510.08484v1.

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[33] N. Takahashi, M. Gygli, B. Pfister, and L. Van Gool. Deep Convolutional Neural Networks and Data Augmentation for Acoustic Event Detection. *ArXiv e-prints*, April 2016.

[34] George Tzanetakis and Perry Cook. Marsyas: A framework for audio analysis. *Org. Sound*, 4(3):169–175, December 1999.

[35] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *9th ISCA Speech Synthesis Workshop*, page 125, 2016.

[36] Yiming Yang. An evaluation of statistical approaches to text categorization. *Inf. Retr.*, 1(1-2):69–90, May 1999.

[37] Yu Zhang, William Chan, and Navdeep Jaitly. Very deep convolutional networks for end-to-end speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 4845–4849. IEEE, 2017.

TOWARDS EXPLAINABLE MUSIC EMOTION RECOGNITION: THE ROUTE VIA MID-LEVEL FEATURES

Shreyan Chowdhury Andreu Vall Verena Haunschmid Gerhard Widmer
Institute of Computational Perception, Johannes Kepler University Linz, Austria
firstname.lastname@jku.at

ABSTRACT

Emotional aspects play an important part in our interaction with music. However, modelling these aspects in MIR systems have been notoriously challenging since emotion is an inherently abstract and subjective experience, thus making it difficult to quantify or predict in the first place, and to make sense of the predictions in the next. In an attempt to create a model that can give a musically meaningful and intuitive explanation for its predictions, we propose a VGG-style deep neural network that learns to predict emotional characteristics of a musical piece together with (and based on) human-interpretable, mid-level perceptual features. We compare this to predicting emotion directly with an identical network that does not take into account the mid-level features and observe that the loss in predictive performance of going through the mid-level features is surprisingly low, on average. The design of our network allows us to visualize the effects of perceptual features on individual emotion predictions, and we argue that the small loss in performance in going through the mid-level features is justified by the gain in explainability of the predictions.

1. INTRODUCTION

Emotions – portrayed, perceived, or induced – are an important aspect of music. MIR systems can benefit from leveraging this aspect because of its direct impact on human perception of music, but doing so has been challenging due to the inherently abstract and subjective quality of this feature. Moreover, it is difficult to interpret emotional predictions in terms of musical content. In our quest for computer systems that can give musically or perceptually meaningful justifications for their predictions [17], we turn to the notion of *'mid-level perceptual features'* as recently described and advocated by several researchers [1, 5]. These are musical qualities (such as rhythmic complexity, or perceived major/minor harmonic character) that are supposed to be musically meaningful and intuitively recognizable by most listeners, without requiring music-theoretic knowledge. It has been shown previously that there is considerable consistency in human perception of these features; that they can be predicted relatively well from audio recordings; and that they also relate to the perceived emotional qualities of the music [1].

That is the motivation for the work to be reported here. Our goal is to use mid-level features as a basis for providing explanations of (and thus get further insights into, or handles on) a model's emotion predictions, by training it to recognize mid-level qualities from audio, and predict emotion ratings from the mid-level predictions. Further, we wish to quantify the cost – in terms of loss of predictive performance – incurred by this detour. We will call this the 'cost of explainability'.

Focusing our study on a specific benchmark dataset labelled with both perceived emotional qualities and mid-level perceptual features, we first establish our basic VGG-style model architecture [16], showing that it can learn the two individual prediction tasks (mid-level from audio, emotion from mid-level) from appropriate ground-truth data, with accuracies that are at least on par with previously published models. We then present a network with nearly identical architecture that learns to predict emotional characteristics of a piece by explicitly going through a mid-level feature prediction layer. We compare this to predicting emotion directly, using an identical network with the exception of the mid-level layer, and find that the cost of going through the mid-level features is surprisingly low, on average. Finally, we show that by training the network to learn to predict mid-level and emotions jointly, the results can be further improved. A graphical overview of the general scenario is shown in Figure 1.

The fact that in our model, emotions are predicted from the mid-level by a single fully-connected layer, allows us to measure the effects of each of the features on each emotion prediction, providing the basis for interpretability and simple explanations. There are a number of application scenarios in which we believe this could be useful; some of these will be briefly discussed in the final section.

The remainder of this paper is structured as follows: Section 2 briefly discusses related work on which this research is based. In Section 4 the datasets that provide us with emotion and mid-level annotations are described. The three different approaches to modelling emotion are summarized in Section 5. Experimental results and a demonstration of interpretability are given in Sections 6 and 7.

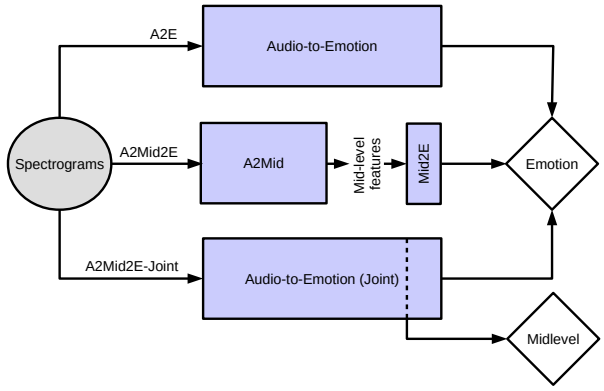


Figure 1: Three different architectures are compared for predicting emotion from audio.

We discuss our findings and conclude in Section 8.

2. RELATED WORK

In the MIR field, audio-based music emotion recognition (MER) has traditionally been done by extracting selected features from the audio and predicting emotion based on subsequent processing of these features [7]. Methods such as linear regression, regression trees, support vector regression, and variants have been used for prediction as mentioned in the systematic evaluation study by Huq et al [6]. Techniques using regression-like algorithms have generally focused on predicting arousal and valence as per the well-known Russell's *circumplex model* of emotion [15]. Deep learning methods have also been employed for predicting arousal and valence, for example [18], that investigated BLSTM-RNNs in tandem with other methods, and [3], that used LSTM-RNNs. Others such as [12] and [9] use support vector classification to predict the emotion class. Aljanaki et al. [2] provide a summary of entries to the MediaEval emotion characterization challenge and quote results for arousal and valence prediction.

Deep neural networks are preferable for many tasks due to their high performance but can be considered black boxes due to their non-linear and nested structure. While in some fields such as healthcare or criminal justice the use of predictive analytics can have life-affecting consequences [14], the decisions of MIR models are generally not as severe. Nevertheless, also in MIR it would be desirable to be able to obtain explanations for the decisions of a music recommendation or search system, for various reasons (see also Section 8). Many current methods for obtaining insights into deep network-based audio classification systems do not explain the predictions in a human understandable way but rather design special filters that can be visualized [13], or analyze neuron activations [8]. To the best of our knowledge, [19] is the only attempt to build an interpretable model for MER. They performed the task of feature extraction and selection and built models from different model classes on top of them. The only interpretation offered is the reporting of coefficients from their logistic regression models, without further explanation.

3. MID-LEVEL PERCEPTUAL FEATURES

The notion of *('mid-level') perceptual features* for characterizing music recordings has been put forward by several authors, as an alternative to purely sound-based or statistical low-level features (e.g., MFCCs, ZCR, spectral centroid) or more abstract music-theoretic concepts (e.g., meter, harmony). The idea is that they should represent musical characteristics that are easily perceived and recognized by most listeners, without any music-theoretical training. Research on such features has quite a history in the fields of music cognition and psychology (see [5] for a compact discussion). Such features are attractive for our purposes because they could provide the basis for intuitive explanations of a MIR system's decisions, relating as they do to the musical experience of most listeners.

Various sets of such perceptual features have been proposed in the literature. For instance, Friberg et al.'s set [5] contains such concepts as speed, rhythmic clarity/complexity, articulation, dynamics, modality, overall pitch hight, etc. In our study, we will be using the seven mid-level features defined by Aljanaki & Soleymani [1], because they come with an openly available set of annotated audios (see below). We recapitulate the features and their definitions in Table 1, for convenience.

4. DATASETS

For our experiments, we need music recordings annotated both with mid-level perceptual features, and with human ratings along some well-defined emotion categories. Our starting point is Aljanaki & Soleymani's *Mid-level Perceptual Features* dataset [1], which provides mid-level feature annotations. For the actual emotion prediction experiments, we then use the *Soundtracks* dataset, which is contained in the Aljanaki collection as a subset, and comes with numeric emotion ratings along 8 dimensions.

4.1 Mid-level Perceptual Features Dataset

The *Mid-level Perceptual Features Dataset* [1] consists of 5000 song snippets of around 15 seconds each annotated according to the seven mid-level descriptors listed in Table 1. The annotators were required to have some musical education and were selected based on passing a musical test. The ratings range from 1 to 10 and were scaled by a factor of 0.1 before being used for our experiments.

4.2 Emotion Ratings: The Soundtracks Dataset

The *Soundtracks (Stimulus Set 1)* [1] dataset, published by Eerola and Vuoskoski [4], consists of 360 excerpts from 110 movie soundtracks. The excerpts come with expert ratings for five categories following the discrete emotion model (happy, sad, tender, fearful, angry) and three categories following the dimensional model (valence, energy,

[1] https://www.jyu.fi/hytk/fi/laitokset/mutku/en/research/projects2/past-projects/coe/materials/emotion/soundtracks

Perceptual Feature	Question asked to human raters
Melodiousness	To which excerpt do you feel like singing along?
Articulation	Which has more sounds with staccato articulation?
Rhythmic Stability	Imagine marching along with the music. Which is easier to march along with?
Rhythmic Complexity	Is it difficult to repeat by tapping? Is it difficult to find the meter? Does the rhythm have many layers?
Dissonance	Which excerpt has noisier timbre? Has more dissonant intervals (tritones, seconds, etc.)?
Tonal Stability	Where is it easier to determine the tonic and key? In which excerpt are there more modulations?
Modality ('Minorness')	Imagine accompanying this song with chords. Which song would have more minor chords?

Table 1: Perceptual mid-level features as defined in [1], along with questions that were provided to human raters to help them interpret the concepts. (The ratings were collected in a pairwise comparison scenario.) In the following, we will refer to the last one (*Modality*) as *'Minorness'*, to make the core of the concept clearer.

tension). This makes it a suitable dataset for musically conveyed emotions [4]. The ratings in the dataset range from 1 to 7.83 and were scaled by a factor of 0.1 before being used for our experiments. As stated above, all the songs in this set are also contained in the Mid-level Features Dataset, so that both kinds of ground truth are available.

5. AUDIO-TO-EMOTION MODELS

In the following, we describe three different approaches to modeling emotion from audio, all based on VGG-style convolutional neural networks (CNNs). The architectures are summarized in Figure 2. For all models, we use an Adam optimizer with a learning rate of 0.0005 and a batch size of 8, and employ early stopping with a patience of 50 epochs to prevent overfitting.

In terms of preprocessing, the audio samples are first converted into 149-point spectrograms calculated on randomly selected 10-second sections of the original snippets. The audio is resampled at 22.05 kHz, with a frame size of 2048 samples and a frame rate of 31.25 frames per second, and amplitude-normalized before computing the logarithmic-scaled spectrogram. This results in input vectors of size 313×149. These spectrograms are used as inputs for the following model architectures.

5.1 A2E Scheme

The first model, which we term "A2E", is the most straightforward one. The spectrograms are fed into a VGG-style CNN to directly predict emotion values from audio. This is the leftmost path in Figure 2. This model is not interpretable due to its black box architecture and is used as a baseline and for computing the *cost of explainability* when comparing to more interpretable but possibly worse performing models.

5.2 A2Mid2E Scheme

In order to obtain a more interpretable model, an intermediate step is introduced. First, a VGG-style network is used to predict mid-level features from audio. This model is trained on the mid-level features dataset described in Section 4.1 above. Next, a linear regression model is trained to predict the 8 emotion ratings in the Soundtracks dataset from the 7 mid-level feature values that we get as an output from the mid-level predictor network. This corresponds to a fully connected layer with 7 input units and 8 outputs and linear (identity) activation function – see the middle path in Figure 2. We call this scheme "A2Mid2E". A linear model is chosen because its weights can easily be interpreted to understand the importance of each mid-level feature in predicting the emotion ratings.

5.3 A2Mid2E-Joint Scheme

In an attempt to replace the step-wise training of two separate models with a single model that, ideally, could learn an internal representation useful for both prediction tasks, while keeping the interpretability of the linear weights, we propose a third architechture, called "A2Mid2E-Joint" (rightmost path in Figure 2). This network learns to predict mid-level features and emotion ratings jointly, but still predicts the emotions directly from the mid-level via a linear layer. This is achieved by the second last layer having exactly the same number of units as there are mid-level features (7), followed by a linear output layer with 8 outputs. From this network, we extract two outputs – one from the second last layer ("mid-level layer"), and one from the last layer ("emotion layer"). We compute losses for both the outputs and optimize the combined loss (summation of both the losses).

6. EXPERIMENTS

The audio clips are preprocessed as described in Section 5 to obtain the input spectrograms. During training, one ran-

	Valence	Energy	Tension	Anger	Fear	Happy	Sad	Tender	Avg.
Mid2E (Aljanaki)	0.88	0.79	0.84	0.65	0.82	0.81	0.73	0.72	0.78
Mid2E (Ours)	0.88	0.80	0.84	0.65	0.82	0.81	0.74	0.73	0.79
A2E	0.81	0.79	0.84	0.82	0.81	0.66	0.60	0.75	0.76
A2Mid2E	0.79	0.74	0.78	0.72	0.77	0.64	0.58	0.67	0.71
A2Mid2E-Joint	0.82	0.78	0.82	0.76	0.79	0.65	0.64	0.72	0.75
$CoE_{A2Mid2E}$	0.02	0.05	0.06	0.10	0.03	0.02	0.02	0.08	0.05
$CoE_{A2Mid2E\text{-}Joint}$	-0.02	0.01	0.02	0.06	0.02	0.01	-0.04	0.03	0.01

Table 2: Summaries of the different model performances on predicting emotion. The last two rows show the "cost of explainability" (CoE), as the difference between our baseline (A2E) and the newly proposed models (A2Mid2E, A2Mid2E-Joint). A positive cost indicates a loss in performance.

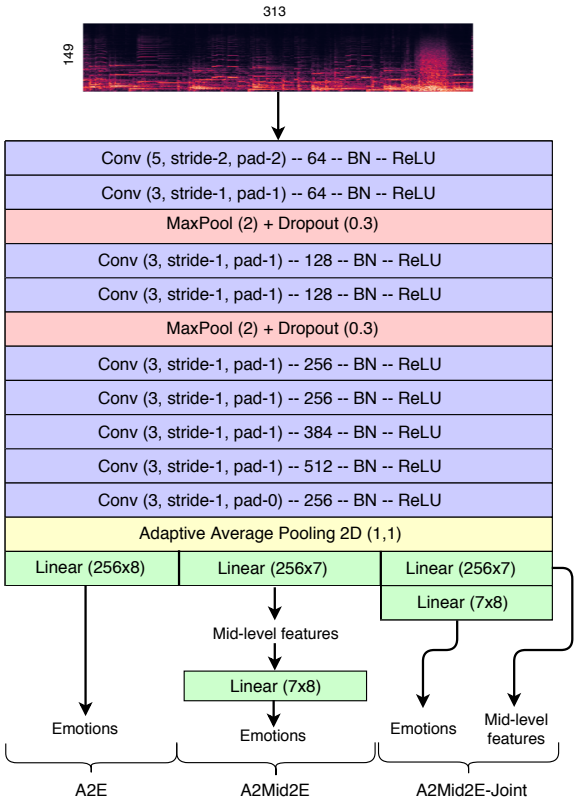


Figure 2: The same architecture from the first layer up to the 'Adaptive Average Pooling 2D' layer is shared by all networks.

dom 10-second snippet from each spectrogram is taken as input. We optimize the mean squared error, and use Pearson's correlation coefficient as the evaluation metric for emotion rating prediction. Each of the paths (A2E, A2Mid2E, A2Mid2E-Joint) is run ten times and the average correlation values are reported. Each run has a different seed which reshuffles the train-test split.

6.1 Verifying our Basic Architecture

Before going any further, we first want to verify that our VGG-style network model performs on par with comparable methods on the basic component tasks of predicting mid-level features from audio (A2Mid) and emotions from (given) mid-level features (Mid2E).

For the A2Mid scenario (Table 3), we train in two ways: first on the entire Mid-level features dataset with 8% test set selected as described in [1]. We call this A2Mid+. This is the result that should be directly compared to column 1. Second, we train only on the songs from the Soundtracks dataset with 20% test set, and call this A2Mid. The column 'Joint' in Table 3 gives the mid-level predictions produced by our A2Mid2E-Joint model. As can be seen, our models are broadly comparable to the results reported in [1], with A2Mid+ and *Joint* performing slightly better, on average.

Regarding the prediction of emotions from given mid-level feature annotations (Table 2, first two rows), there is not much space for deviation, as the models used by Aljanaki [1] and us are very simple.

Mid-level feature	Aljanaki	A2Mid+	A2Mid	Joint
Melodiousness	0.70	0.70	0.69	0.72
Articulation	0.76	0.83	0.84	0.79
R. Stability	0.46	0.39	0.39	0.34
R. Complexity	0.59	0.66	0.45	0.46
Dissonance	0.74	0.74	0.73	0.74
Tonal Stability	0.45	0.56	0.61	0.63
Minorness	0.48	0.55	0.51	0.57

Table 3: Correlation values for mid-level features predictions using our models, compared with those reported by Aljanaki et al. [1].

6.2 Quantifying the Cost of Explainability

We now compare our three model architectures (A2E, A2Mid2E, A2Mid2E-Joint) on the full task of predicting emotion from audio, We train on the Soundtracks dataset as described above, with 10 runs with randomly selected train-test 80:20 splits. The A2E model serves as reference for the subsequent models with explainable linear layers. The results can be found in Table 2.

In the case of direct emotion prediction (A2E), the final layer is connected to 256 input nodes. However, in the A2Mid2E scheme, due to the fact that we introduce a bottleneck (viz. the 7 mid-level predictions) as inputs to the subsequent linear layer predicting emotions, our hypothesis is that doing so should result in a decrease in the perfor-

mance of emotion prediction. We calculate this cost as the difference in correlation coefficients between the two models for each emotion. The results (rows A2E and A2Mid2E in Table 2) reflect the expected trend, but as can be seen, the decrease in performance is quite small (less than 7% of the original correlation coefficient on average).

6.3 Joint Learning of Mid-level and Emotions

Further improvements in the performance of the mid-level-based network can be obtained by training jointly on the mid-level and emotion annotations (as described in Section 5.3). This model reduces the cost even further, as can be seen in Table 2, row A2Mid2E-Joint. The decrease in performance is now less than 1.5% of the correlation coefficients for the A2E case on average. We believe this is acceptable in view of the possibility of obtaining explanations from this network (see below).

7. OBTAINING EXPLANATIONS

Since the mapping between mid-level features and emotions is linear in both proposed schemes (A2Mid2E, A2Mid2E-Joint), it is now straightforward to create human-understandable explanations. Linear models can be interpreted by analyzing their weights: increasing a numerical feature by one unit changes the prediction by its weight. A more meaningful analysis is to look at the *effects*, which are the weights multiplied by the actual feature values [10]. An effects plot shows the distribution, over a set of examples, of the effects of each feature on each target. Each dot in an effects plot can be seen as the amount this feature contributes (in combination with its weight) to the prediction, for a specific instance. Instances with effect values closer to 0 get a prediction closer to the intercept (bias term). Figure 3 shows the effects of the model A2Mid2E-Joint.

First we will show how this can be used to provide model-level explanations and then we will explain a specific example at the song level.

7.1 Model-level Explanation

Before a model is trained, the relationship between features and response variables can be analyzed using correlation analysis. The pairwise correlations between mid-level and emotion annotations in our data are shown in Figure 4. When we compare this to the effect plots in Figure 3, or the actual weights learned for the final linear layer (Figure 5) it can be seen that for some combinations (e.g., valence and melodiousness, happy and minorness) positive correlations go along with positive effect values and negative correlations with negative effect values, respectively. This is not a general rule, however, and there are several examples (e.g., tension and dissonance, energy and melody) where it is the other way around. The explanation for this is simple: correlations only consider one feature in isolation, while learned feature weights (and thus effects) also depend on the other features and must hence be interpreted in the overall context. Therefore it is not sufficient to look at the data in order to understand what a model has learned.

	predicted		annotated	
	#153	#322	#153	#322
valence	0.28	0.39	0.38	0.46
energy	0.37	0.50	0.37	0.54
tension	0.40	0.46	0.50	0.56
anger	0.28	0.23	0.15	0.22
fear	0.41	0.27	0.18	0.28
happy	0.17	0.21	0.17	0.17
sad	0.20	0.23	0.27	0.28
tender	0.18	0.23	0.10	0.10

Table 4: Emotion prediction profiles for the two example songs #153 and #322.

To get a better understanding, we will look at each emotion separately, using the effects plot given in Figure 3. In addition to the direction of the effect – which we can also read from the learned weights in Figure 5 (but only because all of our features are positive) – we can also see the spread of the effect which tells us more about the actual contribution the feature can have on the prediction, or how different combinations of features may produce a certain prediction.

7.2 Song-level Explanations

Effect plots also permit us to create simple example-based explanations that can be understood by a human. The feature effects of single examples can be highlighted in the effects plot in order to analyze them in more detail, and in the context of all the other predictions. To show an interesting case we picked two songs with similar emotional but different midlevel profiles. To do so we computed the pairwise euclidean distances between all songs in emotion (d_{E}) and midlevel space (d_{Mid}) separately, scaled both to the range $[0, 1]$ and combined them as $d_{\mathrm{comb}} = d_{\mathrm{E}} - (1 - d_{\mathrm{Mid}})$. We then selected the two songs from the Soundtracks dataset that maximised d_{comb}. The samples are shown in Figure 3 as a red square (song #153) and a blue dot (song #322). The reader can listen to the songs/snippets by downloading them from the Soundtracks dataset page [1] .

As can be seen from Figure 3 and from the emotion prediction profile of the two songs (see Table 4), both songs have relatively high predicted values for *tension* and *energy*, but apparently for different reasons: song #322 more strongly relies on “minorness" and “articulation" for achieving its “tense" character; on the other hand, its rhythmic stability counteracts this more strongly than in the case of song #153. The higher score on the “energy" emotion scale for #322 seems to be primarily due to its much more articulated character (which can clearly be heard: 153 is a saxophone playing a chromatic, harmonically complex line, 322 is an orchestra playing a strict, *staccato* passage).

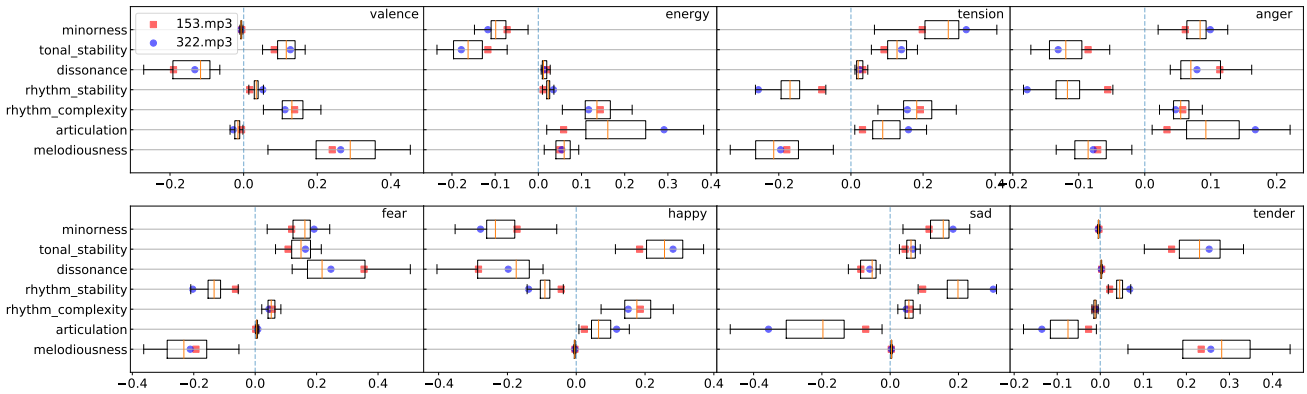


Figure 3: Effects of each mid-level feature on prediction of emotion. The boxplots show the distribution of feature effects of the model 'A2Mid2E-Joint' helping us to understand the model globally. Additionally, two example songs (blue dots, red squares) are shown to provide song-level explanations (see Section 7.2 for a discussion).

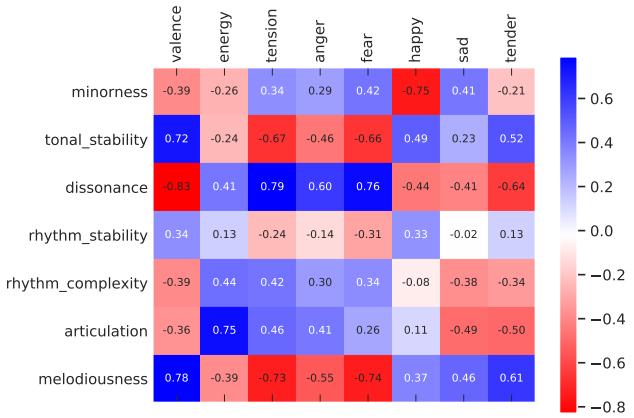


Figure 4: Pairwise correlation between mid-level and emotion annotations.

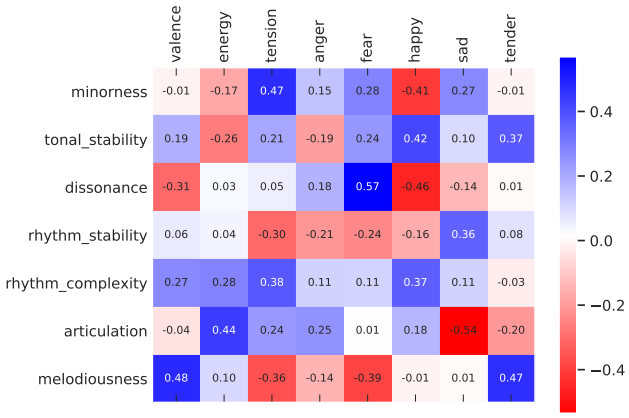


Figure 5: Weights from the linear layer of the 'A2Mid2E-Joint' model.

8. DISCUSSION AND CONCLUSION

Model interpretability and the possibility to obtain explanations for a given prediction are not ends in themselves. There are many scenarios where one may need to understand why a piece of music was recommended or placed in a certain category. Concise explanations in terms of mid-level features would be attractive, for example, in recommender systems or search engines for 'program music' for professional media producers, where mid-level qualities could also be used as additional search or preference filters [2] . As another example, think of scenarios where we want a music playlist generator to produce a music program with a certain prevalent mood, but still maintain musical variety within these limits. This could be achieved by using the mid-level features underlying the mood/emotion classifications to enforce a certain variability, by sorting or selecting the songs accordingly.

There are several obvious next steps that need to be taken in the research. The first is to extend this analysis to a larger set of diverse datasets and emotion-related dimensions. [3]

Second, we plan to extend the models and sets of perceptual features. One rather obvious (and obviously relevant) perceptual dimension that is conspicuously missing from our (Aljanaki & Soleymani's) set of mid-level features is *perceived speed* (which is not the same as tempo). Adding this intuitive musical dimension is an obvious next step towards improving our model. Of course, this will require an appropriate ground truth for training.

Generally, the relation between the space of musical qualities (such as our mid-level features) and the space of musically communicated emotions and affects deserves more detailed study. A deeper understanding of this might even give us means to control or modify emotional qualities in music by manipulating mid-level musical properties.

[2] A demonstration of mid-level explanations of emotional variability in multiple versions of songs can be found in `https://shreyanc.github.io/ismir_example.html`

[3] In fact, we do have preliminary results on a second dataset – the *MIREX-like Mood Dataset)* of [11], which is also covered by the Mid-level Perceptual Features Dataset of [1] and differs from *Soundtracks* in that is comes with discrete mood labels. The results confirm the general trends reported in the present paper, but because of the different emotion/mood encoding scheme, further optimisations on our models may still improve the results further.

9. ACKNOWLEDGMENTS

This research has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No. 670035 (project "Con Espressione").

10. REFERENCES

[1] Anna Aljanaki and Mohammad Soleymani. A data-driven approach to mid-level perceptual musical feature modeling. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, pages 615–621, 2018.

[2] Anna Aljanaki, Yi-Hsuan Yang, and Mohammad Soleymani. Developing a benchmark for emotional analysis of music. *PLOS ONE*, 12(3):1–22, 03 2017.

[3] Eduardo Coutinho, George Trigeorgis, Stefanos Zafeiriou, and Björn W Schuller. Automatically estimating emotion in music with deep long-short term memory recurrent neural networks. In *MediaEval*, 2015.

[4] Tuomas Eerola and Jonna K. Vuoskoski. A comparison of the discrete and dimensional models of emotion in music. *Psychology of Music*, 39(1):18–49, 2011.

[5] Anders Friberg, Erwin Schoonderwaldt, Anton Hedblad, Marco Fabiani, and Anders Elowsson. Using listener-based perceptual features as intermediate representations in music information retrieval. *The Journal of the Acoustical Society of America*, 136(4):1951–1963, 2014.

[6] Arefin Huq, Juan Pablo Bello, and Robert Rowe. Automated music emotion recognition: A systematic evaluation. *Journal of New Music Research*, 39(3):227–244, 2010.

[7] Youngmoo E Kim, Erik M Schmidt, Raymond Migneco, Brandon G Morton, Patrick Richardson, Jeffrey Scott, Jacquelin A Speck, and Douglas Turnbull. Music emotion recognition: A state of the art review. In *Proc. ISMIR*, pages 255–266. Citeseer, 2010.

[8] Andreas Krug, René Knaebel, and Sebastian Stober. Neuron activation profiles for interpreting convolutional speech recognition models. In *NIPS 2018 Interpretability and Robustness for Audio, Speech and Language Workshop (IRASL'18)*, 2018. to appear.

[9] Chingshun Lin, Mingyu Liu, Weiwei Hsiung, and Jhihsiang Jhang. Music emotion recognition based on two-level support vector classification. In *2016 International Conference on Machine Learning and Cybernetics (ICMLC)*, volume 1, pages 375–389. IEEE, 2016.

[10] Christoph Molnar. *Interpretable Machine Learning*. 2019. `https://christophm.github.io/interpretable-ml-book/`.

[11] Renato Panda, Ricardo Malheiro, Bruno Rocha, António Oliveira, and Rui Pedro Paiva. Multi-modal music emotion recognition: A new dataset, methodology and comparative analysis.

[12] Renato Panda, Ricardo Malheiro, Bruno Rocha, António Oliveira, and Rui Pedro Paiva. Multi-modal music emotion recognition: A new dataset, methodology and comparative analysis. In *International Symposium on Computer Music Multidisciplinary Research*, 2013.

[13] Mirco Ravanelli and Yoshua Bengio. Speaker recognition from raw waveform with sincnet. In *2018 IEEE Spoken Language Technology Workshop, SLT 2018, Athens, Greece, December 18-21, 2018*, pages 1021–1028, 2018.

[14] Cynthia Rudin and Ustun Berk. Optimized scoring systems: Toward trust in machine learning for healthcare and criminal justice. *Interfaces*, 48(5):449–466, 2018.

[15] James A Russell. A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161, 1980.

[16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[17] Gerhard Widmer. Getting closer to the essence of music: The Con Espressione Manifesto. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):19, 2017.

[18] Mingxing Xu, Xinxing Li, Haishu Xianyu, Jiashen Tian, Fanhang Meng, and Wenxiao Chen. Multi-scale approaches to the mediaeval 2015" emotion in music" task. In *MediaEval*, 2015.

[19] JiangLong Zhang, XiangLin Huang, Lifang Yang, and Liqiang Nie. Bridge the semantic gap between pop music acoustic feature and emotion: Build an interpretable model. *Neurocomputing*, 208:333 – 341, 2016. SI: BridgingSemantic.

COMMUNITY-BASED COVER SONG DETECTION

Marc Sarfati
Spotify
marcs@spotify.com

Anthony Hu
Spotify
anthonyh@spotify.com

Jonathan Donier
Spotify
jdonier@spotify.com

ABSTRACT

Audio-based cover song detection has received much attention in the MIR community in the recent years. To date, the most popular formulation of the problem has been to compare the audio signals of two tracks and to make a binary decision based on this information only. However, leveraging additional signals might be key if one wants to solve the problem at an industrial scale. In this paper, we introduce an ensemble-based method that approaches the problem from a many-to-many perspective. Instead of considering pairs of tracks in isolation, we consider larger sets of potential versions for a given composition, and create and exploit the graph of relationships between these tracks. We show that this can result in a significant improvement in performance, in particular when the number of existing versions of a given composition is large.

1. INTRODUCTION

With the rise of online streaming services, it is becoming easier for artists to share their music with the rest of the world. With catalogs that can reach up to tens of millions of tracks, one of the rising challenges faced by music streaming companies is to assimilate ever-better knowledge of their content – a key requirement for enhancing user and artist experience. From a musical perspective, one highly interesting aspect is the detection of composition similarities between tracks, often known as the *cover song* detection problem. This is, however, a very challenging problem from a content analysis point of view, as artists can make their own version of a composition by modifying any number of ingredients – instruments, harmonies, melody, rhythm, structure, timbre, vocals, lyrics, among others.

Over the years, it has become customary in the Music Information Retrieval (MIR) literature to address the cover song detection problem in what is arguably the most challenging setting. Indeed, most papers attempt to detect composition relationships between pairs of tracks based on their two audio signals only – in other words, completely out of context and without using any metadata information. While this well-defined task makes sense from an academic perspective, it might not be the optimal approach for solving the problem at an industrial scale [4].

The second starting point of our work is the fact, often mentioned in cognitive science, that commonly observed patterns are represented and stored in a redundant fashion in the human brain, which makes them more likely to be retrieved, recognised and identified than patterns that are observed less frequently [14]. If true, this would apply to our assessment of composition similarities as well. The main idea behind our work is that the corpus of existing versions of a composition can be precisely a substitute for these multiple representations.

Following these guiding intuitions, we turn to a new use case, where we do not just have pairs but a pool of *candidates* that are likely to be instances of some given musical work (according *e.g.* to some first metadata analysis). We then compare these candidates not only to *one* reference version (*e.g.* the original track, if it exists) but also to other candidate versions. We then build a graph of all these versions to identify composition clusters. Sometimes, when hundreds or thousands of versions of a given work exist (which is quite common in the catalogue of a streaming company), this ensemble-based approach can result in substantial improvements on the cover detection task.

In Section 2 we present a review of the literature on cover identification. In Section 3, we present the 1-vs-1 cover identification algorithm that we use throughout the paper, which is heavily based on [23]. The main contribution of this paper lies in Section 4, in which we present the new use case for cover identification described in the previous paragraph.We then showcase our method with examples in Section 5 and discuss some challenges in Section 6.

2. RELATED WORK

A number of possible approaches to cover song identification [12, 15] have been developed in the last decade. The authors of [9] introduced a first solution to this problem which has been used as a starting point for many subsequent studies. The main idea is to extract a list of beat-synchronous [6] chroma features from two input tracks and quantify their similarity by applying dynamic programming algorithms to a cross-similarity matrix derived from these features. This algorithm has been refined in [8] by adding a few modifications such as tempo biasing. Harmonic Pitch Class Profile (HPCP) features (chroma features) have proven very useful in cover identification [9, 18–20] as they capture meaningful musical information for composition. Other features have subsequently been introduced, such as self-similarity matrices of MFCC features [23, 24]. To take advantage

of the complementary properties of different types of features, [23] further introduced a method to combine several audio features by fusing the associated cross-similarity matrices, which resulted in a significant increase in performance compared to single-feature approaches. Having extracted audio features from two tracks to be compared, most methods use dynamic programming (either Dynamic Time Warping or the Smith-Waterman algorithm [22]) to assign a score to the pair [9, 23, 24]. One drawback of these methods is that they are computationally expensive and cannot be run at scale. Scalable solutions have been developed by mapping audio features to smaller latent spaces. For instance, [1, 13] use Principal Component Analysis (PCA) to compute a condensed representation of audio features which they use to perform a large-scale similarity search (*e.g.* a nearest neighbor search). In the same vein, [10, 17] use deep neural networks to learn low-dimensional representations of chroma features.A smaller number of papers take an ensemble-based approach and report increases in performance. In particular, [3, 23] leverage the network of songs using Similarity Network Fusion thereby fusing scores obtained via different methods, while [21] investigate several clustering methods and report in particular that the original song tends to be central within their communities.

3. PAIRWISE MATCHING

As mentioned above, our ensemble-based cover identification method consists of two steps. For a given work, we proceed to: ***(i)*** a pairwise (1-vs-1) comparison of all the tracks in a pool of potential candidates, ***(ii)*** a clustering of these candidates based on the results of step (i). In this section we present the 1-vs-1 cover song identification algorithm (i) which will be used as a starting point for our ensemble-based approach, and evaluate its performance on two distinct cover datasets.

3.1 The algorithm

For the purposes of this work, any 1-vs-1 similarity measure could be used for step (i), as we are mainly interested in quantifying the impact of step (ii) on the overall performance. We have chosen to rely on an implementation of the algorithm introduced in [23], as the algorithm achieves the best results to date on the *Covers80* [7] and *MSD* (*Covers1000*) datasets [2]. For a high-level overview of the pipeline, please refer to Figure 2 in [23]. As with most algorithms presented in Section 2, it can be decomposed into two stages: first, it extracts a list of meaningful audio features from the two tracks to be compared, then it computes a similarity score based on these. The details of this method are not directly relevant to our work, so we will focus here on a quantitative assessment of its performance, to give the reader a quantitative idea of our starting point. More details on the algorithm can be found in [23].

3.2 Quantitative evaluation of the 1-vs-1 method

We evaluate our implementation of [23] on two different datasets, and compare it with the numbers reported in the original paper as well as with a publicly available implementation of [23] by its author. [1] To make the comparison more interpretable, we evaluate two versions of our implementation with two sets of parameters: *Params1* mimics the parameters used in [23], and should therefore produce numbers that very similar to those described in the original paper, while *Params2* uses shorter 8-beats-long blocks.

We first compare the algorithms on the widely used *Covers80* dataset [7] to enable comparison with other published methods. The dataset is composed of 160 tracks that are divided into two sets (A and B) of 80 tracks each, with every track in set A matching one (and only one) track in set B. For each of the 160 tracks, we compute its score with all the other 159 tracks and report the rank of its true match. Table 1 reports the Mean Rank (MR) of the true match (1 is best), the Mean Reciprocal Rank (MRR) [5], as well as the Recall@1 (R@1) and Recall@10 (R@10). We also compute the so-called *Covers80 scores* by querying each track in set A against all the tracks in set B and reporting the number of matches found with rank 1. [2] Overall, our results are close to the ones reported in [23] – even though we could not quite reach the numbers given in their paper.

	Covers80					**Internal Dataset**	
	MR	MRR	R@1	R@10	*Covers80* score	Recall	Recall (no Jazz)
[23] paper	**7.8**	**0.85**	**82.4%**	89.9%	**68/80**	-	-
[23] code	8.6	0.77	71.7%	**91.2%**	62/80	73.2%	81.8%
Params1	10.5	0.81	78.6%	85.5%	64/80	79.2%	87.8%
Params2	13.2	0.75	73.0%	80.5%	60/80	**85.8%**	**95.1%**

Table 1: Comparison of our implementations (*Params1* and *Params2*) against the implementations of [23], on the *Covers80* dataset and on our internal dataset. The recall rates for the internal dataset correspond to a false positive rate of 0.5%. For each column, the best performance is printed in bold.

To complement this baseline, we have created an internal dataset of 452 pairs of covers grouped into several categories, obtained by metadata filtering based on the keywords *Acoustic Cover, Instrumental Cover, Karaoke, Live, Remix, Tribute* as well as some *Classical* and *Jazz* covers. Such granularity allows us to compare the performance of our algorithm across genres and cover types, providing a new perspective on the problem, as shown in Table 2. We have tested the two versions of our algorithm on the 452 positive pairs and 10,000 negative pairs selected uniformly at random. We selected the classification threshold to ensure a very low false positive rate below 0.5%. Results are presented in Table 1. Our algorithm reaches 85.8% recall, versus 73.2% for the publicly available implementation of [23] [3] . Note that jazz is the most challenging genre to

[1] https://github.com/ctralie/GeometricCoverSongs

[2] Each track from set A is now queried against the 80 tracks from set B, instead of all other 159 tracks.

[3] As the computational time is much higher for this algorithm, we only computed the false positive rate using 500 negative pairs.

detect, as jazz covers include a lot of improvisation that can be structurally different from their parent track (see Table 2). If we remove jazz covers from the dataset, the recall increases to 95.1% with the *Params2* implementation.

Type	Acoustic	Instr.	Karaoke	Live	Remix	Tribute	Classical	Jazz
# of pairs	57	63	46	57	31	53	77	68
Recall	94%	84 %	97%	100%	93 %	96 %	100 %	35 %

Table 2: Recall rates for each genre in our internal dataset, with a $< 0.5\%$ false positive rate.

In view of these results, we will use our own implementation with *Params2* throughout the rest of this paper, as it is faster and performs best on our internal dataset, which is larger and more diverse than *Cover80*.

3.3 Distributions of scores

Figure 1 presents the histogram of pairwise scores for all the positive and negative pairs in our internal dataset. The distribution of scores for the negative pairs is short-tailed and tightly concentrated around $s = 2$. This means that above $s \simeq 5$, all the pairs can be matched with high confidence. The distribution of scores for the positive pairs is much wider. As we can see from the histogram, a non-negligible fraction of these pairs lies below the classification threshold (dashed vertical line) and thus cannot be detected with this 1-vs-1 method. The purpose of the next section will be to apply an ensemble method to a pool of candidate versions of a given work, to bring these undetected candidates above the threshold by exploiting the many-to-many relationships between the candidates.

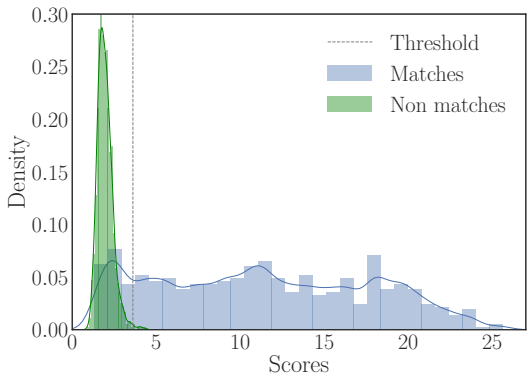
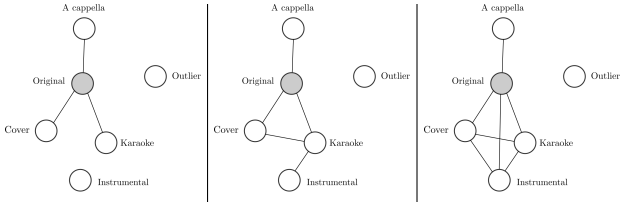


Figure 1: Histogram of the scores for positive *(blue)* and negative *(green)* pairs on our internal dataset. The threshold corresponds to the threshold used for Table 2.

4. ENSEMBLE ANALYSIS

While the 1-vs-1 algorithm we presented in Section 3 gives satisfying results overall, it still struggles on covers that are significantly different from their original track. Here we show how analyzing a large pool of candidate covers for one given reference track can improve the quality of the matching. The intuition behind this idea is that a cover version can match the reference track poorly, but match another intermediate version which is closer to the reference. For instance, an acoustic cover can be difficult to detect on a 1-vs-1 basis, but might match a karaoke version which itself strongly matches the reference track. We therefore turn to a new use case, where we not only compare single pairs (*e.g.* one reference track against one possible cover), but instead start from a pool of candidates that are all likely to be instances of some given composition (or *work*). Usually, this pool corresponds to candidates that have been pre-filtered according to some non-audio related signal, *e.g.* their title, and might comprise up to a few thousands candidates, depending on the popularity of the work and the specificity of the pre-filtering step.



(a) Computing scores versus the reference track

(b) Computing all pairwise scores

(c) Final graph

Figure 2: Direct (a) vs. ensemble-based approach (b)-(c).

4.1 Computing all pairwise scores

Given a set of N candidate versions of a work, we first compare all possible pairs of candidates within the set, resulting in $\frac{N(N-1)}{2}$ distinct scores $\{s_{ij}\}_{1 \leq i < j \leq N}$. As mentioned above, if the candidates have been pre-filtered using some metadata-matching algorithm, N typically varies from a few dozen to a few thousand candidates.

4.2 Scores to distances

Figure 1 shows that almost all negative pairs have scores between 0 and 4 while scores above 8 always correspond to positives. Scores above 8 should thus indicate a high probability of a true match regardless of the score, while a variation in score around 4 should have a significant impact on that probability. To account for this fact, we convert our scores into more meaningful distances using a logistic function: $d_{ij} = \left(1 + e^{-\frac{s_{ij}-m}{\sigma}}\right)^{-1}$, where s_{ij} is the score associated to pair (i, j) and d_{ij} is the resulting distance. We have found that the values $\sigma = 0.5$ and $m = 4.3$ work well with the distance-collapsing algorithm introduced in the next section.

4.3 Collapsing the distances

Let $D = \{d_{ij}\}$ denote the pairwise distance matrix between all pairs of candidates (see Figure 3, top left). The idea behind the ensemble-based approach is to exploit the geometry of the data to enhance the accuracy of the classification – for example, the fact that a track can match the reference track better through intermediate tracks than

directly. We use a loose version of the Floyd-Warshall algorithm [11] to update the distances in D, such that the new distances satisfy the triangular inequality most of the time [4]. The method is presented in Algorithm 1.

Algorithm 1 Loose Floyd-Warshall

procedure COLLAPSEDISTANCES(distance matrix D)
  while D still updates **do**
    for i, j in $1..N$ **do**
      $\widetilde{D}(i,j) \leftarrow \mathbf{min}^{(2)}_{k \neq i,j} D(i,k) + D(k,j) + \eta$
      $D(i,j) \leftarrow \mathbf{min}\Big(D(i,j),\ \widetilde{D}(i,j)\Big)$

Here $\min^{(k)}(x)$ denotes the k^{th} smallest value of a vector x. Using $k > 0$ allows to gain robustness as several short paths are required to merge clusters. We have found that the algorithm is slightly more robust when imposing a penalty $\eta > 0$ for using an intermediate node, which we have set to $\eta = 0.01$ after performing a grid-search optimization.

Figure 3 shows the distance matrix before (top left) and after (top right) updating the distances using Algorithm 1, for a set of candidates versions of *Get Lucky* by Daft Punk. We can see that the updated distance matrix has a more neatly defined division between clusters of tracks. The figure shows one large cluster in which all tracks are extremely close to each other (the white area), a few smaller clusters (white blocks on the first diagonal) and a number of isolated tracks that match only themselves.

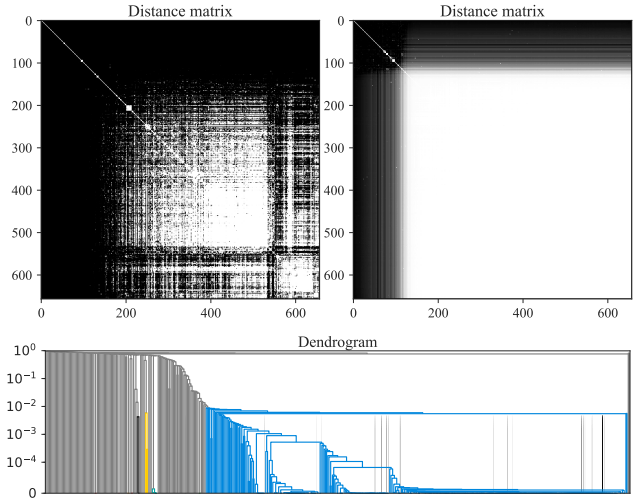


Figure 3: Top: The Floyd-Warshall algorithm applied to the distance matrix of *Get Lucky*, with (left) original distance matrix and (right) the distance matrix after applying the Floyd-Warshall algorithm. For better visual interpretation, tracks are reordered along the axes according to proximity. Darker shades correspond to larger distances. Bottom: dendrogram representation of the hierarchical clustering on the Floyd-Warshall distance matrix.

[4] The distances that would be obtained by applying the original Floyd-Washall algorithm to D would always satisfy the triangular inequality, but the resulting configuration would be very sensitive to outliers. Our method is more robust to outliers, as it requires to find more than one better path to update the distance between two points.

4.4 Hierarchical clustering

We then proceed to a clustering of the tracks using the updated distance matrix defined in 4.3, denoted D'. We use hierarchical clustering with centroid linkage [16] as we have no prior knowledge on the number of clusters in the graph. Figure 3 (bottom) shows a dendrogram representation of the hierarchical clustering applied to D'. In this example, if we apply a relatively selective threshold, we find one major cluster (colored in blue in Figure 3) that contains 97% of the true positives and no false positives. Most other clusters contain a single element, which are all the negative tracks and the remaining 3% of the positives. If we set the clustering threshold lower, then we can get more granular clusters within a same work.

4.5 Final score

In order to assign each track a final score that measures its similarity to the reference track, we use the *cophenetic distance* to the reference track that is produced by the hierarchical clustering (*i.e.* the minimum distance threshold for which they would find themselves in the same cluster as the reference track). Each track is thus assigned a final score in $0 - 100$, simply taken equal to $100 \times$ (1 - cophenetic distance), such that exact matches have a score of 100.

5. ANALYSIS OF REAL WORLD EXAMPLES

5.1 Data

We now apply the above to real world data. Our dataset consists of 10 sets of candidates that correspond to 10 works that we want to find the versions of. These 10 works span multiple genres and musical styles, including Hip Hop, R&B, Rap, Pop and Jazz. For a given work, we create the set of candidates by performing a metadata search of the given work's title on the whole Spotify catalogue, which we then annotate manually. Across the given works that we study, this produces sets of candidates whose sizes vary from a few hundred to a few thousand candidate tracks. Each set includes a reference track, which will be the anchor point for that composition. More details on the dataset can be found in Table 3.

Work	# tracks	% positives	Reference artist
Airplane	811	19%	B.o.B
Believer	2552	5%	Imagine Dragons
Blurred Lines	386	71%	Robin Thicke
Bodak Yellow	110	78%	Cardi B
Brown Sugar	721	5.8%	D'Angelo
Embraceable You	1319	94%	Sarah Vaughan
Get Lucky	657	83%	Daft Punk
Halo	2995	7.9%	Beyoncé
Heartless	1747	5.3%	Kayne West
Imagine	2044	50%	John Lennon

Table 3: The "10 works" dataset. For each work, we have selected a reference track that will be our anchor point for that composition. *Click on a work to play the reference track in the browser.*

5.2 Outline of the analysis

For each of these works, we analyze the set of candidates following the steps outlined in the previous two sections, providing us with two sets of outputs for each work: ***(a)*** the *direct score*, defined as the output of the 1-vs-1 algorithm between each candidate and the reference track, as described in Section 3 (rescaled between 0 and 100); ***(b)*** the *ensemble-based score*, produced by the method described in Section 4 (also between 0 and 100).

In the next section we start by quantitatively evaluating our ensemble-based approach (b) against the direct approach (a), before turning to some qualitative examples.

5.3 Quantitative results

We define two different metrics to evaluate the direct and the ensemble-based methods:

Ranking metric: For each work, we pick the value of the threshold that minimizes the number of classification errors, and report the number of errors. We call this a *ranking metric* as the number of errors is minimized when positives and negatives are perfectly ranked, regardless of their scores. We also report the corresponding recall and false positive rates for this threshold.

Classification metric: We fix a universal classification threshold and compute the corresponding number of classification errors.

Work	Best thr.	Ranking errors - direct					
		False negatives		False positives		Both	
		Abs.	Rel.	Abs.	Rel.	Abs.	Rel.
Airplane	12.1	33	21.9%	1	0.2%	34	4.2%
Believer	18.2	6	5.2%	0	0.0%	6	0.2%
Blurred Lines	10.1	19	7.0%	9	8.3%	28	7.3%
Bodak Yellow	6.1	6	7.0 %	6	33.3%	12	10.9%
Brown Sugar	12.1	2	4.8%	1	0.1%	3	0.4%
Embraceable You	4	0	0%	74	98.7 %	74	5.6 %
Get Lucky	10.1	17	3.1%	3	2.6%	20	3.0%
Halo	11.1	8	3.4%	8	0.3%	16	0.5%
Heartless	12.2	15	16.3%	2	0.1%	17	1.0%
Imagine	15.2	72	7.1%	17	1.7%	89	4.4%

(a) Direct approach.

Work	Best thr.	Ranking errors - ensemble-based					
		False negatives		False positives		Both	
		Abs.	Rel.	Abs.	Rel.	Abs.	Rel.
Airplane	70.7	4	2.6%	3	0.5%	7	0.9%
Believer	85.9	0	0.0%	0	0.0%	0	0.0%
Blurred Lines	52.5	0	0.0%	0	0.0%	0	0.0%
Bodak Yellow	29.3	9	10.5%	0	0.0%	9	8.2%
Brown Sugar	70.7	0	0.0%	1	0.1%	1	0.1%
Embraceable You	40.4	22	1.8 %	19	25.3 %	41	3.1 %
Get Lucky	78.8	5	0.9%	2	1.7%	7	1.1%
Halo	98.0	4	1.7 %	20	0.7%	24	0.8%
Heartless	83.8	8	8.7%	1	0.1%	9	0.5%
Imagine	96.0	1	0.1%	5	0.5%	6	0.3%

(b) Ensemble-based approach.

Table 4: Optimal thresholds and corresponding results for the ranking metric.

Table 4 shows the results for the ranking metric for each work in our dataset. For the optimal thresholds, we report the number of false negatives, false positives and the sum of both (*i.e.* the total number of classification errors). We also compute the corresponding false negative rate, false positive rate and total error rate.

Table 4a shows the ranking results for the direct approach. Interestingly, the number of false negatives tends to be higher than the number of false positives. [5] This is in line with the histogram in Figure 1, which shows a short-tailed distribution for the negatives and a wider distribution for the positives. Overall, the error rate lies between $0 - 10\%$, corresponding to a recall rate between 80% and 97% and a false positive rate below 10% (except for *Bodak Yellow* and *Embraceable You* which have a very small number of negatives to begin with – the latter case is in fact degenerate as nearly all tracks are classified as matching).

Table 4b shows the ranking results for our ensemble-based approach. The number of ranking errors is substantially lower than for the direct approach, including both the number of false positives and false negatives, as the total error rate goes down below 1% in most cases. Again, the main exception is *Bodak Yellow*, which has the smallest number of candidates. [6] *Embraceable You* is the second most challenging work, but remarkably its threshold is no longer degenerate, meaning that the method has now found a way to separate the candidates. Notably, the number of false negatives no longer outnumbers the number of false positives: the ensemble-based approach has successfully caught most of the difficult tracks that poorly matched the reference track. Among the few tracks that are still missed, several are actually very close to the threshold, and only a handful are still completely undetected (cf Table 7).

Table 5 shows the results for the classification metric. The universal threshold for each approach is defined as the median of the optimal thresholds obtained in the ranking experiment above. Again, we report the number of false negatives, the number of false positives and the sum of both. We also compute the corresponding false negative rate, false positive rate and total error rate. Here again, the results of the ensemble-based approach are overall superior to the direct approach, mostly due to an increase in recall. Although Table 5a is quite similar to Table 4a, which is a sign that the threshold on direct scores can be chosen in a nearly universal way, Table 5b differs considerably from Table 4b for some specific works (namely *Halo*, *Imagine* and *Believer*). This happens as the optimal threshold is significantly higher on these works (often $> 95\%$), letting a large number of false positives above the 78.8% threshold.

5.4 Examples

For each work, we can identify the cases where the ensemble-based approach has allowed us to detect previously undetected tracks, and trace back the optimal path that joined the reference track and the newly found track. Table 6 shows a few examples of such paths for various works. For each example, the reference track is shown at

[5] *Embraceable You* is an exception, as its threshold is degenerate and all tracks are classified as matching.

[6] It was also a genuinely difficult example and we struggled to annotate it.

Work	Thr.	Classification errors - direct					
		False negatives		False positives		Both	
		Abs.	Rel.	Abs.	Rel.	Abs.	Rel.
Airplane	12.1	33	21.9%	1	0.2%	34	4.2%
Believer	12.1	4	3.5%	91	3.7%	95	3.7%
Blurred Lines	12.1	28	10.3%	0	0%	28	7.3%
Bodak Yellow	12.1	49	57 %	0	0%	49	44.5%
Brown Sugar	12.1	2	4.8%	1	0.1%	3	0.4%
Embraceable You	12.1	753	60.5 %	3	4 %	756	57.3 %
Get Lucky	12.1	23	4.2%	1	0.9%	24	3.7%
Halo	12.1	12	5.1%	4	0.1%	16	0.5%
Heartless	12.1	15	16.3%	2	0.1%	17	1.0%
Imagine	12.1	49	4.8%	94	9.3%	143	7%

(a) Direct approach.

Work	Thr.	Classification errors - ensemble-based					
		False negatives		False positives		Both	
		Abs.	Rel.	Abs.	Rel.	Abs.	Rel.
Airplane	78.8	9	6.0%	1	0.2%	10	1.2%
Believer	78.8	0	0.0%	27	1.1%	27	1.1%
Blurred Lines	78.8	2	0.7%	0	0.0%	2	0.5%
Bodak Yellow	78.8	19	22.1%	0	0.0%	19	17.3%
Brown Sugar	78.8	2	4.8%	1	0.1%	3	0.4%
Embraceable You	78.8	70	5.6 %	1	1.3 %	71	5.4 %
Get Lucky	78.8	5	0.9%	2	1.7%	7	1.1%
Halo	78.8	0	0 %	163	5.9%	163	5.4%
Heartless	78.8	7	7.6%	4	0.2%	11	0.6%
Imagine	78.8	0	0%	158	15.6%	158	7.7%

(b) Ensemble-based approach.

Table 5: Universal threshold and corresponding results for the classification metric.

Work	Depth	Main artist (& link)	Direct scores	Ensemble-based scores
Imagine	0	John Lennon	**100**	**100**
	1	Classic Gold Hits	**60.0**	**99.99**
	2	A Perfect Circle	**21.6**	**97.9**
	3	Yoga Pop Ups	8.6	**97.9**
Heartless	0	Kanye West	**100**	**100**
	1	The Fray	**34.1**	**99.85**
	2	William Fitzsimmons	9.5	**90.7**
Get Lucky	0	Daft Punk	**100**	**100**
	1	Samantha Sax	**40.4**	**99.95**
	2	Dallas String Quartet	6.9	**86.6**
Halo	0	Beyonce	**100**	**100**
	1	LP	**27.6**	**99.96**
	2	Dion Lee	7.96	**99.16**
Halo	0	Beyonce	**100**	**100**
	1	Karaoke Universe	**20.5**	**99.96**
Halo Halo	2	Fajters	7.45	**99.35**

Table 6: Some examples of tracks that are undetected by the direct approach and captured by the ensemble-based approach, in the ranking experiment. The scores that are above the detection thresholds for each method are displayed in bold (the corresponding detection thresholds can be found in Table 4). *Click on an artist to play in the browser.*

the top of the cell (depth 0), and the newly found track at the bottom of the cell (depth > 1), with the intermediate tracks that allowed to bridge the gap in between. All the examples are true positives, except for the last example (*Halo Halo* by Fajters), which has been erroneously matched to a karaoke version of the reference track.

What about the tracks that are still undetected? Table 7 shows examples of tracks that are still undetected by our ensemble-based approach for a couple of works. No clear pattern emerges – apart from the fact that they are often in a very different musical style from the original.

Work	Main artist - Title	Direct score	Ensemble -based score
Get Lucky	The Getup - *Get Lucky*	6.4	24.9
Halo	Amanda Sense - *Halo*	12.4	94.3
Imagine	Dena De Rose - *Imagine*	10.4	86.2
Embraceable You	Earl Hines - *Embraceable You*	9.6	36.3
	Samina - *Embraceable You*	5.8	17.0
Heartless	Bright Light - *Heartless*	11.9	50.3
	Rains - *Heartless*	6.4	47.7
Bodak Yellow	Josh Vietti - *Bodak Yellow*	5.8	14.2
	J-Que Beenz - *Bodak Yellow*	5.6	13.0
Airplanes	Em Fresh - *Airplanes*	5.5	66.1
	Lisa Scinta - *Airplanes*	9.6	55.0

Table 7: Some example of tracks that are undetected by the ensemble-based approach in the ranking experiment, with their scores for both methods. *Click on a title to play in the browser.*

6. DISCUSSION

One main challenge associated with our ensemble-based approach is how to correctly handle transitivity. This issue emerges from the fact that compositions are not mutually exclusive. For example, a medley might constitute a bridge between two distinct composition groups, which our algorithm would then merge together (which is undesirable). There are probably at least two ways around this issue: one is metadata-based (*i.e.* identify these potential outliers from the metadata and exclude them from the graph computation), while another is to detect them directly from the graph structure (identify bridges between otherwise unrelated clusters).

7. CONCLUSION

In this paper, we consider the following formulation of the cover song identification problem: among a pool of candidates that are likely to match one given reference track, find the actual positives. We have introduced a two-step approach, with a first step that computes pairwise similarities between every pair of tracks in the pool of candidates (for which any known 1-vs-1 approach can be used), and a second ensemble-based step that exploits the relationships between all the candidates to output final results. We have shown that this second step can significantly improve the performance compared to a pure 1-vs-1 approach, in particular on the ranking task, where the error rate is down from a few percents to less than 1% in general. The classification task is naturally more challenging as the optimal threshold might vary from work to work, suggesting that the method would be best exploited as a complement to human annotations – where the human's task would mainly be to find the optimal threshold for the classification. Automating this last step turned out to be non-trivial and is left for future work.

8. REFERENCES

[1] Thierry Bertin-Mahieux and Daniel PW Ellis. Large-scale cover song recognition using the 2d fourier transform magnitude. In *International Conference on Music Information Retrieval (ISMIR)*, pages 241–246, 2012.

[2] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *International Conference on Music Information Retrieval (ISMIR)*, 2012.

[3] Ning Chen, Wei Li, and Haidong Xiao. Fusing similarity functions for cover song identification. *Multimedia Tools and Applications*, 77(2):2629–2652, 2018.

[4] Albin Andrew Correya, Romain Hennequin, and Mickaël Arcos. Large-scale cover song detection in digital music libraries using metadata, lyrics and audio features. *arXiv preprint arXiv:1808.10351*, 2018.

[5] Nick Craswell. Mean reciprocal rank. In *Encyclopedia of Database Systems*, pages 1703–1703. Springer, 2009.

[6] Simon Durand, Juan Pablo Bello, Bertrand David, Gaël Richard, Simon Durand, Juan Pablo Bello, Bertrand David, Gael Richard, Bertrand David, Gael Richard, et al. Robust downbeat tracking using an ensemble of convolutional networks. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 25(1):76–89, 2017.

[7] Daniel PW Ellis. The "Covers80" cover song data set. *URL: http://labrosa. ee. columbia. edu/projects/coversongs/covers80*, 2007.

[8] Daniel PW Ellis and C Cotton. The 2007 labrosa cover song detection system. *Music Information Retrieval Evaluation eXchange (MIREX)*, 2007.

[9] Daniel PW Ellis and Graham E Poliner. Identifying cover songs with chroma features and dynamic programming beat tracking. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 4, pages 1429–1432. IEEE, 2007.

[10] Jiunn-Tsair Fang, Chi-Ting Day, and Pao-Chi Chang. Deep feature learning for cover song identification. *Multimedia Tools and Applications*, 76(22):23225–23238, 2017.

[11] Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[12] Peter Grosche, Meinard Müller, and Joan Serrà. Audio content-based music retrieval. In *Dagstuhl Follow-Ups*, volume 3. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2012.

[13] Eric J Humphrey, Oriol Nieto, and Juan Pablo Bello. Data driven and discriminative projections for large-scale cover song identification. In *International Society for Music Information Retrieval (ISMIR)*, pages 149–154, 2013.

[14] Ray Kurzweil. *How to create a mind: The secret of human thought revealed*. Penguin, 2013.

[15] Meinard Müller. *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015.

[16] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.

[17] Xiaoyu Qi, Deshun Yang, and Xiaoou Chen. Audio feature learning with triplet-based embedding network. In *AAAI*, pages 4979–4980, 2017.

[18] Suman Ravuri and Daniel PW Ellis. Cover song detection: from high scores to general classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 65–68. IEEE, 2010.

[19] Joan Serra, Emilia Gómez, and Perfecto Herrera. Transposing chroma representations to a common key. In *IEEE CS Conference on The Use of Symbols to Represent Music and Multimedia Objects*, pages 45–48, 2008.

[20] Joan Serra, Emilia Gómez, Perfecto Herrera, and Xavier Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, 16(6):1138–1151, 2008.

[21] Joan Serrà, Massimiliano Zanin, Perfecto Herrera, and Xavier Serra. Characterization and exploitation of community structure in cover song networks. *Pattern Recognition Letters*, 33(9):1032–1041, 2012.

[22] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

[23] Christopher J Tralie. Early MFCC and HPCP fusion for robust cover song identification. In *International Society for Music Information Retrieval Conference (ISMIR)*, page 294–301, 2017.

[24] Christopher J Tralie and Paul Bendich. Cover song identification with timbral shape sequences. In *International Society for Music Information Retrieval Conference (ISMIR)*, page 38–44, 2015.

TRACKING BEATS AND MICROTIMING IN AFRO-LATIN AMERICAN MUSIC USING CONDITIONAL RANDOM FIELDS AND DEEP LEARNING

Magdalena Fuentes[1,2], **Lucas S. Maia**[2,3], **Martín Rocamora**[4], **Luiz W. P. Biscainho**[3]
Hélène C. Crayencour[1], **Slim Essid**[2], **Juan P. Bello**[5]
[1] L2S, CNRS–Univ.Paris-Sud–CentraleSupélec, France
[2] LTCI, Télécom Paris, Institut Polytechnique de Paris, France
[3] Federal University of Rio de Janeiro, Brazil
[4] Universidad de la República, Uruguay
[5] Music and Audio Research Laboratory, New York University, USA

ABSTRACT

Events in music frequently exhibit small-scale temporal deviations (microtiming), with respect to the underlying regular metrical grid. In some cases, as in music from the Afro-Latin American tradition, such deviations appear systematically, disclosing their structural importance in rhythmic and stylistic configuration. In this work we explore the idea of automatically and jointly tracking beats and microtiming in timekeeper instruments of Afro-Latin American music, in particular Brazilian *samba* and Uruguayan *candombe*. To that end, we propose a language model based on conditional random fields that integrates beat and onset likelihoods as observations. We derive those activations using deep neural networks and evaluate its performance on manually annotated data using a scheme adapted to this task. We assess our approach in controlled conditions suitable for these timekeeper instruments, and study the microtiming profiles' dependency on genre and performer, illustrating promising aspects of this technique towards a more comprehensive understanding of these music traditions.

1. INTRODUCTION

Across many different cultures, music is meter-based, i.e., it has a structured and hierarchical organization of pulsations. Within this metrical structure, the different pulsations interact with one another and produce the sensation of rhythm, inducing responses in the listeners such as foot tapping or hand clapping. In the so-called "Western" music tradition, that hierarchical structure often includes the beat and downbeat levels, where the former corresponds to the predominant perceived pulsation, and the latter has a longer time-span that groups several beats into bars. In some cases, the events in music present small-scale temporal deviations with respect to the underlying regular metrical grid, a phenomenon here referred to as microtiming. The interaction between microtiming deviations and other rhythmic dimensions contribute to what has been described as the sense of 'swing' or 'groove' [8, 9, 30]. The systematic use of these deviations is of structural importance in the rhythmic and stylistic configuration of many musical genres. This is the case of jazz [9, 10, 20, 38], Cuban *rumba* [2], Brazilian *samba* [32, 39] and Uruguayan *candombe* [22], among others. Consequently, the analysis of these music genres without considering microtiming leads to a limited understanding of their rhythm.

Samba and *candombe* are musical traditions from Brazil and Uruguay, respectively, that play a huge role in those countries' popular cultures. Both genres have deep African roots, partly evidenced by the fact that their rhythms result from the interaction of several rhythmic patterns played by large ensembles of characteristic percussive instruments. *Candombe* rhythm is structured in 4/4 meter, and is played on three types of drums of different sizes and pitches—*chico*, *repique* and *piano*—, each with a distinctive rhythmic pattern, the *chico* drum being the timekeeper. [1] *Samba* rhythm is structured in 2/4 meter, and comprises several types of instruments—*tamborim*, *pandeiro*, *chocalho*, *reco-reco*, *agogô*, and *surdo*, among others. Each instrument has a handful of distinct patterns [16], and more than one instrument may act as the timekeeper. Because of this combination of several timbres and pitches, the texture of a *samba* performance can become more complex than that of a *candombe* performance, where only three types of drums are present. Nevertheless, both rhythms have in common that they exhibit microtiming deviations at the sixteenth note level [22, 28, 32], with no deviations in the beat positions. [2] This is illustrated in Figure 1 for the recording of a *tamborim* playing in the *samba de enredo* style.

[1] In this musical context, the role of timekeeper is assigned to an instrument that plays an invariable rhythmic pattern (i.e., an *ostinato*) usually at a high rate, thus defining the subdivision of the beat.

[2] In other musical forms, such as waltz, microtiming may be mostly on beats.

1.1 Related Work

Microtiming has been studied in the context of Music Information Retrieval (MIR) for many years [2, 8, 17]. Besides the interest in characterizing microtiming for musicological studies, it is important for music synthesis applications, since it is a central component for "humanizing" computer generated performances [18]. Depending on the musical context, microtiming can take the form of tempo variations, like *rubato* or *accelerando*, or small-scale deviations of events with respect to an underlying regular metrical grid [22]. Therefore, in order to study microtiming deviations one has to know the expected position of the events within the metrical grid and the actual articulated positions—which can be inferred from information related to the onset positions, the tempo and/or the beats.

Most of the proposed methods for microtiming analysis are based on manually annotated data. Laroche et al. [27] proposed a method for the joint estimation of tempo, beat positions and swing in an ad-hoc fashion. The proposal exploits some simplifications: assuming constant tempo and swing ratio, and propagating beat positions based on the most likely position of the first beat. More recent works perform semi-automatic analysis still relying on informed tempo [9, 10], or using an external algorithm for its estimation [30]. Within the context of *candombe* and *samba*, microtiming characterization has also been addressed using either semi-automatic or heuristic methods [17, 22, 32].

In other rhythm-related MIR tasks such as beat and downbeat tracking, graphical models (GM) such as hidden Markov models or dynamic Bayesian networks are widely used [4, 19, 25, 36]. GMs are capable of encoding musical knowledge in a flexible and unified manner, providing structure to the estimations and usually a gain in performance for different models across genres [14]. In particular, Conditional Random Fields (CRFs) are discriminative undirected GMs for structured data prediction [37]. CRFs relax some conditional independence assumptions of Bayesian Networks, which allows for modeling complex and more general dependency structures, thus making them appealing for music modeling. CRFs have been applied in MIR tasks such as beat tracking [13] or audio-to-score alignment [21], and have been successfully combined with deep neural networks (DNNs) [11, 15, 24].

1.2 Our Contributions

This work takes a first step towards fully-automatic tracking of beats and microtiming deviations in a single formalism, applied to the analysis of two (usually underrepresented) Afro-Latin American music genres, namely Brazilian *samba* and Uruguayan *candombe*. More precisely, we introduce a CRF model that uses beat and onset activations derived from deep learning models as observations, and combines them to jointly track beats and microtiming profiles within rhythmic patterns at the sixteenth note level. To the best of our knowledge, this is the first work that explores the use of CRFs for tracking microtiming and beats jointly. This temporal granularity is in accordance with the type of microtiming deviations present in the music traditions under study. Following previous works [9], we derive microtiming labels from annotated onsets and use them to evaluate the proposed system, attaining promising results towards more holistic and descriptive models for rhythm analysis. We also study the usefulness of this approach in controlled conditions, as a first assessment of its capabilities. We explore our microtiming representation in some applications, namely the extraction of microtiming profiles of certain instruments, and the study of differences between musical genres based on their microtiming traits.

2. PROPOSED METHOD

2.1 Language Model

The proposed language model consists of a linear-chain CRF [26, 37]. Formally, the conditional probability of a label sequence $\mathbf{y} = (y_1, ..., y_T)$ of length T given an input sequence of observations $\mathbf{x} = (x_1, ..., x_T)$ is given by:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \psi(y_t, y_{t-1})\, \phi(y_t, x_t), \qquad (1)$$

where ψ is the transition potential and ϕ is the observation potential. They play a role similar to transition and observation probabilities in dynamic Bayesian networks or hidden Markov models, with the difference that the potentials in a CRF do not need to be proper probabilities, hence the need for the normalization factor $Z(\mathbf{x})$.

In our model, depicted in Figure 2, the output labels $\mathbf{y}$ are a function of three variables that describe the position inside the beat, the length of the beat interval in frames, and the microtiming within the beat-length pattern at the sixteenth note level. Formally,

$$y_t := (f_t, l_t, m_t), \qquad (2)$$

where f_t is the frame counter with $f_t \in \mathcal{F} = \{1, ..., l_t\}$, $l_t \in \mathcal{L} = \{l_{\min}, ..., l_{\max}\}$ is the number of frames per beat, which relates to the tempo of the piece; and the microtiming $\mathbf{m}_t \in \mathcal{M} = \{\mathbf{m}_1, ..., \mathbf{m}_N\}$. The observations $\mathbf{x}$ are based on estimated beat and onset likelihoods, as detailed later. The problem of obtaining the beat positions and microtiming profiles is then formulated as finding the sequence of labels $\mathbf{y}^*$ such that $\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$.

2.1.1 Microtiming Tracking

Both in *samba* and *candombe*, timekeeper instruments usually play a beat-length rhythmic pattern that articulates several sixteenth notes [16], as shown in Figures 1 and 3 for the *tamborim*. In order to provide a common framework for comparing both music genres, we focus our study on the microtiming deviations of beat-length rhythmic patterns articulated by timekeeper instruments in groups of four sixteenth notes. [3] To that end, we consider the following hypothesis, which we explain further below:

- The tempo is constant within a beat.

[3] Note that minor adjustments to the proposed model allow for the tracking of microtiming deviations in other kinds of rhythmic patterns.

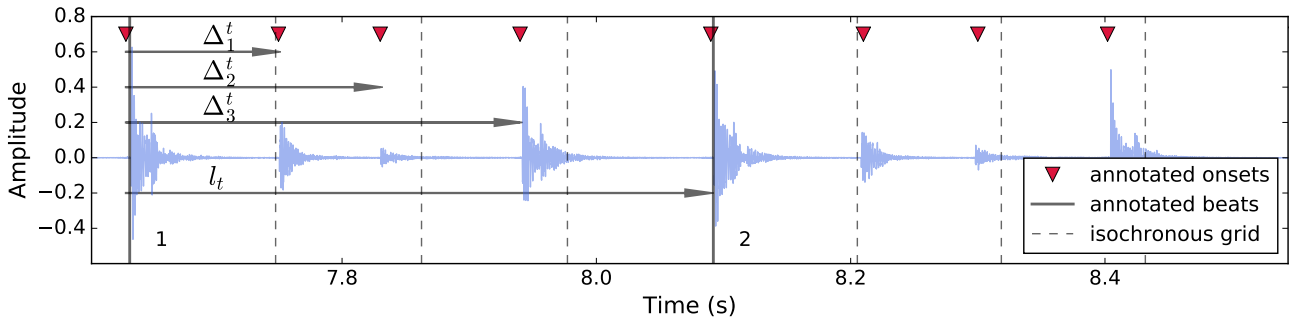


Figure 1. Example of microtiming deviations at the sixteenth note level for a beat-length rhythmic pattern from the *tamborim* in *samba de enredo*.

- The microtiming profile changes smoothly only at beat transitions.
- The tempo is between 120 and 135 BPM, to ensure an appropriate temporal resolution.

We define the microtiming descriptor $\mathbf{m}$ at frame t as:

$$\mathbf{m}_t := (m_t^1, m_t^2, m_t^3)$$

where $m_t^i := \frac{\Delta_t^i}{l_t} \in [\frac{i}{4}+\delta_L^i, \frac{i}{4}+\delta_U^i]$, and Δ_t^i is the distance in frames between an articulated sixteenth note and the beginning of the beat interval, as shown in Figure 1. Thus, each m_t^i models the position of the i-th sixteenth note with respect to the beginning of the beat, relative to the total beat length. For instance, the value of the microtiming descriptor for a rhythmic pattern of four isochronous sixteenth notes, i.e., located exactly on an equally-spaced metrical grid, is $\mathbf{m} = (0.25, 0.50, 0.75)$, indicating the articulation of events at $1/4, 1/2$ and $3/4$ of the beat interval respectively. To account for different microtiming profiles, the value of m_t^i is estimated within an interval determined by lower and upper deviations bounds, δ_L^i and δ_U^i, modeled as positive or negative percentages of the beat interval length. The proposed microtiming descriptor provides an intuitive idea of how the articulated sixteenth notes deviate within the rhythmic pattern from their isochronous expected positions. It is independent of tempo changes, since it is normalized by the estimated beat interval length, allowing for studies on microtiming–tempo dependencies.

The definition of the microtiming descriptor $\mathbf{m}_t$ can be related to the *swing-ratio*, s, proposed in previous work [9,30], though the two differ in various aspects. The *swing-ratio* is defined in terms of the inter-onset intervals (IOIs) of a long–short rhythmic pattern, such that $s \geq 1$ is the ratio between the *onbeat* IOI (longer interval) and the off-beat IOI (shorter interval). In contrast, the $\mathbf{m}_t$ descriptor is composed by three *microtiming–ratios*, m_t^i, whose IOIs are defined with respect to the beginning of the beat instead of the previous onset as in [9, 30]. However, it is possible to convert the model proposed here into the *swing-ratio* by redefining $\mathbf{m}$ as $m_s := m_t^1$, and then, from the estimated m_s, computing $s = \frac{m_t^1}{1-m_t^1}$. With such modifications, the model could be applied to the studies presented in [9, 30].

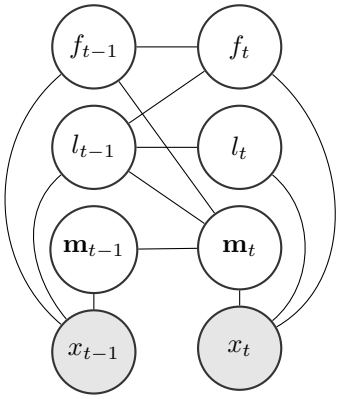


Figure 2. CRF graph. Observations and labels are indicated as gray and white nodes respectively.

2.1.2 *Transition Potential* ψ

The transition potential is given in terms of f_t, l_t, and m_t (see Equation 2) by:

$$\psi(y_t, y_{t-1}) := \psi_f(f_t, f_{t-1}, l_t, l_{t-1})\psi_m(\mathbf{m}_t, \mathbf{m}_{t-1}, f_{t-1}, l_{t-1})$$

Similar to [13,25], we force frame counter f_t to increase by one, at each step, up to the maximum beat length considered, and to switch to one at the end of the beat. Beat duration changes are unlikely (i.e., tempo changes are rare) and only allowed at the end of the beat. We constrain these changes to be smooth, giving inertia to tempo transitions. Those rules are formally expressed by:

$$\psi_f(f_t, f_{t-1}, l_t, l_{t-1}) := \begin{cases} 1 & \text{if } f_t = (f_{t-1} \bmod l_{t-1}) + 1, \\ & \quad f_{t-1} \neq l_{t-1} \\ 1 - p_f & \text{if } l_t = l_{t-1}, \\ & \quad f_t = 1,\ f_{t-1} = l_{t-1} \\ \frac{p_f}{2} & \text{if } l_t = l_{t-1} \pm 1,\ f_t = 1 \\ 0 & \text{otherwise} \end{cases}$$

The microtiming descriptor m_t changes smoothly and only at the end of the beat, that is:

$$\psi_m(\mathbf{m}_t, \mathbf{m}_{t-1}, f_{t-1}, l_{t-1}) := \begin{cases} 1 & \text{if } \mathbf{m}_t = \mathbf{m}_{t-1}, \\ & \quad f_{t-1} \neq l_{t-1} \\ 1 - p_m & \text{if } \mathbf{m}_t = \mathbf{m}_{t-1}, \\ & \quad f_{t-1} = l_{t-1} \\ \frac{p_m}{2} & \text{if } m_i^t = m_i^{t-1} \pm 0.02\ \forall i, \\ & \quad f_{t-1} = l_{t-1} \\ 0 & \text{otherwise} \end{cases}$$

In the transition potential, p_f and p_m represent the probability of changing the beat interval length (i.e., tempo) and the probability of changing the microtiming profile at the end of the beat, respectively. The values of $1-p_f$ and $p_f/2$ were chosen following previous works, whereas $1-p_m$ and $p_m/2$ were similarly set in order to make the possible microtiming transitions equally likely.

Since m_t^i is given in percentage with respect to the inter-beat-interval (IBI), the resolution with which microtiming can be estimated in the model is also percentual, and it is given by the relation between the sampling rate SR of the features and the BPM: $\mathrm{res} = \frac{\mathrm{BPM}}{60\mathrm{SR}}$. It has been shown in the literature that a resolution of 0.02 of the IBI is sufficient for representing microtiming deviations [17, 32]. To keep computational complexity low but at the same time guaranteeing a resolution $\mathrm{res} = 0.02$, we use observation features sampled at 110 Hz and we study pieces whose tempo is within range of 120 to 135 BPM. Note that these assumptions are valid in the music under study, and they could be adapted to a different music genre, e.g. increasing sampling rate to increase the BPM interval.

2.1.3 Observation Potential ϕ

The observation potential depends on the beat and onset likelihoods, the frame counter f_t and the microtiming m_t:

$$\phi(f_t, \mathbf{m}_t, x_t) := \begin{cases} b_t & \text{if } f_t = 1 \\ o_t - b_t & \text{if } \frac{f_t}{l_t} \in \mathbf{m}_t \\ 1 - o_t & \text{otherwise} \end{cases}$$

where b_t and o_t are beat and onset likelihoods, respectively. The onset likelihood was estimated using the ensemble of recurrent neural networks for onset activation estimation from *madmom* [3]—we refer the interested reader to [5, 12] for further information. We designed a simple DNN for the beat likelihood estimation and trained it on *candombe* and *samba*.[4] It consists of 6 layers, namely: batch normalization, dropout of 0.4, bidirectional gated recurrent unit (Bi-GRU) [6] with 128 units, batch normalization, another identical Bi-GRU layer, and a dense layer with two units and a softmax activation.

We use a mel-spectrogram as input feature for the DNN. The short–time Fourier transform is computed using a window length of 2048 samples and a hop of 401 samples, to ensure a sampling rate of 110 Hz with audio sampled at 44.1 kHz. We use 80 mel filters, comprising a frequency range from 30 Hz to 17 kHz.

3. DATASETS

In our experiments we use a subset of the *candombe* dataset [35] and the BRID dataset [29] of Brazilian *samba*.

candombe dataset: it comprises audio recordings of Uruguayan *candombe* drumming performances in which ensembles of three to five musicians play the three different *candombe* drums: *chico*, *piano* and *repique*. It has separated stems of the different drums, which facilitates the microtiming analysis. We focus our study on the *chico* drum, which is the timekeeper of the ensemble. We select a subset of the recordings in the dataset, in which the *chico* drum plays a beat-length pattern of four sixteenth notes, for a total of 1788 beats and 7152 onsets.

BRID dataset: it consists of both solo and ensemble performances of Brazilian *samba*, comprising ten different instrument classes: *agogô*, *caixa* (snare drum), *cuíca*, *pandeiro* (frame drum), *reco-reco*, *repique*, *shaker*, *surdo*, *tamborim* and *tantã*. We focus our study on the *tamborim*, which is one of the timekeepers of the ensemble. We select a subset of the solo tracks, in which the *tamborim* plays a beat–length rhythmic pattern of four sixteenth notes (shown in music notation[5] in Figure 3), for a total of 396 beats and 1584 onsets.

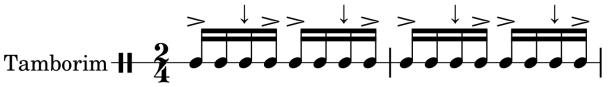


Figure 3. Example of the beat-length rhythmic pattern of the *tamborim* from the *samba* dataset in music notation.

3.1 Ground-Truth Generation

The microtiming ground-truth is inferred following the approach of [9], in which the onsets are used to derive the swing-ratio annotations. Analogously, we compute the microtiming ground-truth using the annotated onsets, obtaining one value of $\mathbf{m} = (m_1, m_2, m_3)$ for each beat. In order to mitigate the effect of onset annotation errors and sensori-motor noise, we use a moving-median filter to smooth the microtiming ground-truth, with a centered rectangular window of length 21 beats, as shown in Figure 4.

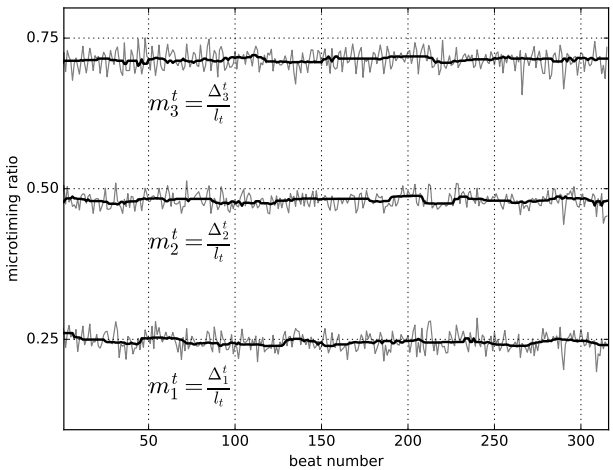


Figure 4. Example of the microtiming values for a *chico* drum recording in the *candombe* dataset. Dark and light lines represent the ground-truth with and without median filtering, respectively.

[4] The training proved necessary because the timekeeper pattern of *candombe* rhythm has a distinctive accent displaced with respect to the beat that misleads beat–tracking models trained on "Western" music [34].

[5] The symbol '>' refers to an accent, and '↓' implies to turn the *tamborim* upside down and execute the strike backwards.

4. EXPERIMENTS

4.1 Experimental Setup

We investigate the performance of the model and whether the microtiming descriptor is useful for analyzing the music at hand. To that end, we scale the *candombe* dataset to match the size of the *samba* dataset at test time by selecting excerpts in each track. We assess the model's performance using manually annotated onsets and beats, from which we derive our ground-truth as explained in Section 3.1. To evaluate if the microtiming estimation affects the beat tracking, we compare the model's performance with a simplified version of it that only tracks beats. This version has only variables f_t and l_t (see Figure 2); the same potential ψ_f is used, and the observation potential is simply b_t (the beat likelihood) at beat positions and $1 - b_t$ otherwise. We assess the microtiming estimation by: varying the p_m microtiming transition parameter—allowing smooth changes within the piece or no changes at all; and varying the tolerance used on the F-measure (F1) score. Finally, we discuss our main findings on the potential of jointly tracking beats and microtiming.

4.1.1 Implementation, Training and Evaluation Metrics

The DNN beat likelihood model is implemented in *Keras* 2.2.4 and *Tensorflow* 1.13.1 [1, 7]. We use the *Adam* optimizer [23] with default parameters. Training is stopped after 10 epochs without improvement in the validation loss, to a maximum of 100 epochs. We train the network with patches of 500 frames and a batch size of 64, leaving one track out and training with the rest, which we split in 30% and 70% for validation and training respectively, among the same genre. The onset activation was obtained with *madmom* version 0.16.1 [3], and the mel-spectra was computed using *librosa* 0.6.3 [31].

We evaluate the model using the F1 score for beat tracking with a tolerance window of 70 ms, as implemented in *mir_eval* 0.5 [33]. To evaluate the microtiming estimation, we first select the correctly estimated beats, then compute F1 for each estimated m_t^i with tolerance windows of different lengths, and the overall score as the mean F1 ($\text{F1}_{\mathbf{m}_t} = \sum_i \text{F1}_{m_t^i}/3$).

4.2 Results and Discussion

The results on the microtiming tracking are depicted in Figure 5, which shows the F1 scores as a function of the tolerance. The different colors represent the different p_m values. We evaluate the model for the set of values $p_m = \{0, 0.001, 0.06\}$, that is no, very unlikely and more likely microtiming changes respectively. Those values were obtained from statistics on the data in preliminary experiments. We searched for microtiming ratios within the interval $[0.25, 0.29] \times [0.42, 0.5] \times [0.67, 0.75]$, for microtiming dimensions $i = 1, 2, 3$ respectively. This corresponds to $\boldsymbol{\delta}_L = (0, -0.03, -0.08)$ and $\boldsymbol{\delta}_U = (0.04, 0, 0)$.[6] As illustrated in Figure 5, we found that the restriction of a constant microtiming profile ($p_m = 0$) along the piece relates to a worse performance, specially with small tolerances. We hypothesize that this occurs because in some *samba* excerpts the microtiming ratio changes are percentually bigger than those tolerances, leading to an inaccurate estimation. From considering the dependency of the F1 score with respect to the tolerance, we observe that is possible to achieve a reasonable F1 score from 0.025 on. The results with the best compromise in terms of variance and median are achieved with $p_m = 0.001$, which aligns with the hypothesis that microtiming profiles change very smoothly over time. We explored different tolerances since we are working with frames which are noisy, and the comparison with the smoothed ground truth still makes sense with large tolerances.

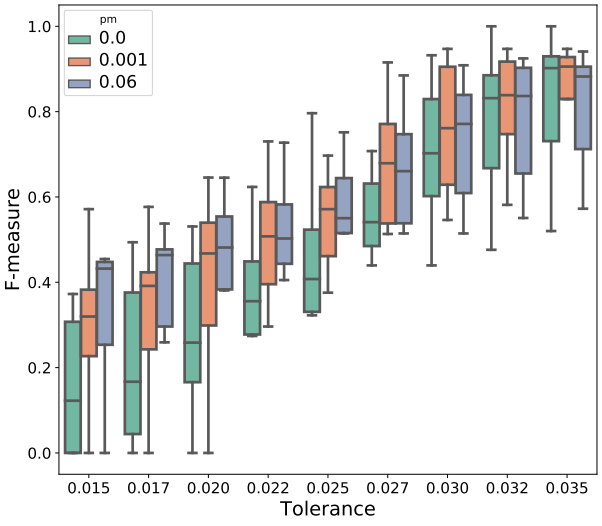


Figure 5. Mean microtiming F-measure score on the two datasets.

We found that the beat tracking performance of the model reaches a 95.7% F1 score, being equivalent to the beat tracking only version. The high F1 score in beat tracking is not surprising given that the DNN was trained using data of the same nature (acoustic conditions and genre) and the sets are homogeneous. As mentioned before, state-of-the-art beat tracking systems based on DNNs fail dramatically in this specific scenario [34], particularly tracking the beats in time-keeper instruments in *candombe*, because the data is too different from what was used in their training. We do not consider this as a challenging beat tracking case, but a training stage was needed to perform adequately.

During our experiments we observed that the microtiming descriptor $\mathbf{m}_t$ could be used to help beat tracking in some cases. Informing the microtiming profile a priori, by setting δ_L^i and δ_U^i, can disambiguate beat positions by helping the joint inference. This could allow to apply non pre-trained beat tracking models to *candombe* recordings, which usually fail in estimating the beat location by displacing it one sixteenth note (due to an accent in the rhythmic pattern). Aligned to that idea, the model could be use-

[6] Symmetric windows around the isochronus sixteenth note positions were used for the microtiming ratios in preliminary experiments with no gain in tracking performance and a higher computational burden.

ful in scenarios where onsets from other instruments are present. Besides, when the beat tracking is incorrect, the obtained microtiming profile can be descriptive of the type of mistake that occurs by contrasting the obtained profile with the expected one. The same case mentioned before—a lag of a sixteenth note in the beat estimation—shows in the microtiming estimation as unexpected forward positions in the second and third sixteenth notes, with a synchronous fourth one, which is the *candombe* microtiming profile lagged by a sixteenth note position.

Microtiming description and insights: Figure 4 illustrates an example of microtiming profile for an excerpt of the *candombe* dataset. This example shows the microtiming variations per beat interval along the complete recording. In the performance of the example, the rhythmic pattern is played with the same microtiming profile in the whole track. This microtiming template is characteristic of some patterns of *candombe* drumming [34], and it is present in several recordings of the dataset. We noticed that microtiming profiles do not present significant variations within tempo changes in the *candombe* dataset. However, the presented method can be used to characterize curves of microtiming vs. tempo that could be informative of musical phenomena for other music genres or datasets.

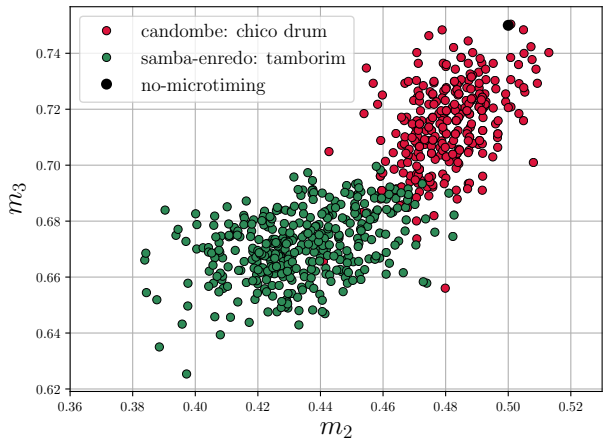


Figure 6. Microtiming distribution depending on style, view of the plane $m_t^2 m_t^3$, denoted as $m_2 m_3$ for simplicity. A dot at $(0.50, 0.75)$ indicates the expected position of a beat that shows no microtiming, that is, where all onsets are evenly spaced.

As shown in Figures 6 and 7, the microtiming descriptor $\mathbf{m}_t$ encodes musical features that are informative about the music genre, instrument type or performer. These two figures show the microtiming profile for all beats from *tamborim* and *chico* recordings, using the annotations for better visualization. Firstly, by observing the 'no-microtiming' reference in the figures that corresponds to $\mathbf{m}_t = (0.25, 0.50, 0.75)$, it becomes clear that both *samba* and *candombe* present considerable microtiming deviations in their time-keeper instruments. Even though the rhythmic patterns from both instruments present deviations that tend to compress the IOI in a similar manner, the microtiming profile differs for each music style, being more drastic in the case of the *tamborim*. This analysis should be extended to other *samba* instruments in order to determine if differences are due to the rhythmic pattern of a particular instrument; or if different patterns within the same genre tend to follow the same microtiming profile (characteristic of the genre). Figure 7 shows the microtiming profiles of each performer. It is quite clear that performers tend to be consistent with their microtiming, opening the perspective of studying microtiming profiles for performer characterization, as was done for jazz [9].

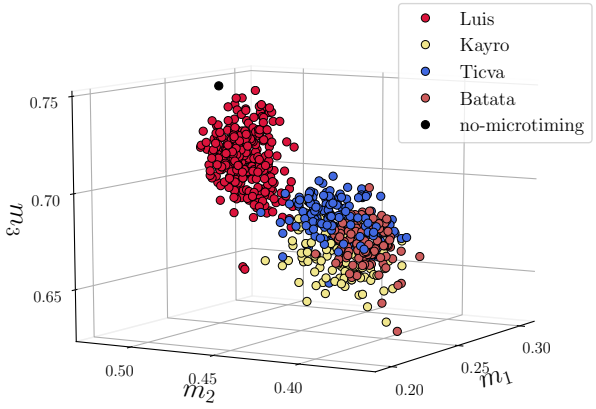


Figure 7. Microtiming distribution depending on performer (top musician plays *candombe* and the others play *samba*). A dot at $(0.25, 0.50, 0.75)$ indicates the point of no microtiming.

5. CONCLUSIONS AND FUTURE WORK

In this work, we introduced a language model that performs automatic tracking of beats and microtiming deviations in a single formalism. We applied this model to Afro-Latin American music, particularly Brazilian *samba* and Uruguayan *candombe*, and we focused our study on beat–length rhythmic patterns of timekeeper instruments, with four articulated sixteenth notes. The promising results we obtained with our method using a ground-truth derived from annotated onsets indicate it can facilitate automatic studies of these rhythms. This work intends to take a further step towards holistic systems that produce consistent and coherent estimations of music content.

As future work, we plan to extend our model to describe the microtiming profile depending on the nature of the rhythmic pattern being played, i.e., whether they articulate 2, 3, 4 or more notes, and to explore the usefulness of our model in challenging scenarios in comparison to heuristic methods.

6. ACKNOWLEDGEMENTS

This work was partially funded by CAPES, CNPq, ANII, CNRS, and STIC-AmSud program project 18-STIC-08. The authors would like to thank the reviewers for their valuable feedback.

7. REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] J. A. Bilmes. *Timing is of the essence: Perceptual and computational techniques for representing, learning, and reproducing expressive timing in percussive rhythm*. Master's thesis, Massachusetts Institute of Technology, Cambridge, USA, 1993.

[3] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer. madmom: a new python audio and music signal processing library. In *24th ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, October 2016.

[4] S. Böck, F. Krebs, and G. Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *17th Int. Society for Music Information Retrieval Conf. (ISMIR)*, pages 255–261, New York, USA, August 2016.

[5] S. Böck, A. Arzt, F. Krebs, and M. Schedl. Online real-time onset detection with recurrent neural networks. In *15th Int. Conf. on Digital Audio Effects (DAFx)*, York, UK, September 2012.

[6] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[8] M. E. P. Davies, G. Madison, P. Silva, and F. Gouyon. The effect of microtiming deviations on the perception of groove in short rhythms. *Music Perception*, 30(5):497–510, June 2013.

[9] C. Dittmar, M. Pfleiderer, S. Balke, and M. Müller. A swingogram representation for tracking micro-rhythmic variation in jazz performances. *Journal of New Music Research*, 47(2):97–113, 2018.

[10] C. Dittmar, M. Pfleiderer, and M. Müller. Automated estimation of ride cymbal swing ratios in jazz recordings. In *16th Int. Society for Music Information Retrieval Conf. (ISMIR)*, pages 271–277, Málaga, Spain, October 2015.

[11] S. Durand and S. Essid. Downbeat detection with conditional random fields and deep learned features. In *17th Int. Society for Music Information Retrieval Conf. (ISMIR)*, pages 386–392, New York, USA, August 2016.

[12] F. Eyben, S. Böck, B. Schuller, and A. Graves. Universal onset detection with bidirectional long-short term memory neural networks. In *11th Int. Society for Music Information Retrieval Conf. (ISMIR)*, pages 589–594, Utrecht, The Netherlands, August 2010.

[13] T. Fillon, C. Joder, S. Durand, and S. Essid. A conditional random field system for beat tracking. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 424–428, South Brisbane, Australia, April 2015.

[14] M. Fuentes, B. McFee, H. Crayencour, S. Essid, and J. Bello. Analysis of common design choices in deep learning systems for downbeat tracking. In *19th Int. Society for Music Information Retrieval Conf. (ISMIR)*, pages 106–112, Paris, France, September 2018.

[15] M. Fuentes, B. McFee, H. Crayencour, S. Essid, and J. Bello. A music structure informed downbeat tracking system using skip-chain conditional random fields and deep learning. In *44th Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 481–485, Brighton, UK, May 2019.

[16] G. Gonçalves and O. Costa. *The Carioca Groove: The Rio de Janeiro's Samba Schools Drum Sections*. Groove, Rio de Janeiro, Brazil, 2000.

[17] F. Gouyon. Microtiming in 'Samba de Roda' — preliminary experiments with polyphonic audio. In *11th Brazilian Symposium on Computer Music (SBCM)*, pages 197–203, São Paulo, Brazil, September 2007.

[18] H. Hennig, R. Fleischmann, A. Fredebohm, Y. Hagmayer, J. Nagler, A. Witt, F. J. Theis, and T. Geisel. The nature and perception of fluctuations in human musical rhythms. *PloS one*, 6(10):e26457, October 2011.

[19] A. Holzapfel, F. Krebs, and A. Srinivasamurthy. Tracking the 'odd': Meter inference in a culturally diverse music corpus. In *15th Int. Society for Music Information Retrieval Conf. (ISMIR)*, pages 425–430, Taipei, Taiwan, October 2014.

[20] V. Iyer. Embodied mind, situated cognition, and expressive microtiming in african-american music. *Music Perception*, 19(3):387–414, 2002.

[21] C. Joder, S. Essid, and G. Richard. A conditional random field framework for robust and scalable audio-to-score matching. *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, 19(8):2385–2397, November 2011.

[22] L. Jure and M. Rocamora. Microtiming in the rhythmic structure of Candombe drumming patterns. In *4th Int. Conf. on Analytical Approaches to World Music (AAWM)*, New York, USA, June 2016.

[23] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] F. Korzeniowski, S. Böck, and G. Widmer. Probabilistic extraction of beat positions from a beat activation function. In *15th Int. Society for Music Information Retrieval Conf. (ISMIR)*, pages 513–518, Taipei, Taiwan, October 2014.

[25] F. Krebs, S. Böck, M. Dorfer, and G. Widmer. Downbeat tracking using beat synchronous features with recurrent neural networks. In *17th Int. Society for Music Information Retrieval Conf. (ISMIR)*, pages 129–135, New York, USA, August 2016.

[26] J. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289, June 2001.

[27] J. Laroche. Estimating tempo, swing and beat locations in audio recordings. In *IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 135–138, New Paltz, USA, October 2001.

[28] K. A. Lindsay and P. R. Nordquist. More than a feeling: Some technical details of swing rhythm in music. *Acoustics Today*, 3(3):31–42, July 2007.

[29] L. S. Maia, P. D. T. Jr., M. Fuentes, M. Rocamora, L. W. P. Biscainho, M. V. M. Costa, and S. Cohen. A novel dataset of Brazilian rhythmic instruments and some experiments in computational rhythm analysis. In *2018 AES Latin American Congress of Audio Engineering (AES LAC)*, pages 53–60, Montevideo, Uruguay, September 2018.

[30] U. Marchand and G. Peeters. Swing ratio estimation. In *18th Int. Conf. on Digital Audio Effects (DAFx)*, pages 423–428, Trondheim, Norway, December 2015.

[31] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenbergk, and O. Nieto. librosa: Audio and music signal analysis in Python. In *14th Python in Science Conf. (SciPy)*, pages 18–24, Austin, USA, July 2015.

[32] L. Naveda, F. Gouyon, C. Guedes, and M. Leman. Microtiming patterns and interactions with musical properties in samba music. *Journal of New Music Research*, 40(3):225–238, 2011.

[33] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis. mir_eval: A transparent implementation of common MIR metrics. In *15th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 367–372, Taipei, Taiwan, October 2014.

[34] M. Rocamora. *Computational methods for percussion music analysis : The Afro-Uruguayan Candombe drumming as a case study*. PhD thesis, Universidad de la República (Uruguay). Facultad de Ingeniería. IIE, April 2018.

[35] M. Rocamora, L. Jure, B. Marenco, M. Fuentes, F. Lanzaro, and A. Gómez. An audio-visual database of Candombe performances for computational musicological studies. In *II Congreso Int. de Ciencia y Tecnología Musical (CICTeM)*, pages 17–24, Buenos Aires, Argentina, September 2015.

[36] A. Srinivasamurthy, A. Holzapfel, A. T. Cemgil, and X. Serra. A generalized bayesian model for tracking long metrical cycles in acoustic music signals. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 76–80, Shanghai, China, March 2016.

[37] C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*, chapter 4, pages 93–128. MIT Press, Cambridge, USA, 2006.

[38] C. H. Waadeland. "It dont mean a thing if it aint got that swing"—Simulating expressive timing by modulated movements. *Journal of New Music Research*, 30(1):23–37, 2001.

[39] M. Wright and E. Berdahl. Towards machine learning of expressive microtiming in Brazilian drumming. In *ICMC*. Citeseer, 2006.

HARMONY TRANSFORMER: INCORPORATING CHORD SEGMENTATION INTO HARMONY RECOGNITION

Tsung-Ping Chen and Li Su
Institute of Information Science, Academia Sinica, Taiwan
{tearfulcanon, lisu}@iis.sinica.edu.tw

ABSTRACT

Musical harmony analysis is usually a process of unfolding and interpreting the hierarchical structure of music. Computational approaches to such structural analysis are still challenging, owing to the fact that the boundary between different harmonic states (such as chord functions) is not explicitly defined in the audio or symbolic music data. It is a novel approach to improve chord recognition by jointly identifying chord change using end-to-end sequence learning. In this paper, we propose the Harmony Transformer, a multi-task music harmony analysis model aiming to improve chord recognition through incorporating chord segmentation into the recognition process. The integration of chord segmentation and chord recognition is implemented with the Transformer, a deep sequential learning model yielding fruitful results in the field of natural language processing. A non-autoregressive decoding framework is also adopted here in aid of concatenating the two highly correlated tasks. Experiments of both chord symbol recognition and functional harmony recognition on audio and symbolic datasets demonstrate that explicitly learning the hierarchical structural information of musical data can facilitate and improve the harmony recognition.

1. INTRODUCTION

Automatic chord recognition is a hallmark research topic in the field of music information retrieval (MIR) and has been widely studied. In the past decade, this problem has been dealt with by using a variety of deep learning methods ranging from multi-layer perceptrons (MLPs) [1] to more complex models such as convolutional neural networks (CNNs) [2–6] and recurrent neural networks (RNNs) [7–11]. Hybrid systems such as convolutional recurrent neural network (CRNN) are also introduced to the chord recognition task for taking advantage of different neural networks [12–15].

In spite of the significant achievements for deep learning models to advance the state of the art, further improvement in the chord recognition task appears to be limited and has reached a glass ceiling [4]. Such limitations may result from the incompetence for deep learning methods to infer the hierarchical structure of music based on frame-level data. An overlooked tendency in many previous studies is that the majority of chord recognition models regard a music piece as a concatenation of semantically incomplete segments, and therefore are ineffective to capture complex musical grammar such as chord transition [13]. Figure 2 (see Section 4.4) depicts an example where an RNN model cannot segment chord boundaries adequately when arpeggios or key modulation exists. This problem was dealt with by integrating temporal knowledge such as the metric and beat information [16, 17], by adding post-processing such as hidden Markov models (HMM) [18, 19], or by feature engineering such as the harmonic change detection function (HCDF) [20] and beat-synchronous audio features [21]. One recent study attempted to solve this problem by incorporating models which recognize chords at different levels, i.e., the frame level and the chord level, and the cooperative models gained improved chord recognition results [14].

In this paper, we propose the Harmony Transformer, which jointly integrates chord recognition and chord segmentation into an end-to-end chord recognition system. The Harmony Transformer is based on the Transformer, a deep neural network nowadays applied to a wide variety of sequence modeling problems such as machine translation [22], natural language understanding [23], music generation [24], and so forth. For the encoder-decoder architecture of the Transformer, we assign the chord segmentation task to the encoder, and the chord recognition task to the decoder. This division allows the chord recognition network to benefit from the prediction of the chord segmentation. In comparison to other designs using RNN, which is considerably time-consuming owing to heavy sequential computations, the Transformer is more effective in speed, as it enforces parallelization to process sequential data. The Harmony Transformer is also capable of a complex analytic scenario—harmonic function [1] recognition [25]. Our experiment results highlight the advantage of the proposed model to improve both the chord symbol recognition and the harmonic function recognition. To the best of our knowledge, this is the first work that connects the chord segmentation task with the chord recognition task through the Transformer framework. We hope

[1] 'Harmonic function' refers to the diatonic function in music which denotes the relationship of a chord to a tonal center.

that our contribution facilitates more developments in automatic chord recognition.

2. HIERARCHICAL STRUCTURES AND CHORD RECOGNITION

The chord recognition can be formulated as a *sequence labeling* task [26] that assigns a categorical label (e.g., chord name) to each element of a given sequence (e.g., musical sequence). Considering the sequential and structural nature of music, chord recognition task resembles many natural language processing (NLP) problems, in such a way that both the sequential arrangements of chords in music and words in language are dominated by higher-level rules—the function of harmony and the grammar in language. A compelling recognition model for music or language is, therefore, expected to unfold the intrinsic hierarchical structure of the sequences. A typical example is the part-of-speech (POS) tagging task in NLP, which assigns a specific part of speech to each word in a sentence. Albeit many words can represent more than one part of speech at different times, deep learning models have shown their ability to regulate local-level assignments with a given context, and demonstrated satisfactory performance. [2]

On the other hand, the chord recognition results in recent years still could not be satisfactory [1,8,13–15]. A primary difference between language and music is that each word in the input sentence of the POS tagging task represents a semantically meaningful unit, whereas the musical segments to be labeled in the chord recognition task are not necessary harmonically completed. It can be argued that the unsatisfying achievements in the chord recognition task can be associated with the deficiency of explicit knowledge of the harmonic boundaries within a musical sequence. Since the sequential property and the hierarchical structure are shared characteristics between language and the harmonic progression, it would be advantageous to introduce the segmentation information of harmony in higher-level musical hierarchy into deep learning models of chord recognition.

Recent research tackled the hierarchical issue of frame-wise chord recognition using two RNN-based models to separately predict the duration and the transition of each chord in a *parallel* fashion [14]. Concretely, frame-wise chord predictions are first generated through a CNN-based *acoustic model*, which are then simultaneously fed into the two RNN-based models—the *chord duration model* and the *harmonic language model* to predict the time points of chord change at the frame level and the chord progression at the chord level respectively. The two RNN-based models are trained separately from the acoustic model, and are utilized as *language model* which computes the probability distribution of each token sequentially.

It has been demonstrated that incorporating musical hierarchy is promising to improve chord recognition. Nevertheless, the local assignment of chord labels in practice is dependent *successively* on the knowledge from different hierarchical levels. We hence argue that 1) the chord change and the chord progression should be modeled in a vertical rather than a horizontal manner; in other words, we suppose that the chord recognition problem belongs to a higher level in the musical hierarchy than the chord segmentation problem, and the recognition of chords should base on the result of segmentation. 2) The models processing higher-level information such as the two RNN-based language models should be trained *jointly* with the lower-level ones. In this study, we propose a new framework that predicts chord progression for a given sequence according to the chord segmentation result, and is trained in an end-to-end manner.

[2] The per-token accuracy of the POS tagging task had surpassed 96% in 2000. For an overview of the state of the art of the POS tagging task, see: `https://aclweb.org/aclwiki/POS_Tagging_(State_of_the_art)`.

3. THE HARMONY TRANSFORMER

The proposed model, as shown in Figure 1, follows the encoder-decoder architecture of the Transformer [22], while adopting a non-autoregressive framework for treating segmentation as the intermediate before the recognition process [27]. The encoder of the Harmonic Transformer performs the chord segmentation task from the input data sequence (e.g., chroma), and the decoder performs the chord recognition task from the input data sequence along with the segmentation result from the encoder.

3.1 Basic units

The Harmony Transformer model is built on two computational units, that is, *multihead-head attention* (MHA) and *feed-forward network* (FFN). The MHA unit takes three inputs $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, which stand for *queries*, *keys*, and *values*, respectively. The three inputs are matrices with the first dimension representing the number of time steps and the second dimension being the number of feature; the feature sizes of the three inputs are usually the same, while their number of time steps may be different.

In general, the MHA unit computes a weighted sum of $\mathbf{V}$ based on the similarity between $\mathbf{Q}$ and $\mathbf{K}$. To be more exact, the inputs are first 'transformed' through an *attention* mechanism which outputs a *head* (denoted as $\mathbf{D}$). The MHA unit is then constructed by concatenating multiple heads out of several attention mechanisms:

$$\mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d}}\right)\mathbf{V}\,, \quad (1)$$

$$\mathbf{D}_i = \mathrm{Attention}(\mathbf{Q}\mathbf{W}_i^{\mathrm{Q}}, \mathbf{K}\mathbf{W}_i^{\mathrm{K}}, \mathbf{V}\mathbf{W}_i^{\mathrm{V}})\,, \quad (2)$$

$$\mathrm{MHA}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{Concat}(\mathbf{D}_1,\cdots,\mathbf{D}_h)\mathbf{W}^{\mathrm{D}}\,, \quad (3)$$

where h is the number of heads, $\mathbf{D}_i$ is the ith head, and d is the dimension of feature. $\mathbf{W}^{\mathrm{D}} \in \mathbb{R}^{d\times d}$, and $\mathbf{W}^{\mathrm{Q}}$, $\mathbf{W}^{\mathrm{K}}$, $\mathbf{W}^{\mathrm{V}} \in \mathbb{R}^{d\times \frac{d}{h}}$ are all learnable parameter matrices. Note that if $\mathbf{Q} = \mathbf{K} = \mathbf{V}$, such attention mechanism is called *self-attention* as the three inputs stand for the same sequence; if $\mathbf{Q} \neq \mathbf{K} = \mathbf{V}$, such attention mechanism is called *encoder-decoder attention* in the case when $\mathbf{K}$ and $\mathbf{V}$ come from the encoder and $\mathbf{Q}$ comes from the decoder.

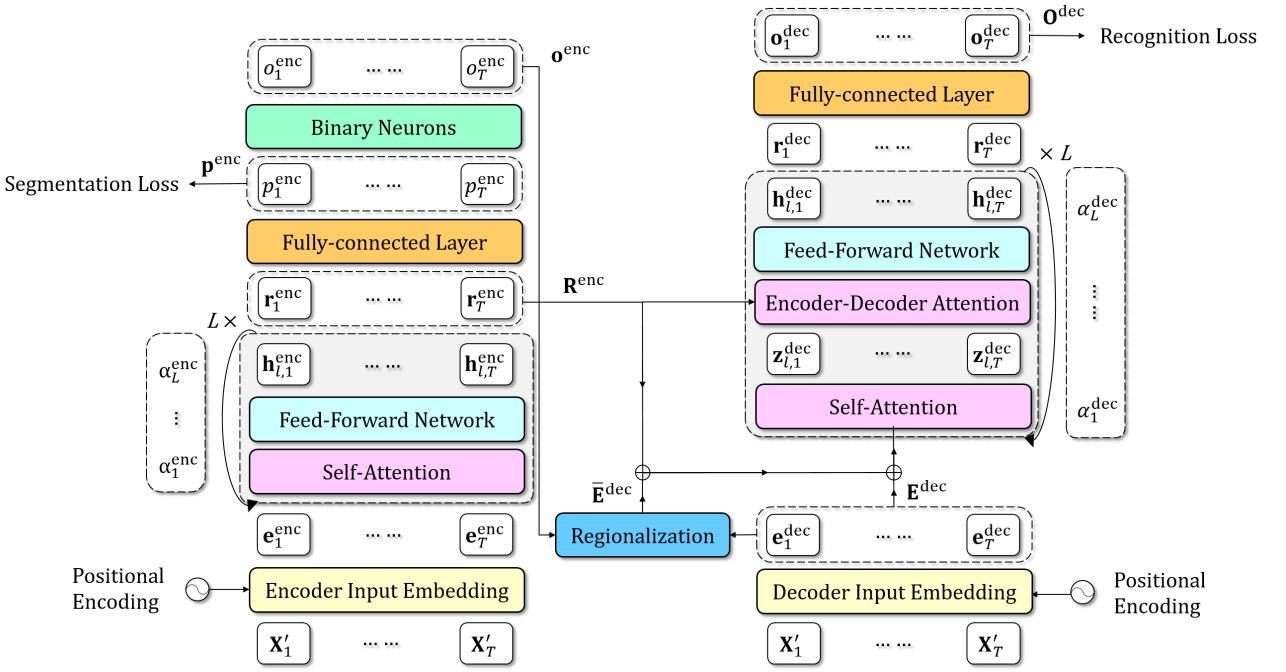


Figure 1. The architecture of the Harmony Transformer. During training, the chord change prediction $\mathbf{p}^{\text{enc}}$ from the encoder is used to calculate the segmentation loss, while the decoder output $\mathbf{O}^{\text{dec}}$ is used to calculate the recognition loss.

On the other hand, the FFN unit is constructed by two fully-connected layers parameterized by weighted matrices and bias vectors $(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2)$ and one activation function in between the two layers:

$$\text{FFN}(\mathbf{X}) = \text{ReLU}(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2\,, \qquad (4)^{3}$$

where $\mathbf{X}$ is also a matrix like $\mathbf{Q}$, $\mathbf{K}$, or $\mathbf{V}$, and $\text{ReLU}(\mathbf{X}) := \max(0, \mathbf{X})$ is the element-wise rectified linear unit activation. In practice, both the MHA and the FFN computations include residual connections [28] and layer normalization [29]:

$$\text{MHA}^*(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{LayerNorm}(\text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) + \mathbf{Q})\,,$$
$$\text{FFN}^*(\mathbf{X}) = \text{LayerNorm}(\text{FFN}(\mathbf{X}) + \mathbf{X})\,.$$

In the rest of the paper, we omit the $*$ symbol for simplicity. More detailed descriptions of MHA and FFN can be found in Section 3.2 and Section 3.3 of [22].

3.2 Input embedding

We denote the sequence of frame-level features (e.g., piano roll or chromagram) with T time steps as $\mathbf{X} := [\mathbf{x}_1, \ldots, \mathbf{x}_T]^{\text{T}}$,[4] in which $\mathbf{x}_t$ is a feature vector of $\mathbf{X}$ at time t. The data representation entering the Harmony Transformer at time t, as denoted by $\mathbf{X}'_t$ here, is a segment of $\mathbf{X}$ around t: $\mathbf{X}'_t := [\mathbf{x}_{t-\tau}, \cdots, \mathbf{x}_{t+\tau}]^{\text{T}}$, where the length of the segment is $2\tau+1$. For the embedding process, the encoder (or decoder) maps each $\mathbf{X}'_t$ into an encoder (or decoder) embedding through MHA and FFN units, then flattens it into a vector $\mathbf{e}_t^{\text{enc}}$ (or $\mathbf{e}_t^{\text{dec}}$). More specifically, the embedding vector $\mathbf{e}_t$ which is to be fed into time step t of either the encoder or the decoder is:

$$\mathbf{e}_t = \text{Flatten}(\text{FFN}(\text{MHA}(\mathbf{X}'_t, \mathbf{X}'_t, \mathbf{X}'_t))\mathbf{W}^{\text{e}}), \qquad (5)$$

where $\mathbf{W}^{\text{e}}$ is the parameter matrix to be learned, and the encoder and the decoder use different set of parameters. The embedding sequences for the encoder and the decoder are then $\mathbf{E}^{\text{enc}} = [\mathbf{e}_1^{\text{enc}}, \cdots, \mathbf{e}_T^{\text{enc}}]^{\text{T}}$ and $\mathbf{E}^{\text{dec}} = [\mathbf{e}_1^{\text{dec}}, \cdots, \mathbf{e}_T^{\text{dec}}]^{\text{T}}$, respectively.

Moreover, since the computational units mentioned in Section 3.1 are intrinsically unaware of the sequential order of their inputs, the positional information is added to the feature embeddings. In this work, we adopt the absolute positional encoding composed of sinusoidal functions, as used in [22]:

$$\text{PE}_{t,i} = \begin{cases} \sin(t/10000^{\frac{i}{d}}) \text{ if } i \text{ is even,} \\ \cos(t/10000^{\frac{i}{d}}) \text{ if } i \text{ is odd,} \end{cases} \qquad (6)$$

where $t = 1, \ldots, T$ is the time step, d is the embedding size, and $i = 1, \ldots, d$ is the index of the embedding dimension. And $\text{PE}_{t,i}$ is added into the ith element of $\mathbf{e}_t$.

3.3 Encoder: chord segmentation

The encoder is composed of L layers, each of which comprises a self-attention MHA unit and a FFN unit. Inspired by [30], L softmax-normalized parameters are introduced to the encoder for weighting the hidden states of each layer. Formally, for the embedded sequence of the encoder, $\mathbf{E}^{\text{enc}}$, the encoder computes hidden states

[3] Note that the two additions in the equation are element-wise additions which will 'broadcast' the bias vectors to the same dimensions as the matrices to be added.

[4] In our notation, the normal T means the transpose of a matrix, while the italic T indicates the number of time steps.

$\mathbf{H}_l^{\text{enc}} := [\mathbf{h}_{l,1}^{\text{enc}}, \cdots, \mathbf{h}_{l,T}^{\text{enc}}]^{\text{T}}$ at layer l, where $\mathbf{h}_{l,t}$ denotes the output at time t in the lth layer:

$$\begin{aligned} \mathbf{H}_l^{\text{enc}} &= \text{FFN}(\text{MHA}(\mathbf{H}_{l-1}^{\text{enc}}, \mathbf{H}_{l-1}^{\text{enc}}, \mathbf{H}_{l-1}^{\text{enc}}))\,, \\ \mathbf{H}_0^{\text{enc}} &= \mathbf{E}^{\text{enc}}\,. \end{aligned} \tag{7}$$

The hidden states of all layers are then weighted by the softmax-normalized parameters $\boldsymbol{\alpha}^{\text{enc}} := \{\alpha_l^{\text{enc}}\}_{l=1}^{L}$ to produce a weighted-sum representation $\mathbf{R}^{\text{enc}} := [\mathbf{r}_1^{\text{enc}}, \cdots, \mathbf{r}_T^{\text{enc}}]^{\text{T}}$ with $\mathbf{r}_t^{\text{enc}} = \sum_{l=1}^{L} \alpha_l^{\text{enc}} \mathbf{h}_{l,t}^{\text{enc}}$, which is later taken as one of the inputs of the decoder. The purpose of using the weighted sum of hidden states from all layers instead of using the output of the last layer is that different layers tend to encode specific information which individual tasks may rely on to different extents [30–33].

To predict chord change, $\mathbf{R}^{\text{enc}}$ is then fed into a fully-connected layer followed by a sigmoid activation. That is, the likelihood of chord change at time t is estimated by the equation: $p_t^{\text{enc}} = \text{sigmoid}(\mathbf{w}^{\text{T}}\mathbf{r}_t^{\text{enc}})$, where $\mathbf{w}$ are the parameters to be learned for mapping each $\mathbf{r}_t^{\text{enc}}$ in $\mathbf{R}^{\text{enc}}$ into a real number. And the *segmentation loss* is calculated with $\mathbf{p}^{\text{enc}} := \{p_t^{\text{enc}}\}_{t=1}^{T}$ during training, as depicted in Figure 1. In order to make use of the chord change prediction for the later part of the chord recognition task, we utilize deterministic binary neurons (DBNs) [34, 35] to binarize the real-valued chord change probabilities with hard thresholding. Accordingly, the final output of the encoder $\mathbf{o}^{\text{enc}} := \{o_t^{\text{enc}}\}_{t=1}^{T}$ is a sequence of binary chord segmentation prediction, which is 1 at the point of chord change, or 0 otherwise: $o_t^{\text{enc}} = 1$ if $p_t^{\text{enc}} > 0.5$ and $o_t^{\text{enc}} = 0$ if $p_t^{\text{enc}} \leq 0.5$. That means, when $o_t^{\text{enc}} = 1$, there is a chord change at time t. For example, for a source sequence of 6 segments, $\mathbf{o}^{\text{enc}} = [1, 0, 0, 1, 1, 0,]$ means that there are three *chord regions* with the first region containing three segments, the second containing one segment, and the final containing two segments.

3.4 Decoder: segmentation-informed chord recognition

Similar to the encoder, the decoder also consists of L layers, while in each layer, there is an additional encoder-decoder attention module besides the MHA and the FNN modules to imitates the classical attention mechanism in sequence-to-sequence models [36, 37]. Different from the encoder, the input of the decoder is derived from three sources: $\mathbf{E}^{\text{dec}}$, $\mathbf{o}^{\text{enc}}$, and $\mathbf{R}^{\text{enc}}$. First of all, the embedded sequence of the decoder $\mathbf{E}^{\text{dec}}$ is *regionalized* in line with $\mathbf{o}^{\text{enc}}$ to generate $\bar{\mathbf{E}}^{\text{dec}}$. Precisely, let $\mathbf{c} := \{c_k\}_{k=1}^{K}$ be the K time steps where chord changes, i.e., $o_{c_k}^{\text{enc}} = 1$, then the regionalization unit in Figure 1 replaces each member $\mathbf{e}_t^{\text{dec}}$ in $\mathbf{E}^{\text{dec}}$ with average pooling:

$$\bar{\mathbf{e}}_t^{\text{dec}} := \frac{1}{c_{k+1} - c_k} \sum_{i=c_k}^{c_{k+1}-1} \mathbf{e}_i^{\text{dec}} \quad \text{for } t \in [c_k, c_{k+1})\,, \tag{8}$$

and the resulting embedding is $\bar{\mathbf{E}}^{\text{dec}} := [\bar{\mathbf{e}}_1^{\text{dec}}, \cdots, \bar{\mathbf{e}}_T^{\text{dec}}]^{\text{T}}$. Next, the decoder takes the original embedding $\mathbf{E}^{\text{dec}}$, the regionalized embedding $\bar{\mathbf{E}}^{\text{dec}}$, and the weighted-sum representation $\mathbf{R}^{\text{enc}}$ to compute hidden states of the decoder $\mathbf{H}_l^{\text{dec}} = [\mathbf{h}_{l,1}^{\text{dec}}, \cdots, \mathbf{h}_{l,T}^{\text{dec}}]^{\text{T}}$ at layer l with the three modules in the layer:

$$\mathbf{H}_l^{\text{dec}} = \text{FFN}(\text{MHA}(\mathbf{Z}_l^{\text{dec}}, \mathbf{R}^{\text{enc}}, \mathbf{R}^{\text{enc}}))\,, \tag{9}$$

$$\mathbf{Z}_l^{\text{dec}} = \text{MHA}(\mathbf{H}_{l-1}^{\text{dec}}, \mathbf{H}_{l-1}^{\text{dec}}, \mathbf{H}_{l-1}^{\text{dec}})\,, \tag{10}$$

$$\mathbf{H}_0^{\text{dec}} = \mathbf{E}^{\text{dec}} + \bar{\mathbf{E}}^{\text{dec}} + \mathbf{R}^{\text{enc}}\,. \tag{11}$$

The intuition of adding the regionalized embeddings $\bar{\mathbf{E}}^{\text{dec}}$ to the decoder inputs is to guide the model to recognize the sequence with the segmentation-level, or chord-level information; and using $\mathbf{R}^{\text{enc}}$ is to take the advantage of the explicit alignment information of the sequence labeling problem for the encoder-decoder architecture [38].

Then, the decoder weighs the hidden states of all layers, as does in the encoder, with the softmax-normalized parameters $\boldsymbol{\alpha}^{\text{dec}} := \{\alpha_l^{\text{dec}}\}_{l=1}^{L}$ to produce the final presentation $\mathbf{R}^{\text{dec}}$ of the decoder inputs:

$$\mathbf{R}^{\text{dec}} = [\mathbf{r}_1^{\text{dec}}, \cdots, \mathbf{r}_T^{\text{dec}}]^{\text{T}} = \sum_{l=1}^{L} \alpha_l^{\text{dec}} \mathbf{H}_l^{\text{dec}}\,. \tag{12}$$

Finally, the representation $\mathbf{R}^{\text{dec}}$ is fed into a fully-connected layer followed by a softmax activation to predict the probability distribution over the chord vocabulary for each time step t in the source sequence:

$$\mathbf{O}^{\text{dec}} = [\mathbf{o}_1^{\text{dec}}, \cdots, \mathbf{o}_T^{\text{dec}}]^{\text{T}} = \text{softmax}(\mathbf{R}^{\text{dec}}\mathbf{W}^{\text{O}}), \tag{13}$$

where $\mathbf{W}^{\text{O}}$ is the parameter matrix of the fully-connected network which maps each $\mathbf{r}_t^{\text{dec}}$ in $\mathbf{R}^{\text{dec}}$ into a vector of the chord vocabulary size. And the *recognition loss* is calculated with $\mathbf{O}^{\text{dec}}$.

We denote the ground truth labels of chord change and chord symbol as $\hat{\mathbf{o}}^{\text{enc}}$ and $\hat{\mathbf{O}}^{\text{dec}}$ respectively. The total loss function L_{total} in the Harmony Transformer is:

$$L_{\text{total}} := \lambda_1 \text{BCE}(\mathbf{p}^{\text{enc}}, \hat{\mathbf{o}}^{\text{enc}}) + \lambda_2 \text{CCE}(\mathbf{O}^{\text{dec}}, \hat{\mathbf{O}}^{\text{dec}})\,, \tag{14}$$

where BCE is the binary crossentropy, CCE is the categorical crossentropy, and λ_1 and λ_2 are coefficients used for balancing the two cross entropies. The two terms in (14) correspond to the segmentation loss and the recognition loss in Figure 1, respectively. [5]

4. EXPERIMENTS

4.1 Data

The proposed model is evaluated on both audio and symbolic datasets. For the audio part, we use the **McGill Billboard** dataset, which consists of 890 musical pieces sampled from the Billboard chart slots [39]. [6] For the symbolic data, we use the **BPS-FH** dataset, in which the first movements of Beethoven's 32 piano sonatas are included [25]. The chord annotations are available in the two datasets; chord segmentation labels are further derived from the chord annotations for the supervised training of the chord segmentation task.

[5] The implementation can be found at `https://github.com/Tsung-Ping/Harmony-Transformer`.

[6] The dataset can be found at `http://ddmal.music.mcgill.ca/research/billboard`.

Dataset	BLSTM	FK	HT	HT*	F1 (Seg.)
Billboard	77.03	78.90	82.68	**83.00**	57.15
BPS-FH	78.87	-	83.96	**84.18**	66.65

Table 1. The chord symbol recognition results in term of WCSR score. BLSTM stands for the 1-layer bidirectional RNN using LSTM cells; both HT and HT* denote the proposed model, except the tonal centroid vector is added into the input feature of the latter. The F1 scores of the chord segmentation of HT are shown in the rightmost column.

Function	BLSTM	HT
Key	72.62	**78.35**
Degree	47.75	**65.06**
Secondary	48.23	**68.15**
Quality	53.31	**74.60**
Inversion	61.59	**62.13**
F1 (Seg.)	-	67.34

Table 2. The accuracy (in %) of harmonic function recognition. Note that *Degree* stands for the accuracy of correctly predicting both the primary and secondary degrees, and *Secondary* indicates the accuracy of correctly predicting the degrees of secondary chords. The segmentation result (with F1 measure) is shown in the bottom row.

4.1.1 Audio data

For each piece in the McGill Billboard dataset, the non-negative-least-squares (NNLS) chromagram is computed with the Chordino VAMP plugin, [40] in which the default settings of the plugin are adopted. Combining the 12-D treble chroma and the 12-D bass chroma, for each track we obtain a 24-by-T-dimensional chromagram, where T represents the length of the track. [7] Each input sequence for the Harmony Transformer contains 100 segments (around 23 sec), and is generated through a sliding window of frame size 21 with hop size 5. Following [13] (see Section 2.2), pieces with id numbers smaller than 1000 are used for training, and the remaining for testing; also, identical pieces are filtered out. As a result, there are 5,647 sequences for training and 1,628 sequences for testing.

4.1.2 Symbolic data

We represent the piano sonatas in the BPS-FH dataset as pianorolls with the pitch ranging from A0 to G$^{\#}$7 (middle C = C4), where the duration of each note is measured in term of crotchet beats, and is quantized to 32th note. The length of each sequence is 64, and each element in the sequence is a pianoroll segmented with window size of 33 and hop size of 2. As the input element for the Harmony Transformer, each pianoroll segment is flattened into a vector whose length is 84×33. We use 4-fold cross-validation for evaluation; the number of sequences of each fold varies from 368 to 585.

4.1.3 Data augmentation

All the training data are augmented with key modulation and thus expanded to 12 times.

4.2 Experimental trials

For the Harmony Transformer (denoted as HT), we use the embedding size $d = 512$, the number of heads $h = 8$, and the number of layers of both the encoder and the decoder $L = 2$. The two coefficients λ_1 and λ_2 in the loss function are set to be 3 and 1 respectively. The chord symbol recognition task is conducted for both audio and symbolic data, and the harmonic function recognition task, as defined in [25], is further applied to the symbolic dataset. Besides, we employ the *tonal centroid* vector [20], which models the relationship between chords in a 6-D tonal space, as the additional input feature of the HT for better representing the input data.

4.2.1 Chord symbol recognition

The chord symbol recognition model has a 26-dimensional output, in which 24 of them represent major and minor triads, 1 represents 'others' for chords other than major or minor triads, and the remaining one represents the 'no-chord' case. The weighted chord symbol recall (WCSR) is used as the evaluation metric. We employ a 1-layer bidirectional RNN using LSTM cells of 512 hidden units (abbreviated as BLSTM) as the baseline for the evaluations of the two datasets. Additionally, we include the best evaluation result achieved by the ConvNet-HMM model (denoted as FK here) in [13] (see Section 3) for the comparison of the evaluation on audio data. The FK model is similar to the Madmom audio chord recognition framework, [8] which achieves many state-of-the-art scores in the MIREX Audio Chord Estimation (ACE) campaign. [9]

4.2.2 Harmonic function recognition

We formulate the harmonic function recognition task as a multi-task learning problem, in which the model outputs segment-wise predictions of the 5 chord functions: *local key* of 24 classes, *primary degree* of 21 classes, *secondary degree* of 21 classes, *chord quality* of 10 classes, and *chord inversion* of 4 classes. We use the classification accuracy to measure the performance of the proposed model. The BLSTM, as mentioned in Section 4.2.1, is also employed for comparison. For more information about the terminology of harmonic function, please refer to [25].

[7] For more information of the Chordino VAMP plugin and the NNLS chromagram, please refer to `http://www.isophonics.net/nnls-chroma`.

[8] `https://madmom.readthedocs.io/en/latest/modules/features/chords.html`.

[9] See `https://www.music-ir.org/mirex/wiki/2018:Audio_Chord_Estimation_Results`. The algorithm denoted as FK2 in the website is part of the Madmom audio processing framework.

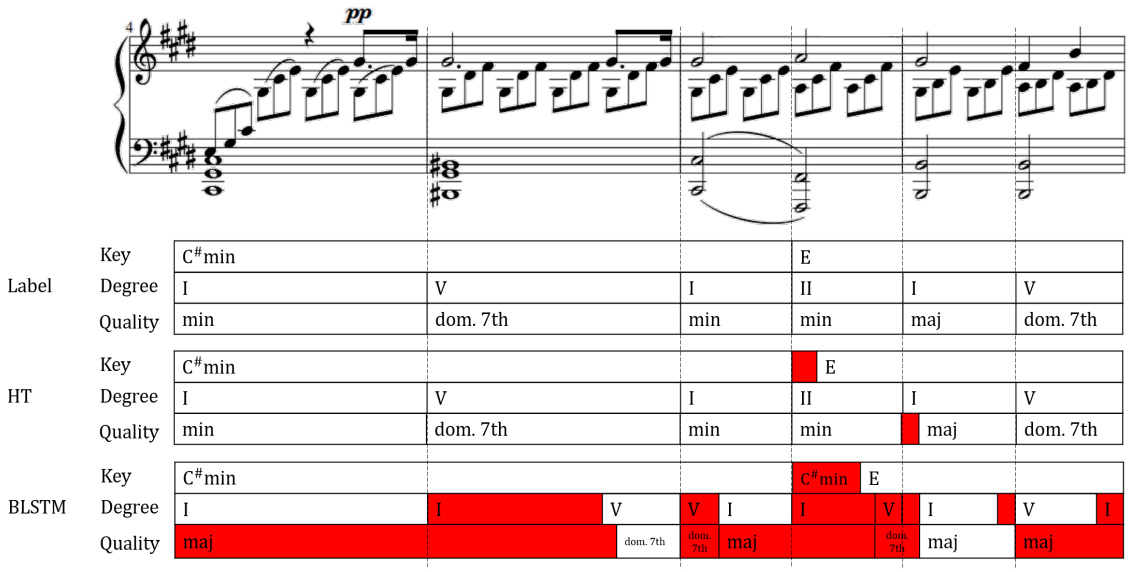


Figure 2. The predictions of harmonic function by the proposed model HT and the BLSTM, where *maj*, *min*, and *dom. 7th* stand for major, minor, and dominant seventh, respectively. Wrong predictions are marked in red. The example here is an excerpt from Beethoven's piano sonata No. 14, Op. 27-2, Mvt. 1, MM. 4-7.

4.3 Hyperparameters and training

The model is trained with the Adam optimization method of the learning rate $= 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$. To avoid overfitting, we employ dropout [41] of rate $= 0.6$ and label smoothing [42] of the value $\epsilon_{ls} = 0.1$. In addition, because the encoder output contains discrete variables, i.e., the binary chord segmentation predictions $\mathbf{o}^{\text{enc}}$, we utilize a straight-through estimator along with the slope annealing trick [43] to estimate the derivative gradient during backpropagation.

4.4 Evaluation results

Table 1 shows the evaluation results of the chord symbol recognition task. For the Billboard dataset, the performance of the Harmony Transformer outperforms the BLSTM and the FK by 5.65% and 3.78% respectively. For the BPS-FH dataset, our model also surpasses the BLSTM by 5.09%. When we employ the tonal centroid vector for the input feature, the proposed model further boosts the recognition results. This indicates that the explicit information of harmonic change is useful for chord recognition. The fact that the proposed model outperforms the BLSTM in both audio and symbolic data exhibits our model's capability of sequence learning even though the model consists of no temporally recurrent computation. In addition, based on the same training and evaluation data, the proposed model performs better than the FK and therefore may compete with the ACE framework of Madmom. Nevertheless, the F1 scores for the segmentation task are not satisfied due to the low recall rates (57.10% and 58.69% for Billboard and BPS-FH). This indicates the challenge to identify the exact time when chord changes.

For the harmonic function recognition task, the HT outperforms the BLSTM in all of the five chord functions, as shown in Table 2. In particular, for local key, chord degree, secondary chord degree, and chord quality, the HT outperforms the BLSTM greatly by 5.73%, 17.31%, 19.92%, and 21.29%, respectively. This indicates that our model is able to deal with challenging cases such as the key modulation, and chords with special harmonic function. Figure 2 gives an example of such instances. The HT correctly predicts the chord progression in terms of local key, chord degree, and chord quality, except that the timing of modulating to E major is slightly later than the ground truth. Notably, the F minor triad at the second half of measure 6, functioned as a pivot chord in a common chord modulation, is precisely identified by our model in spite of the key change. In contrast, the BLSTM not only fails to recognize the degree of some chords in this example, but also mistakes the minor triads as major ones, such as in measure 4.

5. CONCLUSION AND FUTURE WORK

We have demonstrated that the Harmony Transformer is competent in harmony analysis. Built upon the commonality of musical chord recognition and natural language processing, we emphasize the importance of modeling segmental and hierarchical structures in music. The encoder-decoder architecture and the non-autoregressive decoding in the Harmony Transformer enhance the flexibility in modeling chord sequences. Specifically, the end-to-end combination of chord segmentation and chord recognition contributes great benefits to the chord symbol recognition, as well as to the joint recognition of five harmony functions, a challenging task that relies heavily on the contextual and structural information in music. Our model has the potential to be further improved by using the segmentation results to learn the *word-level* embedding, which has also witnessed success in natural language processing.

6. REFERENCES

[1] Filip Korzeniowski and Gerhard Widmer. Feature learning for chord recognition: the deep chroma extractor. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 37–43, 2016.

[2] Eric J. Humphrey and Juan P. Bello. Rethinking automatic chord recognition with convolutional neural networks. In *Proceedings of the International Conference on Machine Learning and Applications (ICMLA)*, page 357–362, 2012.

[3] Xinquan Zhou and Alexander Lerch. Chord detection using deep learning. In *Proceedings of the 16th International Conference on Music Information Retrieval (ISMIR)*, pages 52–58, 2015.

[4] Eric J. Humphrey and Juan P. Bello. Four timely insights on automatic chord estimation. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 673–679, 2015.

[5] Filip Korzeniowski and Gerhard Widmer. A fully convolutional deep auditory model for musical chord recognition. In *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2016.

[6] Tristan Carsault, Jérôme Nika, and Philippe Esling. Using musical relationships between chord labels in automatic chord extraction tasks. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 18–25, 2018.

[7] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio chord recognition with recurrent neural networks. In *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, pages 335–340, 2013.

[8] Junqi Deng and Yu-Kwong Kwok. Large vocabulary automatic chord estimation with an even chance training scheme. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 531–536, 2017.

[9] Takeshi Hori, Kazuyuki Nakamura, and Shigeki Sagayama. Music chord recognition from audio data using bidirectional encoder-decoder LSTMs. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1312–1315, 2017.

[10] Filip Korzeniowski, David R. W. Sears, and Gerhard Widmer. A large-scale study of language models for chord prediction. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 91–95, 2018.

[11] David R. W. Sears, Filip Korzeniowski, and Gerhard Widmer. Evaluating language models of tonal harmony. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 211–217, 2018.

[12] Brian McFee and Juan P. Bello. Structured training for large-vocabulary chord recognition. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 188–194, 2017.

[13] Filip Korzeniowski and Gerhard Widmer. On the futility of learning complex frame-level language models for chord recognition. In *Proceedings of the Audio Engineering Society (AES) International Conference on Semantic Audio*, 2017.

[14] Filip Korzeniowski and Gerhard Widmer. Improved chord recognition by combining duration and harmonic language models. In *Proceedings of the 19th International Conference on Music Information Retrieval (ISMIR)*, pages 10–17, 2018.

[15] Yiming Wu and Wei Li. Music chord recognition based on midi-trained deep feature and BLSTM-CRF hybird decoding. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 376–380, 2018.

[16] W. Bas de Haas, José Pedro Magalhães, and Frans Wiering. Improving audio chord transcription by exploiting harmonic and metric knowledge. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 295–300, 2012.

[17] Veronika Zenz and Andreas Rauber. Automatic chord detection incorporating beat and key detection. In *Proceedings of the IEEE International Conference on Signal Processing and Communications (ICSPC)*, pages 1175–1178, 2007.

[18] Kyogu Lee and Malcolm Slaney. Automatic chord recognition from audio using an HMM with supervised learning. In *Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR)*, pages 133–137, 2006.

[19] Ruofeng Chen, Weibin Shen, Ajay Srinivasamurthy, and Parag Chordia. Chord recognition using duration-explicit hidden markov models. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 445–450, 2012.

[20] Christopher Harte, Mark Sandler, and Martin Gasser. Detecting harmonic change in musical audio. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 21–26, 2006.

[21] Matthias Mauch, Katy C. Noland, and Simon Dixon. Using musical structure to enhance automatic chord transcription. In *Proceedings of the 10th International*

Society for Music Information Retrieval Conference (ISMIR), pages 231–236, 2009.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL-HLT)*, pages 4171–4186, 2019.

[24] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *arXiv preprint arXiv: 1810.12247*, 2018.

[25] Tsung-Ping Chen and Li Su. Functional harmony recognition of symbolic music data with multi-task recurrent neural networks. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 90–97, 2018.

[26] Hakan Erdogan. Sequence labeling: generative and discriminative approaches, hidden markov models, conditional random fields and structured SVMs. In *the Tutorial of International Conference on Machine Learning and Applications (ICMLA)*, 2010.

[27] Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations (ICLR)*, 2018.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[29] Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. In *arXiv preprint arXiv: 1607.06450*, 2016.

[30] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2227–2237, 2018.

[31] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, 2016.

[32] Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James R. Glass. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 861–872, 2017.

[33] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1923–1933, 2017.

[34] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. In *arXiv preprint arXiv: 1308.3432*, 2013.

[35] Hao-Wen Dong and Yi-Hsuan Yang. Convolutional generative adversarial networks with binary neurons for polyphonic music generation. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 190–196, 2018.

[36] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, pages 1923–1933, 2015.

[37] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1243–1252, 2017.

[38] Bing Liu and Ian Lane. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Proceedings of the 17th Annual Conference of the International Speech Communication Association (ISCA)*, pages 685–689, 2016.

[39] John Ashley Burgoyne, Jonathan Wild, and Ichiro Fujinaga. An expert ground truth set for audio chord recognition and music analysis. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, pages 633–638, 2011.

[40] Matthias Mauch and Simon Dixon. Approximate note transcription for the improved identification of difficult chords. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, pages 135–140, 2010.

[41] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. In *Journal of Machine Learning Research*, pages 1929–1958, 2014.

[42] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[43] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. In *5th International Conference on Learning Representations (ICLR)*, 2017.

STATISTICAL MUSIC STRUCTURE ANALYSIS BASED ON A HOMOGENEITY-, REPETITIVENESS-, AND REGULARITY-AWARE HIERARCHICAL HIDDEN SEMI-MARKOV MODEL

Go Shibata **Ryo Nishikimi** **Eita Nakamura** **Kazuyoshi Yoshii**
Graduate School of Informatics, Kyoto University, Japan
{gshibata, nishikimi, enakamura, yoshii}@sap.ist.i.kyoto-u.ac.jp

ABSTRACT

This paper describes a music structure analysis method that splits music audio signals into meaningful segments such as musical sections and clusters them. In this task, how to model the four fundamental aspects of musical sections, *i.e.*, homogeneity, repetitiveness, novelty, and regularity, in a unified way is still an open problem. Here we propose a solid statistical approach based on a homogeneity-, repetitiveness-, and regularity-aware hierarchical hidden semi-Markov model. The higher-level semi-Markov chain represents a sequence of sections that tend to have *regularly* spaced boundaries. The timbral features in each section are assumed to follow emission distributions that are *homogeneous* over time. The lower-level left-to-right Markov chain in each section represents a chord sequence whose sequential order is constrained to be a *repetition* of a chord sequence in another section of the same cluster. The whole model can be trained unsupervisedly based on Bayesian sparse learning where unnecessary sections automatically degenerate. The proposed method outperformed representative methods in segmentation and clustering accuracies with estimated sections having similar statistical properties as the ground truth data.

1. INTRODUCTION

Music structure analysis is a long-standing research topic [1] because detection of meaningful segments called musical sections (*e.g.*, intro, verse, bridge, and chorus sections in popular music) from music audio signals forms a basis of music information retrieval (MIR). In general, music structure analysis involves a *segmentation* step that splits music signals into sections [2–9], a *clustering* step that categorizes such sections into several classes [10–18], and a *labeling* step that gives each section a concrete label such as "verse A", "verse B", or "chorus" [19–21]. We here tackle the segmentation and clustering for popular music.

In previous studies, sections of popular music have been characterized in three aspects, *i.e.*, *homogeneity* referring to the intra-section characteristics and *repetitiveness* and *novelty* referring to the inter-section relationships [1]. More specifically, homogeneity means that musical characteristics (*e.g.*, timbral features such as mel-frequency cepstrum coefficients (MFCCs)) are consistent within a section. Repetitiveness means that the same sequence of some musical elements (*e.g.*, chroma features and chord progressions) of a section is repeated in sections of the same class. Novelty means that musical characteristics change abruptly at a boundary between sections. In addition, *regularity* of section durations has often been focused on [8–10] because there are typical lengths such as four or eight measures in popular music.

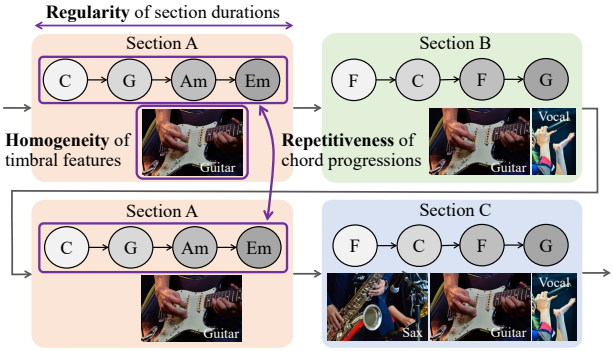


Figure 1. Music structure analysis based on homogeneity of timbral features, repetitiveness of chord progressions, and regularity of section durations.

Most studies on music structure analysis, however, have focused on only one of the above aspects or consider some of them in a separate and/or ad-hoc manner, as reviewed in Section 2. Joint computational modeling of these four aspects is thus the central issue in music structure analysis. Instead of manually designing segmentation and clustering criteria based on these aspects, we pursue a statistical approach to *data-driven* music structure analysis.

In this study, we propose a statistical music structure analysis method that simultaneously deals with the homogeneity, repetitiveness, and regularity of musical sections in a probabilistic framework (Fig. 1). We formulate a unified probabilistic model called a hierarchical hidden semi-Markov model (HHSMM) that represents the hierarchical generative process of musical sections, chord sequences, and music audio signals (timbral features and chroma features). This model has two sequences of latent states in a hierarchical manner. The upper-level sequence

represents a series of section classes following a semi-Markov model with explicit regularity (duration) modeling and the lower-level sequence represents chords following a section-conditioned left-to-right Markov model. To represent the intra-section homogeneity of timbral features, the MFCCs of a section are assumed to be generated from an upper-level state corresponding to the section. To represent the inter-section repetitiveness of chord progressions, the chroma features of a section are assumed to be sequentially generated from lower-level states. Given a music audio signal as observed data, the whole model can be trained unsupervisedly using Gibbs sampling and Viterbi training, where unnecessary sections are automatically degenerated during the Bayesian sparse learning.

The main contribution of this study is to propose a solid Bayesian approach to music signal analysis based on a fully *generative* model that can deal with the homogeneity, repetitiveness, and regularity of sections in a unified way. This enables *unsupervised* learning unlike another statistical approach based on deep *discriminative* models [6–8] that require section annotations for supervised training. Since these two approaches have a mutually complementary relationship, our results open up a door to deep Bayesian integration of discriminative and generative models in a variational autoencoding framework (audio signal $\rightarrow$ sections $\rightarrow$ audio signal) [22] for further improvement.

Another important contribution is to investigate the statistical characteristics of musical sections estimated by the proposed method in comparison with representative methods. Our method is shown to be able to yield distributions of section durations, of the numbers of section classes used for representing music audio signals, and of the metrical positions of section boundaries much more similar to those of the ground-truth data than the other methods.

2. RELATED WORK

Homogeneity, repetitiveness, novelty, and regularity have been considered to be important for music structure analysis. The most standard approach to music structure analysis is to use the self-similarity matrix (SSM) of acoustic features such as chroma features and MFCCs, whose element represents the acoustic similarity between two time frames (Fig. 2). In an SSM, the homogeneity, repetitiveness, novelty, and regularity are observed as block-diagonal structure, short stripes parallel to the diagonal line, grid patterns, and grid intervals, respectively. One or some of these four aspects have been used for segmentation and clustering tasks in music structure analysis.

2.1 Segmentation

Foote [2] proposed a novelty-based method that detects peaks from a time-varying novelty curve obtained by shifting and convoluting a checkerboard kernel along the diagonal elements of an SSM. Jensen [3] attempted to find section boundaries that minimize a homogeneity- and novelty-based cost function. While Goto [19] originally proposed a lag SSM in which repetitions appear not as stripes but as vertical lines and calculated a novelty curve over

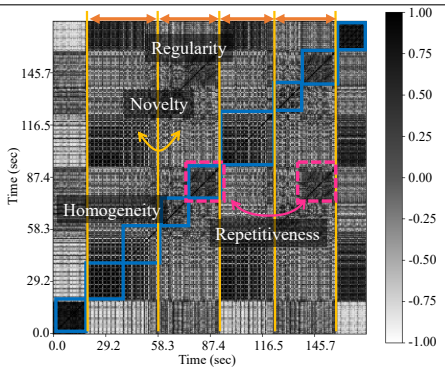


Figure 2. Self-similarity matrix (SSM) of MFCC features (part of RWC-MDB-P-2001 No. 25).

time lags, Serrà [4] proposed another novelty curve over time frames. Peeters and Bisot [5] successfully integrated these two methods [4, 19] for better segmentation. Ullrich *et al.* [6] pioneered a supervised approach based on a convolutional neural network (CNN), which was extended to deal with both coarse- and fine-level boundary annotations [7]. Sargent *et al.* [9] pointed out the effectiveness of focusing on the regularity to favor structural segments of comparable size. Maezawa [8] developed a long short-term memory (LSTM) network with homogeneity-, repetitiveness-, novelty-, and regularity-based cost functions. In this study we take an *unsupervised* approach based on a homogeneity-, repetitiveness-, and regularity-based *generative* model. To keep the model simple, incorporation the aspect of novelty is left for future work.

2.2 Clustering

Cooper *et al.* [12] sequentially performed music segmentation [2] and section clustering based on intra- and inter-section statistical characteristics. Goodwin *et al.* [13] attempted to efficiently detect off-diagonal stripes in an SSM as repetitions using dynamic programming. To deal with repetitiveness and homogeneity, Grohganz *et al.* [14] converted a repetitiveness-dominant SSM with off-diagonal stripes into a homogeneity-dominant SSM with block-diagonal structure. Nieto *et al.* [15] used a convex variant of non-negative matrix factorization for section segmentation and clustering. McFee *et al.* [16] proposed a method that encodes repetitive structures into a graph and performs spectral clustering for graph partitioning.

Several studies took a statistical approach based on generative models for joint segmentation and clustering. Aucouturier *et al.* [11] used a standard hidden Markov model (HMM). Ren *et al.* [17] proposed a nonparametric Bayesian extension of an HMM that can estimate an appropriate number of sections. Barrington *et al.* [18] proposed a nonparametric Bayesian extension of a switching linear dynamical system (LDS) that also has the ability of automatic model complexity control. While these methods mainly focused on the homogeneity, our method simultaneously considers the homogeneity, repetitiveness, and regularity and can incorporate prior knowledge about the statistical characteristics of sections (*e.g.*, durations and metrical positions) in a data-driven manner.

3. PROPOSED METHOD

This section describes the proposed statistical method for music structure analysis.

3.1 Problem Specification

Our problem is formulated as follows:

Input: A chroma feature sequence $\mathbf{X}^c = \mathbf{x}^c_{1:B} \in \mathbb{R}^{B\times 12}$ and an MFCC sequence $\mathbf{X}^m = \mathbf{x}^m_{1:B} \in \mathbb{R}^{B\times 12}$ obtained from a given music audio signal. They are computed in units of beats estimated by a beat tracking method [23].
Output: Boundaries and classes of sections.

Here, B is the number of beats (quarter notes) and a subscript $\bigcirc_{a:b}$ represents the sequence $(\bigcirc_a, \ldots, \bigcirc_b)$.

We use 12-dimensional chroma features obtained by aggregating all spectral information of each pitch class into a single element. We also use 12-dimensional MFCCs as timbral features.

3.2 Model Formulation

The proposed model consists of two-level hierarchical Markov chains and an acoustic model as shown in Fig. 3. The upper-level Markov chain describes the section-level structure (*i.e.*, section classes and durations) and the lower-level Markov chain describes the internal structure of each section. The states of these Markov chains are latent variables that represent abstract musical structure. The acoustic model connects the abstract structure with the observed musical features (chroma vectors and MFCCs).

3.2.1 Upper-Level Markov Chain

The upper-level Markov chain is an ergodic semi-Markov model. The sequence of section classes $\mathbf{Z} = z_{1:T}$ ($z_\tau \in \{1, \ldots, N_Z\}$) and their durations $\mathbf{D} = d_{1:T}$ ($d_\tau \in \{1, \ldots, N_D\}$) are generated by the model, where T is the number of sections, N_Z is the number of distinct section classes, and N_D is the maximum duration of a section. The generative process is described as follows:

$$p(z_1, d_1) = \rho_{z_1}\psi_{d_1}, \tag{1}$$

$$p(z_\tau, d_\tau | z_{\tau-1}, d_{\tau-1}) = \pi_{z_{\tau-1}z_\tau}\psi_{d_\tau}, \tag{2}$$

where ρ_z and $\pi_{zz'}$ are the initial and transition probabilities of section classes and ψ_d is the duration probability.

3.2.2 Lower-Level Markov Chain

The lower-level Markov chain is a left-to-right Markov model with N_K states. The state sequence of this model describes the internal structure of a section corresponding to the chord progression, where each state is expected to correspond to a chord. We consider such a Markov chain for each section class. The model continues state transitions for each beat from the start time of a section until its duration passes. The state sequence $\mathbf{K}_\tau = k_{\tau,1:d_\tau}$ ($k_{\tau,t} \in \{1, \ldots, N_K\}$) is generated as follows:

$$p(k_{\tau,t} | z_\tau, k_{\tau,t-1}) = \phi^{(z_\tau)}_{k_{\tau,t-1}k_{\tau,t}}, \tag{3}$$

where z_τ and d_τ are the corresponding section class and duration, and $\phi^{(z_\tau)}_{kk'}$ is the transition probability from state k to state k'.

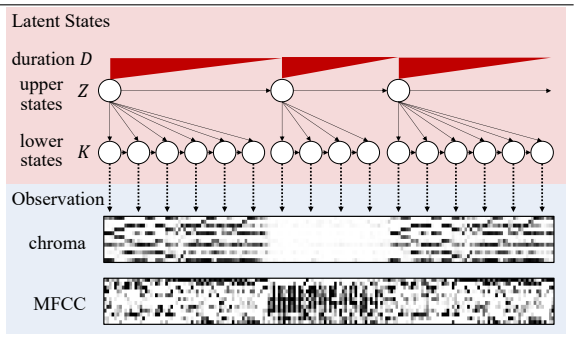


Figure 3. The proposed generative model.

The left-to-right Markov model meets a condition that the initial state has $k_{\tau,1} = 1$ and $k_{\tau,t_1} \le k_{\tau,t_2}$ for $t_1 < t_2$. We introduce a hyperparameter σ that describes the maximum number of states that may be skipped in a transition; a transition from state k to state $k+\sigma$ is possible but a larger skip is forbidden. In this way, the model can describe repetitions with some variations, which is another important aspect of musical structure. $\mathbf{K}$ denotes $\mathbf{K}_{1:T}$.

3.2.3 Acoustic Model

The acoustic model describes the generative processes of the chroma features $\mathbf{x}^c_b \in \mathbb{R}^{12}$ and MFCCs $\mathbf{x}^m_b \in \mathbb{R}^{12}$ by using output probabilities that are defined conditionally on the section classes $\mathbf{Z}$ and their internal states $\mathbf{K}$. The repetitiveness is represented by applying the same set of output probabilities to all sections of the same class. The output probabilities of chroma features $\chi^c_{z,k}$ depend on both $\mathbf{Z}$ and $\mathbf{K}$ to represent the sequential structure of chord progressions. To capture the homogeneity of timbre characteristics of each section, the output probabilities of MFCCs χ^m_z are assumed to depend only on $\mathbf{Z}$. Thus we have

$$p(\mathbf{x}^c_b, \mathbf{x}^m_b) = \chi^c_{z_b,k_b}(\mathbf{x}^c_b)\chi^m_{z_b}(\mathbf{x}^m_b), \tag{4}$$

where z_b and k_b are the section class and the internal state at beat b, respectively. The output probabilities are described as multivariate Gaussian distributions:

$$\chi^c_{z,k}(\mathbf{x}^c_b) = \mathcal{N}(\mathbf{x}^c_b | \boldsymbol{\mu}^c_{z,k}, (\boldsymbol{\Lambda}^c_{z,k})^{-1}), \tag{5}$$

$$\chi^m_z(\mathbf{x}^m_b) = \mathcal{N}(\mathbf{x}^m_b | \boldsymbol{\mu}^m_z, (\boldsymbol{\Lambda}^m_z)^{-1}), \tag{6}$$

where $\boldsymbol{\mu}^c_{z,k}$ and $\boldsymbol{\Lambda}^c_{z,k}$ are the mean and precision for the chroma features, and $\boldsymbol{\mu}^m_z$ and $\boldsymbol{\Lambda}^m_z$ are defined similarly.

3.2.4 Prior Distributions

To find an effective number of distinct section classes, we formulate a Bayesian HHSMM by putting conjugate prior distributions. We put Dirichlet prior distributions for the categorical distributions as follows:

$$\boldsymbol{\rho} \sim \text{Dirichlet}(\mathbf{a}^{\boldsymbol{\rho}}), \tag{7}$$

$$\boldsymbol{\psi} \sim \text{Dirichlet}(\mathbf{a}^{\boldsymbol{\psi}}), \tag{8}$$

$$\boldsymbol{\pi}_z \sim \text{Dirichlet}(\mathbf{a}^{\boldsymbol{\pi}}), \tag{9}$$

$$\boldsymbol{\phi}^{(z)}_k \sim \text{Dirichlet}(\mathbf{a}^{\boldsymbol{\phi}}), \tag{10}$$

where $\boldsymbol{\rho} = \rho_{1:N_Z}$, $\boldsymbol{\psi} = \psi_{1:N_D}$, $\boldsymbol{\pi}_z = \pi_{z(1:N_Z)}$, $\boldsymbol{\phi}^{(z)}_k = \phi^{(z)}_{k(1:N_K)}$, and $\mathbf{a}^{\boldsymbol{\rho}}$, $\mathbf{a}^{\boldsymbol{\psi}}$, $\mathbf{a}^{\boldsymbol{\pi}}$, and $\mathbf{a}^{\boldsymbol{\phi}}$ are Dirichlet parameters. When these parameters are small, the transition probabilities of section classes become sparse. The model can

thus capture repetitions regardless of small acoustic variations and remove unnecessary section classes.

Since section durations tend to be the integer multiples of four measures in popular music (see Fig. 4), such a statistical tendency can be incorporated in the prior distribution. Specifically, we use as $\mathbf{a}^{\boldsymbol{\psi}}$ the empirical distribution of section durations $\mathbf{a}^{\boldsymbol{\psi}}_{\mathrm{emp}}$ multiplied by a constant factor. Since the structure of section classes is quite different over individual musical pieces, we put uniform Dirichlet prior distributions for their transition probabilities.

Finally, we put Gaussian-Wishart prior distributions on multivariate normal distributions as follows:

$$\boldsymbol{\mu}^c_{z,k}, \boldsymbol{\Lambda}^c_{z,k} \sim \mathcal{N}(\boldsymbol{\mu}^c_{z,k}|\mathbf{m}^c_0, (\beta^c_0\boldsymbol{\Lambda}^c_{z,k})^{-1})\, \mathcal{W}(\boldsymbol{\Lambda}^c_{z,k}|\mathbf{W}^c_0, \nu^c_0),$$
$$\boldsymbol{\mu}^m_z, \boldsymbol{\Lambda}^m_z \sim \mathcal{N}(\boldsymbol{\mu}^m_z|\mathbf{m}^m_0, (\beta^m_0\boldsymbol{\Lambda}^m_z)^{-1})\, \mathcal{W}(\boldsymbol{\Lambda}^m_z|\mathbf{W}^m_0, \nu^m_0),$$

where $\mathbf{m}^c_0$, β^c_0, $\mathbf{W}^c_0$, ν^c_0, $\mathbf{m}^m_0$, β^m_0, $\mathbf{W}^m_0$, and ν^m_0 are hyperparameters.

3.3 Bayesian Learning

Letting $\boldsymbol{\Theta} = \{\boldsymbol{\rho}, \boldsymbol{\psi}, \boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}\}$, we aim to calculate the posterior distribution $p(\mathbf{Z}, \mathbf{D}, \mathbf{K}, \boldsymbol{\Theta}|\mathbf{X}^c, \mathbf{X}^m)$. Since this is analytically intractable, we use the Gibbs sampling method. We first sample the latent variables $\mathbf{Z}$, $\mathbf{D}$, and $\mathbf{K}$ from the distribution $p(\mathbf{Z}, \mathbf{D}, \mathbf{K}|\boldsymbol{\Theta}, \mathbf{X}^c, \mathbf{X}^m)$ and we then sample the model parameters $\boldsymbol{\Theta}$ from the distribution $p(\boldsymbol{\Theta}|\mathbf{Z}, \mathbf{D}, \mathbf{K}, \mathbf{X}^c, \mathbf{X}^m)$. Iterating this process, we obtain samples from the true posterior distribution.

3.3.1 Sampling Latent Variables

We use the forward filtering-backward sampling algorithm for sampling the upper- and lower-level latent variables $\mathbf{Z}$, $\mathbf{D}$, and $\mathbf{K}$. We introduce variables z_b and d_b that denote the class and duration of a section starting at beat $b-d_b+1$ and ending at beat b. We also define the marginalized output probabilities for this section $\omega_{z_b}(\mathbf{x}^c_{b-d_b+1:b}, \mathbf{x}^m_{b-d_b+1:b})$, which can be calculated by the forward algorithm for the lower-level Markov chain.

In the forward filtering step for the upper-level model, we initialize and update the forward variables $\alpha_b(z_b, d_b) = p(z_b, d_b, \mathbf{x}^c_{1:b}, \mathbf{x}^m_{1:b})$ as follows:

$$\alpha_b(z_b, d_b = b) = \rho_{z_b}\psi_{d_b}\omega_{z_b}(\mathbf{x}^c_{1:b}, \mathbf{x}^m_{1:b}), \tag{11}$$

$$\alpha_b(z_b, d_b) \tag{12}$$
$$= \sum_{z', d'} \alpha_{b-d_b}(z', d')\pi_{z'z_b}\psi_{d_b}\omega_{z_b}(\mathbf{x}^c_{b-d_b+1:b}, \mathbf{x}^m_{b-d_b+1:b}).$$

In the backward sampling step, the latent variables $\mathbf{Z}$ and $\mathbf{D}$ are sequentially sampled in the reverse order:

$$p(z_B, d_B|\mathbf{X}^c, \mathbf{X}^m) \propto \alpha_B(z_B, d_B). \tag{13}$$

When variables z_b and d_b are already sampled, the variables $z_{b'}$ and $d_{b'}$ at beat $b' = b - d_b$ are sampled according to the probability

$$p(z_{b'}, d_{b'}|z_{b:B}, d_{b:B}, \mathbf{X}^c, \mathbf{X}^m) \propto \alpha_{b'}(z_{b'}, d_{b'})\pi_{z_{b'}z_b}. \tag{14}$$

Next, the latent variables $\mathbf{K}$ are sampled using the sampled $\mathbf{Z}$ and $\mathbf{D}$. Each set of variables $\mathbf{K}_\tau$ is sampled by forward filtering-backward sampling for the lower-level model of section class z_τ. Here we use a beat index $t \in \{1, \ldots, d_\tau\}$ considered in relative to the section boundary. In the forward filtering step, we calculate the probabilities $\zeta_{\tau,k_{\tau,t}}$ recursively as follows:

$$\zeta_{\tau,k_{\tau,1}} = p(k_{\tau,1}, \mathbf{x}^c_1, \mathbf{x}^m_1|z_\tau, d_\tau)$$
$$= \delta_{k_{\tau,1}1}\chi^c_{z_\tau,1}(\mathbf{x}^c_1)\chi^m_{z_\tau}(\mathbf{x}^m_1), \tag{15}$$

$$\zeta_{\tau,k_{\tau,t}} = p(k_{\tau,t}, \mathbf{x}^c_{1:t}, \mathbf{x}^m_{1:t}|z_\tau, d_\tau) \tag{16}$$
$$= \left(\sum_{k_{\tau,t-1}} \zeta_{\tau,k_{\tau,t-1}}\phi^{(z_\tau)}_{k_{\tau,t-1}k_{\tau,t}}\right)\chi^c_{z_\tau,k_{\tau,t}}(\mathbf{x}^c_t)\chi^m_{z_\tau}(\mathbf{x}^m_t).$$

In the backward sampling step, the latent variables $\mathbf{K}$ are sequentially sampled in the reverse order as follows:

$$p(k_{\tau,d_\tau}|z_\tau, d_\tau, \mathbf{x}^c_{1:d_\tau}, \mathbf{x}^m_{1:d_\tau}) \propto \zeta_{\tau,k_{\tau,d_\tau}}, \tag{17}$$

$$p(k_{\tau,t}|z_\tau, d_\tau, k_{\tau,t+1:d_\tau}, \mathbf{x}^c_{1:d_\tau}, \mathbf{x}^m_{1:d_\tau}) \propto \zeta_{\tau,k_{\tau,t}}\phi^{(z_\tau)}_{k_{\tau,t}k_{\tau,t+1}}. \tag{18}$$

3.3.2 Sampling Model Parameters

We use the Gibbs sampling method for updating the model parameters as follows:

$$\boldsymbol{\rho} \sim \mathrm{Dirichlet}(\mathbf{a}^{\boldsymbol{\rho}} + \mathbf{b}^{\boldsymbol{\rho}}), \tag{19}$$
$$\boldsymbol{\pi}_z \sim \mathrm{Dirichlet}(\mathbf{a}^{\boldsymbol{\pi}} + \mathbf{b}^{\boldsymbol{\pi}_z}), \tag{20}$$
$$\boldsymbol{\psi} \sim \mathrm{Dirichlet}(\mathbf{a}^{\boldsymbol{\psi}} + \mathbf{b}^{\boldsymbol{\psi}}), \tag{21}$$
$$\boldsymbol{\phi}^{(z)}_k \sim \mathrm{Dirichlet}(\mathbf{a}^{\boldsymbol{\phi}} + \mathbf{b}^{\boldsymbol{\phi}^{(z)}_k}), \tag{22}$$
$$\boldsymbol{\Lambda}^c_{z,k} \sim \mathcal{W}(\mathbf{W}^c_{z,k}, \nu^c_{z,k}), \tag{23}$$
$$\boldsymbol{\mu}^c_{z,k}|\boldsymbol{\Lambda}^c_{z,k} \sim \mathcal{N}(\mathbf{m}^c_{z,k}, (\beta^c_{z,k}\boldsymbol{\Lambda}^c_{z,k})^{-1}), \tag{24}$$
$$\boldsymbol{\Lambda}^m_z \sim \mathcal{W}(\mathbf{W}^m_z, \nu^m_z), \tag{25}$$
$$\boldsymbol{\mu}^m_z|\boldsymbol{\Lambda}^m_z \sim \mathcal{N}(\mathbf{m}^m_z, (\beta^m_z\boldsymbol{\Lambda}^m_z)^{-1}), \tag{26}$$

where $\mathbf{b}^{\boldsymbol{\rho}} \in \mathbb{R}^{N_Z}$, $\mathbf{b}^{\boldsymbol{\pi}_z} \in \mathbb{R}^{N_Z}$, $\mathbf{b}^{\boldsymbol{\psi}} \in \mathbb{R}^{N_D}$, and $\mathbf{b}^{\boldsymbol{\phi}^{(z)}_k} \in \mathbb{R}^{N_K}$ are vectors counting the sampled data. $b^{\boldsymbol{\rho}}_z$ is 1 if $z = z_1$ and 0 otherwise, $b^{\boldsymbol{\pi}_z}_{z'}$ counts the number of transitions from state z to state z', $b^{\boldsymbol{\psi}}_d$ counts the number of times that sampled sections have a duration of d, and $b^{\boldsymbol{\phi}^{(z)}_k}_{k'}$ counts the number of transitions from state k to state k' in the lower-level model of section class z. The parameters $\mathbf{m}^c_{z,k}$, $\beta^c_{z,k}$, $\mathbf{W}^c_{z,k}$, and $\nu^c_{z,k}$ are calculated as follows:

$$\beta^c_{z,k} = \beta^c_0 + N_{z,k}, \quad \nu^c_{z,k} = \nu^c_0 + N_{z,k}, \tag{27}$$

$$\mathbf{m}^c_{z,k} = \frac{1}{\beta^c_{z,k}}(\beta^c_0\mathbf{m}^c_0 + N_{z,k}\overline{\mathbf{x}}^c_{z,k}), \tag{28}$$

$$(\mathbf{W}^c_{z,k})^{-1} = (\mathbf{W}^c_0)^{-1} + N_{z,k}\mathbf{S}^c_{z,k} + \frac{\beta^c_0 N_{z,k}}{\beta^c_0 + N_{z,k}}(\overline{\mathbf{x}}^c_{z,k} - \mathbf{m}^c_0)(\overline{\mathbf{x}}^c_{z,k} - \mathbf{m}^c_0)^{\mathrm{T}}, \tag{29}$$

where we have defined

$$N_{z,k} = \sum_{b=1}^{B} \delta_{z_b z}\delta_{k_b k}, \tag{30}$$

$$\overline{\mathbf{x}}^c_{z,k} = \frac{1}{N_{z,k}}\sum_{b=1}^{B} \delta_{z_b z}\delta_{k_b k}\mathbf{x}^c_b, \tag{31}$$

$$\mathbf{S}^c_{z,k} = \frac{1}{N_{z,k}}\sum_{b=1}^{B} \delta_{z_b z}\delta_{k_b k}(\mathbf{x}^c_b - \overline{\mathbf{x}}^c_{z,k})(\mathbf{x}^c_b - \overline{\mathbf{x}}^c_{z,k})^{\mathrm{T}}. \tag{32}$$

The parameters $\mathbf{m}^m_z$, β^m_z, $\mathbf{W}^m_z$, and ν^m_z can be calculated similarly.

3.3.3 Refinements

We introduce two refinements to facilitate the learning process. First, since the samples from the Gibbs sampler are not necessarily local optimums of the posterior distribution, we apply Viterbi training in the last step of the parameter estimation. Specifically, we apply the Viterbi algorithm (instead of the forward filtering-backward sampling algorithm) to estimate the latent variables and update the model parameters to the expectation values of the posterior probabilities (instead of samples from those probabilities). It is known that Viterbi training is generally efficient for finding an approximate local minimum [24].

Second, we introduce a weighting factor $w_{\mathrm{dur}}(\geq 1)$ for the duration probability to enhance its effect. Specifically, we replace the probability factor ψ_{d_b} in the forward algorithm (11) and (12) with $(\psi_{d_b})^{w_{\mathrm{dur}}}$. Similar replacements are applied to the Viterbi training step as well as to the final estimation step of latent states explained in Section 3.4. Increasing the weighting factor has the effect of lowering the temperature and putting more focus on more frequent section durations.

3.4 Estimation of Musical Sections

After training the model parameters, we compute the maximum a posteriori (MAP) estimate of the latent variables (musical sections). Specifically, we maximize the posterior probability $p(\mathbf{Z}, \mathbf{D}|\mathbf{\Theta}, \mathbf{X}^c, \mathbf{X}^m)$ with respect to latent variables $\mathbf{Z}$ and $\mathbf{D}$. This can be solved by integrating out the lower-level states $\mathbf{K}$ and applying the Viterbi algorithm for hidden semi-Markov models [25] to the upper-level model.

4. EVALUATION

4.1 Experimental Conditions

We used the RWC popular music database [26] with structure annotations [27] for evaluation. Out of the 100 pieces in the data, we used 85 musical pieces in consistent 4/4 time for simplicity. For running the proposed method, we obtained chroma features using the deep feature extractor [28] and MFCCs using the librosa library [29]. Beat information was obtained using the madmom library [23]. The empirical distribution $\mathbf{a}^{\boldsymbol{\psi}}_{\mathrm{emp}}$ of section durations was trained using the piece-wise cross validation among the 85 pieces. For parameter estimation, we iterated the Gibbs sampling 15 times and the Viterbi training 3 times, which took roughly around 5 times the duration of an input signal with a standard CPU.

The hyperparameters of the proposed models were set as follows: $N_Z = 12$, $N_D = 40$, $N_K = 16$, $\sigma = 1$, $w_{\mathrm{dur}} = 4$, $\mathbf{a}^{\boldsymbol{\rho}} = 0.1 \cdot \mathbb{I}$, $\mathbf{a}^{\boldsymbol{\pi}} = \mathbb{I}$, $\mathbf{a}^{\boldsymbol{\psi}} = 50 \cdot \mathbf{a}^{\boldsymbol{\psi}}_{\mathrm{emp}}$, $\mathbf{a}^{\boldsymbol{\phi}} = \mathbb{I}$, $\mathbf{m}^c_0 = \mathbb{E}[\mathbf{X}^c]$, $\beta^c_0 = 8$, $\mathbf{W}^c_0 = (\nu^c_0 \operatorname{cov}[\mathbf{X}^c])^{-1}$ with $\nu^c_0 = 96$, $\mathbf{m}^m_0 = \mathbb{E}[\mathbf{X}^m]$, $\beta^m_0 = 4$, and $\mathbf{W}^m_0 = (\nu^m_0 \operatorname{cov}[\mathbf{X}^m])^{-1}$ with $\nu^m_0 = 80$, where $\mathbb{I}$ denotes a vector with all entries equal to 1. The first three parameters N_Z, N_D, and N_K were determined by consulting the statistics of the annotated data (see Fig. 4). According to the data, most songs have 12 or less sections and most sections have a length of 40 beats or less. If we expect a section length of 32 beats (8 measures) and a chord duration of 2 beats, the expected number of chords in each section is 16. The value of σ is set to 1 for simplicity. The other parameters were determined by a coarse optimization w.r.t. the two evaluation measures explained below. Each parameter was optimized by a grid search, fixing the other parameters. Further optimization of the parameters is left for future work.

Method	$F_{0.5}$ (%) (segmentation)	F_{pair} (%) (clustering)
VMO [30]	8.72	28.5
CNMF [15]	17.4	41.7
SCluster [16]	23.4	45.5
Proposed	**33.0**	**54.3**

Table 1. Evaluation results.

For comparison, we ran variable Markov oracle (VMO) [30], convex non-negative matrix factorization (CNMF) [15], and spectral clustering (SCluster) [16], which were available in the music structure analysis framework (MSAF) [31]. We used the default settings in the MSAF.

We evaluated the quality of segmentation and clustering in the same way as MIREX [32]. The quality of segmentation is evaluated by the F-measure $F_{0.5}$ of section boundaries [33] defined as follows. An estimated boundary is accepted as correct if there is a boundary in the ground truth data within the range of ± 0.5 seconds. The precision rate is the percentage of correct estimates. The recall rate is the percentage of true boundaries that are correctly estimated. The F-measure $F_{0.5}$ is defined as the harmonic mean of the precision and recall.

The quality of clustering is evaluated by the pairwise F-measure F_{pair} [34] defined as follows. We compare pairs of frames (with a length of 100 ms) that are labeled with the same class in an estimation result with those in the ground truth. The precision, recall, and F-measure are defined as

$$P_{\mathrm{pair}} = \frac{|P_E \cap P_A|}{|P_E|}, \quad R_{\mathrm{pair}} = \frac{|P_E \cap P_A|}{|P_A|}, \tag{33}$$

$$F_{\mathrm{pair}} = \frac{2P_{\mathrm{pair}}R_{\mathrm{pair}}}{P_{\mathrm{pair}} + R_{\mathrm{pair}}}, \tag{34}$$

where P_E denotes the set of similarly labeled frame pairs in the estimation and P_A denotes that in the ground truth. These values are calculated by the `mir_eval` library [35].

4.2 Experimental Results

The results in Table 1 show that SCluster had the best $F_{0.5}$ and F_{pair} among the three conventional methods. The F-measures obtained by VMO were very low and the estimated results included many unnatural short segments (see Fig. 4). This was presumably caused by the implementation in the MASF. In both $F_{0.5}$ and F_{pair}, the proposed method significantly outperformed the three compared methods.

Next, let us examine the estimated results more closely (Fig. 4). The distribution of section durations for the proposed model was similar to that of the ground truth. In particular, both distributions have peaks at the 32 beats (eight measures) and 16 beats (four measures). In contrast, the distributions for the results of the other three methods were

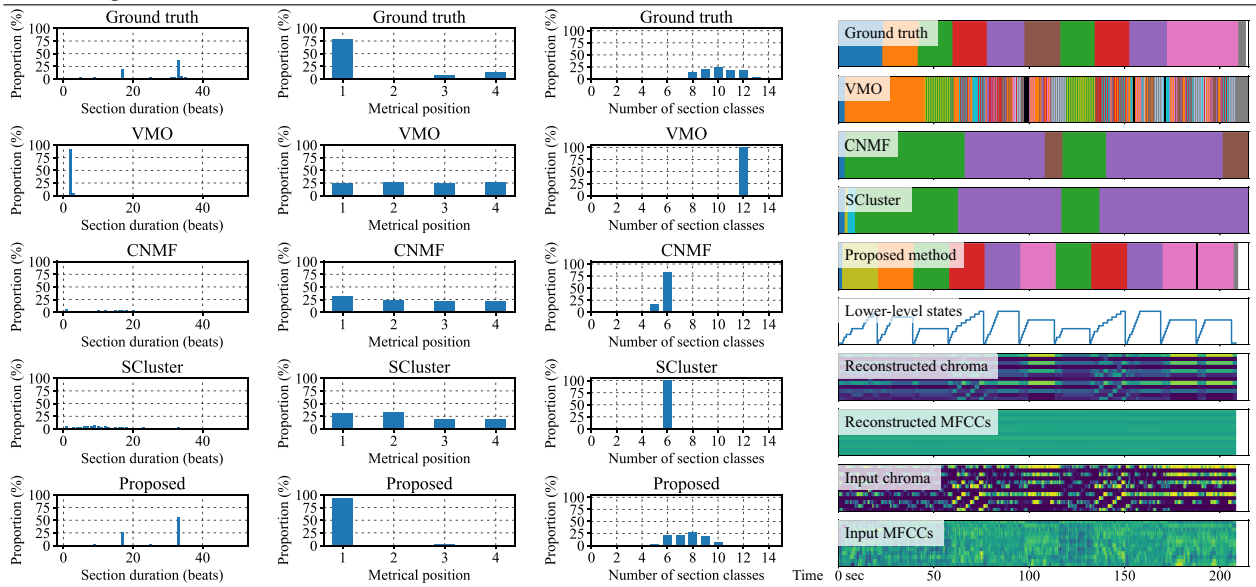


Figure 4. The left panels show the distributions of section durations, those of metrical positions of section boundaries, and those of the numbers of section classes in the estimated results and ground truth data. The right figure shows example results by the proposed and the three existing methods (RWC-MDB-P-2001 No. 29). The lower-level states are obtained by the Viterbi algorithm and the reconstructed features indicate the mean values of the corresponding output probabilities.

significantly different from that of the ground truth. This result clearly demonstrates the effect of explicitly modelling the section durations to capture their regularity. We also found that the distribution of metrical positions of section boundaries for the ground truth data was similar to that for the proposed method, but significantly different from those for the conventional methods.

The numbers of section classes in the ground truth data were roughly distributed in the range from eight to twelve. The distribution for the proposed model had a similar shape, though it is slightly shifted to the lower side. This result demonstrates the nontrivial ability of the proposed method to automatically find the appropriate number of section classes, even though it often finds the number smaller than the actual value. On the other hand, the distributions for the other methods were much more sparse; they found more or less the same number of section classes for all the tested pieces. In particular, CNMF and SCluster estimated too few section classes.

From these analyses, we find that the results of music structure analysis by the proposed method have much more similarity with the human annotated results than the compared existing methods. It is also important to point out that these results could not be made clear only by looking at the F-measures. The F-measures are not sufficient for evaluating results of music structure analysis.

We can observe these tendencies in the example results (Fig. 4). Particularly, CNMF and SCluster estimated too few section classes and irregular section durations. For the proposed method, we see that sections of the same class had similar lower-level sequences of latent states. This suggests that the model successfully captured repeated chord progressions in the sections of the same class. We can also observe that the proposed model often used only a part of lower-level states, which might be improved by imposing more constraints on the lower-level Markov chain.

For a fair comparison, we remark that parameters of the three existing methods were not optimized using our training data. Since we used limited data containing only J-pop pieces, adapting the parameters of these methods to this particular musical style may improve their performance to some extent. In addition, using the state-of-the-art beat tracker [23] to obtain reliable beat information and using that as input to those three methods may also improve their accuracy. However, it is unlikely that these methods can be refined to reproduce the aforementioned statistics of sections simply by re-training the parameters.

5. CONCLUSION

We have presented a statistical method for music structure analysis based on a Bayesian HHSMM that describes intra- and inter-section structures in a unified way. Three of the most important aspects of musical sections, homogeneity, repetitiveness, and regularity are incorporated into the model. Music segmentation and section clustering are performed jointly by unsupervised Bayesian learning of the model, and musically important characteristics such as the repetitive structure and the distribution of section durations are incorporated by the Bayesian extension. The experimental results showed that the proposed method achieved segmentation and clustering accuracies significantly better than the representative existing methods.

For future work, we plan to refine the model to incorporate the aspect of novelty and to deal with more hierarchies [16] because music has a hierarchical structure, from motive and phrase to section and section group [36]. Our unsupervised learning approach is complementary to another approach based on deep discriminative models [6–8]. A promising direction is to integrate these models into a variational autoencoding framework [22].

6. ACKNOWLEDGEMENTS

This work is supported in part by JST ACCEL No. JPMJAC1602, JSPS KAKENHI Nos. 16H01744, 19K20340, and 19H04137, and the Kyoto University Foundation.

7. REFERENCES

[1] J. Paulus, M. Müller, and A. Klapuri. State of the art report: Audio-based music structure analysis. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 625–636, 2010.

[2] J. Foote. Automatic audio segmentation using a measure of audio novelty. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 452–455, 2000.

[3] K. Jensen. Multiple scale music segmentation using rhythm, timbre, and harmony. *EURASIP Journal on Applied Signal Processing*, 2007(1):159–159, 2007.

[4] J. Serrà, M. Müller, P. Grosche, and J. Arcos. Unsupervised detection of music boundaries by time series structure features. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 1613–1619, 2012.

[5] G. Peeters and V. Bisot. Improving music structure segmentation using lag-priors. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 337–342, 2014.

[6] K. Ullrich, J. Schlüter, and T. Grill. Boundary detection in music structure analysis using convolutional neural networks. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 417–422, 2014.

[7] T. Grill and J. Schlüter. Music boundary detection using neural networks on combined features and two-level annotations. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 531–537, 2015.

[8] A. Maezawa. Music boundary detection based on a hybrid deep model of novelty, homogeneity, repetition and duration. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 206–210, 2019.

[9] G. Sargent, F. Bimbot, and E. Vincent. Estimating the structural segmentation of popular music pieces under regularity constraints. *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, 25(2):344–358, 2017.

[10] F. Kaiser and G. Peeters. A simple fusion method of state and sequence segmentation for music structure discovery. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 257–262, 2013.

[11] J.-J. Aucouturier and M. Sandler. Segmentation of musical signals using hidden Markov models. In *Audio Engineering Society (AES) Convention*, pages 1–8, 2001.

[12] M. Cooper and J. Foote. Summarizing popular music via structural similarity analysis. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 127–130, 2003.

[13] M. M. Goodwin and J. Laroche. A dynamic programming approach to audio segmentation and speech/music discrimination. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 309–312, 2004.

[14] H. Grohganz, M. Clausen, N. Jiang, and M. Müller. Converting path structures into block structures using eigenvalue decompositions of self-similarity matrices. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 209–214, 2013.

[15] O. Nieto and T. Jehan. Convex non-negative matrix factorization for automatic music structure identification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 236–240, 2013.

[16] B. McFee and D. P. W. Ellis. Analyzing song structure with spectral clustering. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 405–410, 2014.

[17] L. Ren, D. Dunson, S. Lindroth, and L. Carin. Dynamic nonparametric Bayesian models for analysis of music. *Journal of the American Statistical Association (JASA)*, 105(490):458–472, 2008.

[18] L. Barrington, A. B. Chan, and G. Lanckriet. Modeling music as a dynamic texture. *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, 18(3):602–612, 2010.

[19] M. Goto. A chorus section detection method for musical audio signals and its application to a music listening station. *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, 14(5):1783–1794, 2006.

[20] J. Paulus and A. Klapuri. Music structure analysis using a probabilistic fitness measure and a greedy search algorithm. *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, 17(6):1159–1170, 2009.

[21] T. Cheng, J. B. L. Smith, and M. Goto. Music structure boundary detection and labelling by a deconvolution of path-enhanced self-similarity matrix. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 106–110, 2018.

[22] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, pages 1–14, 2014.

[23] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer. madmom: A new python audio and music signal processing library. In *ACM International Conference on Multimedia (ACMMM)*, pages 1174–1178, 2016.

[24] A. Allahverdyan and A. Galstyan. Comparative analysis of Viterbi training and maximum likelihood estimation for HMMs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1674–1682, 2011.

[25] S.-Z. Yu. Hidden semi-Markov models. *Artificial Intelligence*, 174(2):215–243, 2010.

[26] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Popular, classical and jazz music databases. In *International Conference on Music Information Retrieval (ISMIR)*, pages 287–288, 2002.

[27] M. Goto. AIST annotation for the RWC music database. In *International Conference on Music Information Retrieval (ISMIR)*, pages 359–360, 2006.

[28] Y. Wu and W. Li. Automatic audio chord recognition with MIDI-trained deep feature and BLSTM-CRF sequence decoding model. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 27(2):355–366, 2019.

[29] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto. librosa: Audio and music signal analysis in python. In *Python in Science Conference*, pages 18–24, 2015.

[30] C.-I. Wang and G. J. Mysore. Structural segmentation with the variable Markov oracle and boundary adjustment. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 291–295, 2016.

[31] O. Nieto and J. P. Bello. Systematic exploration of computational music structure research. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[32] A. F. Ehmann, M. Bay, J. S. Downie, I. Fujinaga, and D. De Roure. Music structure segmentation algorithm evaluation: Expanding on MIREX 2010 analyses and datasets. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 561–566, 2011.

[33] D. Turnbull, G. Lanckriet, E. Pampalk, and M. Goto. A supervised approach for detecting boundaries in music using difference features and boosting. In *International Conference on Music Information Retrieval (ISMIR)*, pages 51–54, 2007.

[34] M. Levy and M. Sandler. Structural segmentation of musical audio by constrained clustering. *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, 16(2):318–326, 2008.

[35] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis. mir_eval: A transparent implementation of common MIR metrics. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[36] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, 1983.

TOWARDS MEASURING INTONATION QUALITY OF CHOIR RECORDINGS: A CASE STUDY ON BRUCKNER'S LOCUS ISTE

Christof Weiß, Sebastian J. Schlecht, Sebastian Rosenzweig, Meinard Müller
International Audio Laboratories Erlangen, Germany
christof.weiss@audiolabs-erlangen.de

ABSTRACT

Unaccompanied vocal music is a central part of Western art music, yet it requires excellent skills for singers to achieve proper intonation. In this paper, we analyze intonation deficiencies by introducing an intonation cost measure that can be computed from choir recordings and may help to assess the singers' intonation quality. With our approach, we measure the deviation between the recording's local salient frequency content and an adaptive reference grid based on the equal-tempered scale. The adaptivity introduces invariance of the local intonation measure to global intonation drifts. In our experiments, we compute this measure for several recordings of Anton Bruckner's choir piece *Locus Iste*. We demonstrate the robustness of the proposed measure by comparing scenarios of different complexity regarding the availability of aligned scores and multi-track recordings, as well as the number of singers per part. Even without using score information, our cost measure shows interesting trends, thus indicating the potential of our method for real-world applications.

1. INTRODUCTION

Unaccompanied vocal music constitutes the nucleus of Western art music and the starting point of polyphony's evolution. Despite an increasing number of studies [1, 4, 5, 8–11, 16, 17, 21] dating back to the 1930s [29], many facets of polyphonic a cappella singing are yet to be explored and understood. A central challenge of a cappella singing is the adjustment of pitch in order to stay in tune relative to the fellow singers. Even if choirs achieve locally good intonation, they may suffer from intonation drifts slowly evolving over time [8–10, 15–17, 21, 23]. Thus, one has to deal with different intonation issues that refer to harmonic (or *vertical*) and melodic (or *horizontal*) intonation. In Fig. 1, we show how these different aspects of intonation quality may be visualized separately. Our schematic example illustrates the assessment of note-wise pitch deviations (color-coded) in the presence of a global pitch drift. Fig. 1a shows the idealized situation, where each note-wise pitch lies on a fixed reference grid. In Fig. 1b, all pitches are sharp (too high) compared to the same reference grid. The intonation quality, however, should be considered equivalent in both cases. Fig. 1c illustrates a different situation with downward intonation drift. Using a fixed reference grid, the deviations gradually accumulate. To compensate for this, one can use an adaptive reference grid [12, 15, 31] so that the vertical intonation quality is separated from the horizontal intonation drift. The residual pitch deviations—relative to the adaptive grid—only refer to vertical intonation problems (Fig. 1d), which are in the focus of our analysis.

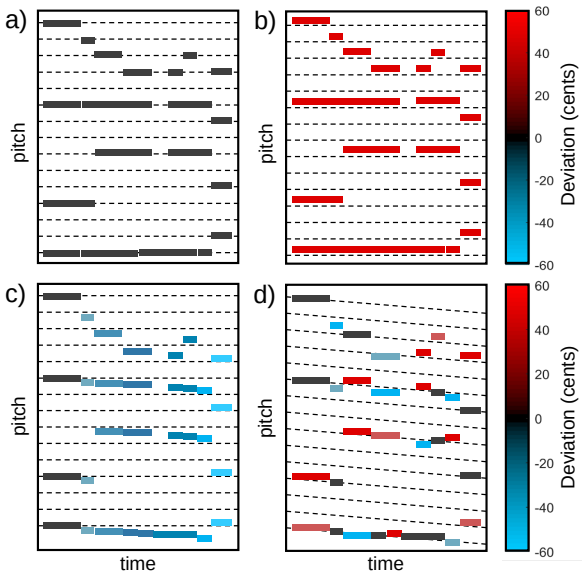


Figure 1. Note-wise analysis for polyphonic music. (a) Global intonation matching a fixed reference grid. (b) Global intonation higher than a fixed reference grid. (c) Intonation drift shown against a fixed reference grid. (d) Intonation drift with deviations from an adaptive grid.

In this paper, we propose an intonation cost measure that can be computed from choir recordings and may help to assess the singers' intonation quality. Developing such a measure encompasses two central challenges: (i) accurate estimation of the local salient frequency content from a choir recording, and, (ii) reliable measurement of intonation quality corresponding to human perception on the one hand and to music theory on the other hand.

Concerning the estimation of the local salient frequency content (i), recordings of polyphonic choir pieces constitute extremely difficult scenarios. Often, the different

parts of a musical composition are highly correlated both in rhythm (joint on- and offsets) and harmony (overlap of partials). In the case of a *mix* recording (single-track), this leads to overlaps in time–frequency representations, which makes the estimation of fundamental frequencies (F0s) [3] or partial tracking [25] hard. On the other hand, capturing intonation at the sub-semitone level requires high frequency resolutions. One can leverage these problems using dedicated recording scenarios where singers are isolated acoustically [5] or recorded sequentially [4]. Alternatively, special devices such as Larynx microphones [6, 16, 28] or additional score information [9] can help to simplify the F0-estimation problem [18, 27]. In Section 3.2, we detail the strategy used for this paper's experiments.

For assessing intonation quality (ii), we aim towards developing a robust intonation cost measure. Ideally, a high value of this measure indicates passages of low intonation quality. Hereby, intonation quality may relate to human perception such as the measure proposed in [30] based on psychometric curves [26]. On the other hand, intonation quality may be guided by music theory or historical performance practice [9]. In particular, choirs often aim for just intonation, which involves complex adjustment strategies according to the harmonic context [10]. In contrast to such ideas, we follow a simplified approach based on 12-tone equal temperament (12-TET). Even though 12-TET is not considered to be the ideal intonation practice for Western choir performance, it is a first approximation and can provide useful feedback [15]. As a major advantage, our strategy estimates the sub-semitone intonation quality for *any* type of notated chord irrespective of its *harmonic* consonance—in contrast to other methods [30] that measure a mixture of harmonic consonance and intonation.

As our main contribution, we propose a 12-TET-based intonation cost measure (Section 2). Inspired by prior work [15, 24], we accumulate the overall deviation of frequency components from an adaptive 12-TET grid weighted by their corresponding amplitudes. For testing this measure, we compiled a small but diverse dataset of Anton Bruckner's Graduale *Locus Iste* using different performances (Section 3). We evaluate the robustness of our method by comparing scenarios of varying complexity regarding the availability of aligned scores and multi-track recordings (Section 4.1). Finally, we apply our method to different performances and show its benefit for assessing the overall intonation quality of a recording (Section 4.2). Section 5 concludes the paper and gives an outlook on future work and practical applications.

2. MEASURING INTONATION QUALITY

In the following, we describe the computation of an intonation cost measure based on frequency deviations from a 12-tone equal-temperament (12-TET) grid.

2.1 Intonation Cost Measure

The proposed measure operates on a set of N frequency components $\mathcal{P} := \{(f_1, a_1), \ldots, (f_N, a_N)\}$, where each tuple $(f_n, a_n) \in \mathcal{P}$ denotes the frequency f_n and amplitude a_n of an individual component.

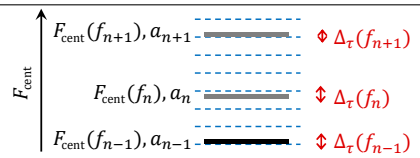


Figure 2. Grid deviations $\Delta_\tau(f_n)$ in cents of frequency components f_n (solid gray lines) from a 12-TET grid (dashed blue lines) shifted by τ. The corresponding amplitudes a_n are indicated by the grayscale colors.

First, we convert a given frequency f in Hertz (Hz) to cents by

$$F_{\text{cent}}(f) := 1200 \cdot \log_2\left(\frac{f}{f_0}\right), \tag{1}$$

where $f_0 := 55\,\text{Hz}$ is an arbitrary but fixed reference frequency. We compute the deviation of the frequency component f from a 12-TET grid

$$\Delta_\tau(f) := \min_{i \in \mathbb{Z}} |F_{\text{cent}}(f) - \tau - 100i|, \tag{2}$$

where $\tau \in [-50, 50[$ specifies the overall grid shift in cents, see Fig. 2. Applying a Gaussian-like function to the grid deviation $\Delta_\tau(f)$, we define the intonation cost (IC) Θ_τ as

$$\Theta_\tau(\mathcal{P}) := \frac{\sum_{(f,a)\in\mathcal{P}} a\left(1 - \exp\left(-\frac{\Delta_\tau^2(f)}{2\sigma^2}\right)\right)}{\sum_{(f,a)\in\mathcal{P}} a}, \tag{3}$$

where the deviations are weighted and then normalized by the corresponding amplitudes. Due to the normalization, we obtain $\Theta_\tau(\mathcal{P}) \in [0, 1]$ where $\Theta_\tau(\mathcal{P}) = 1$ indicates the maximal IC. The parameter σ adjusts the cost for deviating from the grid. As suggested by [24], we choose a value of $\sigma = 16$ cents. To obtain invariance to pitch drifts, i.e., variation of the reference frequency, we choose the grid shift τ in an adaptive way so that the IC is minimized:

$$\tau^* := \operatorname*{arg\,min}_{\tau \in [-50, 50[} \Theta_\tau(\mathcal{P}). \tag{4}$$

We then define the intonation cost Θ as

$$\Theta(\mathcal{P}) := \Theta_{\tau^*}(\mathcal{P}). \tag{5}$$

For instance, in a scenario where a choir performs with good local intonation but is affected by a pitch drift, τ^* slowly changes over time while Θ is constantly small.

2.2 Example with Synthetic Signals

In the following, we illustrate the properties of the IC measure $\Theta(\mathcal{P})$ by means of synthetic examples. To this end, we define a harmonic tone $x_f : \mathbb{R} \to \mathbb{R}$ with K partials and fundamental frequency f as

$$x_f(t) := \sum_{k=1}^{K} a_k \cdot \sin(2\pi k f t), \tag{6}$$

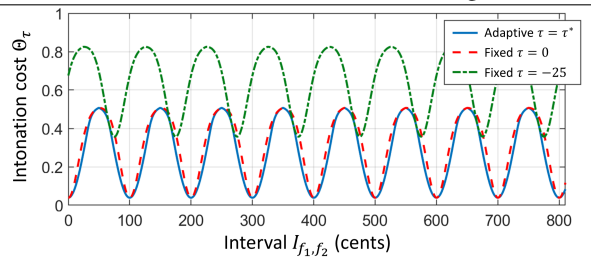


Figure 3. ICs Θ_τ for two harmonic tones x_{duad} with an interval of size I_{f_1,f_2} in cents and three grid shifts: adaptive grid τ^*, fixed grid $\tau = 0$, and, fixed grid $\tau = -25$. The F0 of the lower tone is fixed at $f_1 = 220$ Hz.

where t denotes time and

$$a_k := s^{k-1} \tag{7}$$

denotes the geometrically decaying partial amplitudes for some $s \in [0, 1]$ and $k = 1, \ldots, K$. Thus, the set of frequency components of signal x_f is

$$\mathcal{P}[x_f] := \{(f, a_1), (2f, a_2), \ldots, (Kf, a_K)\}. \tag{8}$$

Let $x_{\text{duad}} := x_{f_1} + x_{f_2}$ be the sum of two harmonic tones whose fundamental frequencies differ by the interval

$$I_{f_1,f_2} := |F_{\text{cent}}(f_2) - F_{\text{cent}}(f_1)| \tag{9}$$

given in cents. Consequently, $\mathcal{P}[x_{\text{duad}}] = \mathcal{P}[x_{f_1}] \cup \mathcal{P}[x_{f_2}]$.

Fig. 3 shows $\Theta(\mathcal{P}[x_{\text{duad}}])$ for two harmonic tones with $K = 16$ and $s = 0.6$ for different interval sizes I_{f_1,f_2}. The lower fundamental frequency is set to $f_1 = 220$ Hz such that $\Delta_0(f_1) = 0$. $\Theta(\mathcal{P}[x_{\text{duad}}],)$ is minimal for I_{f_1,f_2} being an integer multiple of 100 cents. However, $\Theta(\mathcal{P}[x_{\text{duad}}])$ does not reach zero as some partial frequencies $k \cdot f$ of a harmonic tone do not lie on the 12-TET grid even if the fundamental frequency f does. For example, the third partial $3f_1 = 660$ Hz leads to $\Delta_0(3f_1) \approx 2$ cents and the fifth partial $5f_1 = 1100$ Hz leads to $\Delta_0(5f_1) \approx 14$ cents. Since the minimal values are close to zero, this effect is small for a partial decay of $s = 0.6$. On the other hand, even a quarter tone interval $I_{f_1,f_2} = 50$ cents does not lead to the maximal IC of 1, since the grid deviation $\Delta_{\tau^*}(k \cdot f_1) \approx \Delta_{\tau^*}(k \cdot f_2) \approx 25$ cents for $\tau^* = 25$ cents. Fig. 3 further shows that the IC of a fixed grid shift $\tau = 0$ is similar to the adaptive grid τ^*, while a fixed shift $\tau = -25$ significantly increases the overall IC. It is important to note that the IC with adaptive grid shift τ^* is invariant to the choice of f_1 while a fixed grid shift is not. Further, the minimal and maximal values depend on the amplitude decay s and on the Gaussian width σ. Since the IC measure relates to a 12-TET grid, the IC curve is periodic in interval size with a period of 100 cents. As a consequence, each musical interval is only evaluated by its deviation from the 12-TET scale—regardless of its harmonic consonance quality.

In Fig. 4, we expand the previous example to three harmonic tones $x_{\text{triad}} := x_{f_1} + x_{f_2} + x_{f_3}$ with $f_1 \leq f_2 \leq f_3$. For instance, the intervals $I_{f_1,f_2} = 400$ cents, $I_{f_2,f_3} = 300$ cents describe an equal-tempered major triad. The colors in Fig. 4 indicate $\Theta(\mathcal{P}[x_{\text{triad}}])$ with respect to the

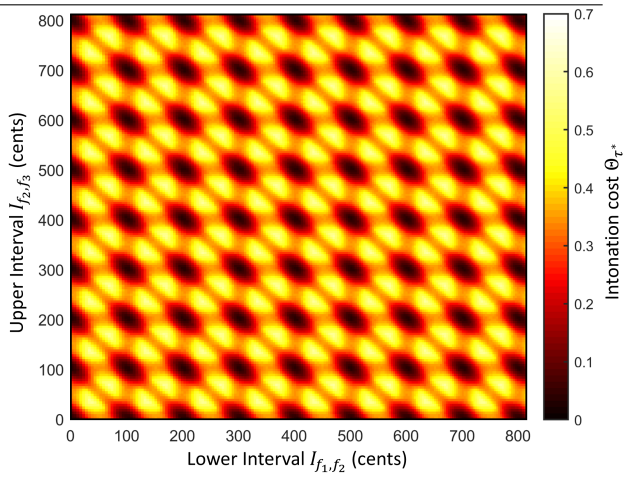


Figure 4. IC Θ for three harmonic tones x_{triad}. The plot axes indicate the size of the lower and upper interval in cents, I_{f_1,f_2} and I_{f_2,f_3} respectively.

lower and upper intervals, I_{f_1,f_2} and I_{f_2,f_3}, respectively. Similar to Fig. 3, we observe a periodic structure with period 100 cents as the IC is invariant to the musical intervals. Thus, the measure is equally suited for estimating the intonation quality of both consonant and dissonant triads.

3. EXPERIMENTAL SCENARIO

This section describes the experimental scenario for applying our intonation cost measure to choir recordings.

3.1 Dataset

We compiled a small but diverse dataset of performances of Anton Bruckner's Gradual *Locus iste* WAB 23 (see Fig. 5). This choir piece is in Latin and lasts approximately three minutes. It is musically interesting, contains several melodic and harmonic challenges—such as the highly chromatic middle part—but also harmonically clear passages, and covers a large part of each voice's tessitura.

Central to this dataset is a publicly available [1] multitrack recording from the Choral Singing Dataset (CSD) [4]. The performance of 16 singers from the semi-professional Anton Bruckner choir (Barcelona) was recorded in a studio setting. The four musical parts—soprano, alto, tenor, and bass—were recorded sequentially using directional hand-held microphones. Rhythmic and harmonic synchronization was ensured by a conducting video and an acoustic reference (MIDI version of the piece). Due to this recording scenario, the individual singers' tracks exhibit a small amount of bleeding from other singers of the same part (e. g., soprano 2, 3, and, 4 slightly bleed into soprano 1 track). Interactive intonation or adaptation across musical parts was not possible since the parts were recorded in isolation and each singer listened to the reference MIDI signal while singing—this also prevented substantial pitch drifts. The restricted interaction limits the usability of the data to study intonation and

[1] https://zenodo.org/record/1319597#.XJor8ShKhaR

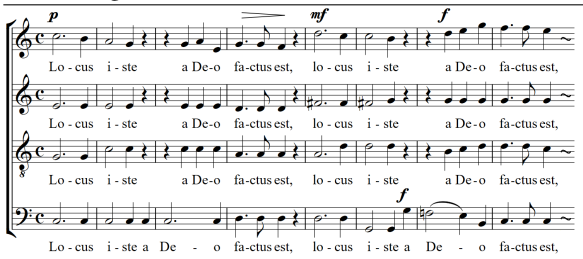


Figure 5. Anton Bruckner, *Locus iste* WAB 23, beginning.

adaptation phenomena in choir performances. Nevertheless, since this paper focuses on measurement strategies, the multi-track data provides an excellent resource. To address subsets of the multi-track recording, we refer to the signals of the four soprano voices as x_{S1}, x_{S2}, x_{S3}, and x_{S4}, and to the first voices of the alto, tenor, and bass as x_{A1}, x_{T1}, and x_{B1}, respectively. We denote the mixed signal of the first voices of each part as x_{SATB1} and the mixed signal of all 16 voices as x_{SATB}. We further manipulated the original tracks x^{orig} with the digital audio correction software Melodyne [2] to augment the dataset: x^{note} is generated by quantizing the median pitch of each note event onto a 12-TET grid with reference frequency $f_0 = 55$ Hz. x^{fine} is generated by quantizing the complete F0-trajectory onto the 12-TET grid. To assist our analysis, we use additional score information from the aligned MIDI file.

In addition to this multi-track recording, we collected several commercial [3] and freely available [4] performances of the piece. As a reference for the real audio measurements, we sonified the piece with harmonic tones (Section 2.2) using random pitch deviations sampled from Gaussian distributions with different standard deviations.

3.2 Measuring Frequency Content

As discussed in Section 1, the extraction of salient frequency content from choir recordings is challenging. In the case of a mix recording, we have to blindly estimate all partial frequencies using a partial tracking algorithm such as [25]. For choir music, partial tracking can be simplified as the singing voice's partial frequencies are located quite precisely at integer multiples of an estimated F0. To estimate F0-trajectories for the CSD, we can use the individual tracks of the multi-track recording. Due to the bleeding of other voices from the same part, traditional F0-estimation techniques [7, 20] may have problems. Therefore, we use a salience-based method similar to Melodia [27]. We compute a log-frequency representation using instantaneous frequency estimation [2, 14, 27] and binning with a resolution of $1\,\mathrm{cent}$. Subsequently, we estimate F0-trajectories using dynamic programming [22]. As post-processing, we apply median filtering with a filter length of 101 frames and downsample by a factor of 50 obtaining trajectories with a time resolution of $290\,\mathrm{ms}$.

[2] `https://www.celemony.com`

[3] Philharmonia Vocalensemble Stuttgart (Profil Medien 2006), Chor des Bayerischen Rundfunks (Decca 2012), Choir of St John's College Cambridge (Classic Mania 2007), NDR Chor Hamburg (Carus 2015)

[4] Internet Archive, `https://archive.org/details/LocusIste`

In the case of the CSD, we can exploit additional score information from the aligned MIDI file [13]. We restrict the F0-estimation to rectangular time–frequency regions ("constraint regions") derived from onsets, durations and center frequencies of the aligned MIDI notes, including a frequency tolerance of $\pm 60\,\mathrm{cents}$ around the center frequencies (according to 12-TET with reference 440 Hz). Such additional information is particularly helpful when estimating F0-trajectories from a mix recording. [5] Especially, the constraint regions prevent the common confusion of the F0 with higher partials.

4. RESULTS

In our experiments, we investigate the robustness of the proposed intonation measure for different scenarios of the CSD where either score information or multi-track recordings are not used (Section 4.1). Furthermore, we compare the measure's behaviour for different synthetic and real performances of *Locus Iste* (Section 4.2).

4.1 Local Analysis and Visualization

As a visual orientation, we show in Fig. 6a a piano roll representation of the entire piece, generated from the aligned MIDI file. Considering the subset of each part's first voice x_{S1}, x_{A1}, x_{T1}, and x_{B1}, we estimate F0-trajectories as described in Section 3.2 and compute the local, sub-semitone deviation of the F0-estimate from the corresponding MIDI reference. The deviations are color-coded with a range of $\pm 60\,\mathrm{cents}$. While there seems to be no significant global drift (which is not surprising because of the CSD's recording scenario, see Section 3.1), we observe a slight dominance of notes sung flat (negative deviation) except for the alto part, which is sharp more often.

The main results are shown in Figures 6b–g, which indicate IC values throughout each performance. We compute the IC measure $\Theta(\mathcal{P})$ of the set $\mathcal{P}$ comprising the frequencies (F0 and higher partials) and amplitudes from all parts that are, according to the aligned score, active in a frame. Assuming that the human voice's partials are harmonic, we calculate the first 16 partial frequencies from the measured F0-trajectories and extract the corresponding amplitudes from the log-frequency spectrogram. For computing Θ, we use the adaptive grid shift proposed in Section 2.1. To remove local outliers, we post-process the IC curves using a moving median filter with a length of 21 frames.

The blue, solid line in Fig. 6b shows the resulting IC curve for x^{orig}, computed from the individual tracks for the first voice of each part x_{S1}, x_{A1}, x_{T1}, and x_{B1} using score constraints. For silent regions (e. g., after 160 sec), the IC is zero since no constraint region is active. Due to the adaptive grid shift, the IC is small for monophonic passages where only one singer is active (see, e. g., the passage at 80 sec). For some of the consonant chords (e. g., at 110 sec), we observe low IC values of about 0.2. During the highly chromatic three-part passage (80–110 sec), the

[5] We do not use harmonic summation as in [27] to avoid smearing of other parts' partials into the constraint regions for mix recordings.

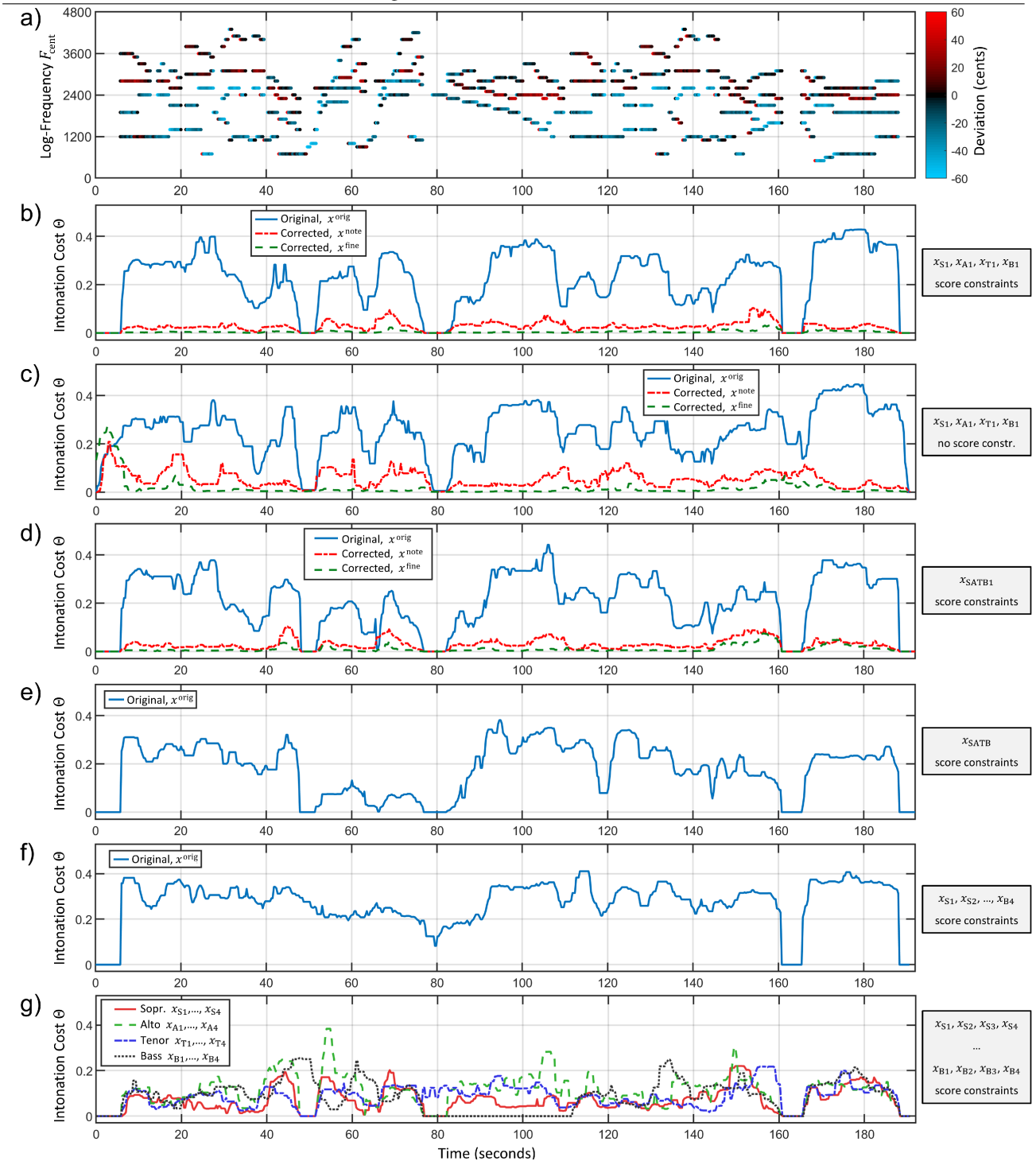


Figure 6. Intonation cost (IC) curves for different CSD versions of *Locus Iste*. (a) Piano roll representation with a log-frequency axis where C2 corresponds to 1200 cents, C3 to 2400 cents etc. The colors encode the deviation of the first voice of each part (x_{S1}, x_{A1}, x_{T1}, x_{B1}) from the MIDI reference. (b) IC curves for x_{S1}, x_{A1}, x_{T1}, x_{B1} with score constraints from four individual tracks. The blue curve corresponds to x^{orig}, the red curve to x^{note}, and the green dashed curve to x^{fine}. (c) IC curves as in (b) without score constraints. (d) IC curves for mixed signal x_{SATB1} with score constraints. (e) IC curve for mixed signal of all voices x_{SATB} with score constraints. (f) IC curve computed from all 16 individual tracks x_{S1}, x_{S2}, ..., x_{B4} with score constraints. (g) Individual IC curves for the four tracks of each part with score constraints.

IC increases, thus indicating that the singers have difficulties to stay in tune in this passage.

As a sanity check, we compare the results for x^{orig} to the the pitch-corrected versions x^{note} and x^{fine}. As we expected, the note-wise corrections x^{note} (red curve) obtain lower values than x^{orig}—with only minor peaks that may be caused by pitch fluctuations during a note event. If we correct such local fluctuations as in x^{fine} (green, dashed curve), the IC is almost constantly zero.

In Fig. 6c, we repeat the experiment of Fig. 6b without using score constraints. In this case, F0-estimation is less reliable and, in particular, confusions between F0 and higher partials may occur. However, since many partials lie on the same 12-TET grid as the F0 (octave-related partials) or very close to that (2 cents for fifth-related partials), the IC measure is largely invariant to such confusions. The high similarity between Fig. 6b and Fig. 6c confirms the IC measure's robustness to F0-extraction errors. Only for silent passages (e. g., after 160 sec or at the beginning), we obtain higher IC values due to erroneously extracted frequency components. [6] We conclude that our strategy is, in principle, applicable for any multi-track recording and does not necessarily require score information.

To test the applicability in absence of multi-track recordings, we compute the IC curve from the mix signal x_{SATB1} using score constraints (Fig. 6d). Due to the constraints, confusions with higher partials cannot occur, but partials of other parts may leak into the same constraint regions, thus affecting the estimated F0s and partials' amplitudes. The comparison of Fig. 6b and Fig. 6d indicates that such phenomena only slightly affect the IC measure.

Next, we measure the IC from the CSD's full recording x_{SATB} (all 16 voices). F0-estimation is more challenging since the four singers of each part contribute to the same constraint regions. The resulting ICs (Fig. 6e) exhibit slightly lower values than in previous cases. We assume that having several voices per part stabilizes the F0-estimation to some degree. Overall, we see a similar trend between Fig. 6d and Fig. 6e. This might be an effect of mutual influence between the singers of each part. To further investigate the multiple-singer effect, we show in Fig. 6f the IC curves computed from all 16 individual tracks $x_{\mathrm{S1}}, x_{\mathrm{S2}}, \ldots, x_{\mathrm{B4}}$. We obtain higher IC values than in Fig. 6e, especially for the monophonic passages (e. g., at 48 sec). Due to the score constraints, this must be caused by deviations between the singers of a part, sometimes denoted as *dispersion* [4]. To analyze this, we repeat the experiment for each part separately (Fig. 6g). The resulting curve supports our hypothesis since. For instance, the high value at around 48 sec (Fig. 6f) is mainly caused by the basses' dispersion (dotted curve in Fig. 6g).

4.2 Global Analysis of Different Performances

Since our experiments on the CSD have shown the robustness of our method even for mix recordings, we finally compare the global IC values of multiple synthetic and real performances of *Locus Iste*. We align the MIDI file to all performances [13], use the resulting constraint regions for extracting F0-trajectories [27], and compute the IC curves. In Fig. 7, we show the statistics of the entire curves over time (median, mean, and standard deviation). First, we report values for sonifications using harmonic tones as defined in Eq. (6). To simulate detuning, we shifted each note's fundamental frequency by a random value sampled from a Gaussian distribution with variance

[6] This problem could be leveraged using a suitable algorithm for voice activity detection [19] or on-/offset detection in vocal music [6].

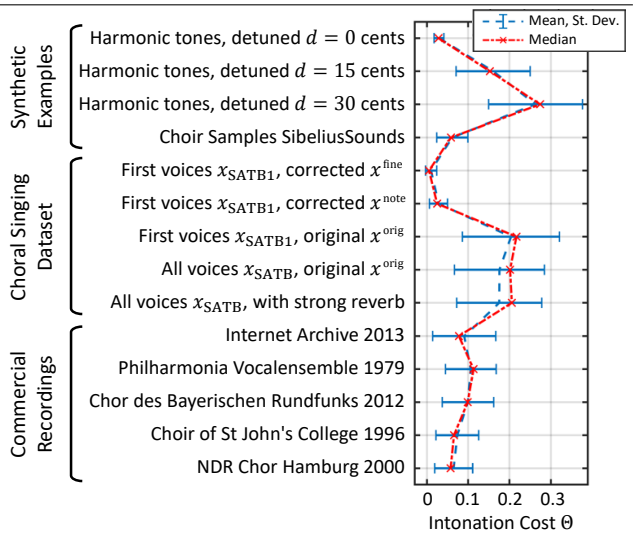


Figure 7. Statistics of the intonation cost measure Θ over full recordings of different type.

d^2. For $d = 0$ cents, the mean IC is almost zero. For $d = 15$ cents and $d = 30$ cents, the IC gradually increases as expected. The average IC of a sample-based sonification [7] is moderately higher than the rendition with ideal harmonic tones, which suggests that some choir effects such as dispersion are also synthesized. Both corrected versions of the CSD show very low IC values, with x^{fine} even lower than x^{note}. In contrast, the versions based on the original recording x^{orig} have high IC values, where the difference between first voices only (x_{SATB1}) and the full mix (x_{SATB}) as well as the effect of adding artificial reverberation is marginal. All commercial recordings performed by professional choirs obtain lower IC than the CSD recording. Several effects might contribute to this observation. Besides the singers' level of training, commercial recordings often feature larger choirs (60 singers or more) and long natural reverberation (recorded in churches). In line with the authors' subjective judgment, the recording by NDR Chor Hamburg exhibits the best intonation quality according to the IC measure. [8] Overall, this experiment indicates that the proposed IC measure may serve as a first indicator for a choir recording's global intonation quality.

5. CONCLUSIONS

In this paper, we proposed a strategy for measuring the intonation quality of choir recordings. Although robust extraction of salient frequencies is challenging, the measure produced meaningful and reliable results once multi-track recordings are available or score information can be utilized. Even though the insights from such a measure are limited, it might be a first indicator for the overall intonation quality and, thus, could be useful for choir singers or choir directors in performances and rehearsals.

[7] Using Sibelius Sounds, see `https://www.avid.com/sibelius`

[8] A 30-seconds mp3 thumbnail of this recording is available at `https://www.carusmedia.com/images-intern/medien/80/8346600/8346600.010s.t1_010.mp3`

Acknowledgments: This work was supported by the German Research Foundation (DFG MU 2686/12-1). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS. The authors want to thank Helena Cuesta and colleagues from UPF Barcelona for creating and publishing the Choral Singing Dataset.

6. REFERENCES

[1] Per-Gunnar Alldahl. *Choral Intonation*. Gehrmans Musikförlag, 1990.

[2] François Auger and Patrick Flandrin. Improving the readability of time-frequency and time-scale representations by the reassignment method. *IEEE Transactions on Signal Processing*, 43(5):1068–1089, 1995.

[3] Mert Bay, Andreas F. Ehmann, and J. Stephen Downie. Evaluation of multiple-f0 estimation and tracking systems. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 315–320, Kobe, Japan, 2009.

[4] Helena Cuesta, Emilia Gómez, Agustín Martorell, and Felipe Loáiciga. Analysis of intonation in unison choir singing. In *Proceedings of the International Conference of Music Perception and Cognition (ICMPC)*, pages 125–130, Graz, Austria, 2018.

[5] Jiajie Dai and Simon Dixon. Analysis of interactive intonation in unaccompanied SATB ensembles. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 599–605, Suzhou, China, 2017.

[6] Sara D'Amario, Helena Daffern, and Freya Bailes. A new method of onset and offset detection in ensemble singing. *Logopedics Phoniatrics Vocology*, 2018.

[7] Alain de Cheveigné and Hideki Kawahara. YIN, a fundamental frequency estimator for speech and music. *Journal of the Acoustical Society of America (JASA)*, 111(4):1917–1930, 2002.

[8] Johanna Devaney. *An Empirical Study of the Influence of Musical Context on Intonation Practices in Solo Singers and SATB Ensembles*. PhD thesis, McGill University, Montreal, Canada, 2011.

[9] Johanna Devaney and Daniel P. W. Ellis. An empirical approach to studying intonation tendencies in polyphonic vocal performances. *Journal of Interdisciplinary Music Studies*.

[10] Johanna Devaney, Michael I. Mandel, and Ichiro Fujinaga. A study of intonation in three-part singing using the automatic music performance analysis and comparison toolkit (AMPACT). In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 511–516, Porto, Portugal, 2012.

[11] Johanna Devaney, Jonathan Wild, and Ichiro Fujinaga. Intonation in solo vocal performance: A study of semitone and whole tone tuning in undergraduate and professional sopranos. In *Proceedings of the International Symposium on Performance Science*, pages 219–224, Toronto, Canada, 2011.

[12] Karin Dressler and Sebastian Streich. Tuning frequency estimation using circular statistics. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 357–360, Vienna, Austria, 2007.

[13] Sebastian Ewert, Meinard Müller, and Peter Grosche. High resolution audio synchronization using chroma onset features. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1869–1872, Taipei, Taiwan, April 2009.

[14] J. L. Flanagan and R. M. Golden. Phase vocoder. *Bell System Technical Journal*, 45:1493–1509, 1966.

[15] Volker Gnann, Markus Kitza, Julian Becker, and Martin Spiertz. Least-squares local tuning frequency estimation for choir music. In *Proceedings of the Audio Engineering Society (AES) Convention*, New York City, USA, 2011.

[16] David M. Howard. Intonation drift in a capella soprano, alto, tenor, bass quartet singing with key modulation. *Journal of Voice*, 21(3):300–315, 2007.

[17] David M. Howard, Helena Daffern, and Jude Brereton. Four-part choral synthesis system for investigating intonation in a cappella choral singing. *Logopedics Phoniatrics Vocology*, 38(3):135–142, 2013.

[18] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello. Crepe: A convolutional representation for pitch estimation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 161–165, Calgary, Canada, 2018.

[19] Bernhard Lehner, Jan Schlüter, and Gerhard Widmer. Online, loudness-invariant vocal detection in mixed music signals. *IEEE/ACM Transactions on Audio, Speech & Language Processing*, 26(8):1369–1380, 2018.

[20] Matthias Mauch and Simon Dixon. pYIN: A fundamental frequency estimator using probabilistic threshold distributions. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 659–663, Florence, Italy, 2014.

[21] Matthias Mauch, Klaus Frieler, and Simon Dixon. Intonation in unaccompanied singing: Accuracy, drift, and a model of reference pitch memory. *Journal of the Acoustical Society of America*, 136(1):401–411, 2014.

[22] Meinard Müller. *Fundamentals of Music Processing*. Springer Verlag, 2015.

[23] Meinard Müller, Peter Grosche, and Frans Wiering. Automated analysis of performance variations in folk song recordings. In *Proceedings of the International Conference on Multimedia Information Retrieval (MIR)*, pages 247–256, Philadelphia, Pennsylvania, USA, 2010.

[24] Tomoyasu Nakano, Masataka Goto, and Yuzuru Hiraga. An automatic singing skill evaluation method for unknown melodies using pitch interval accuracy and vibrato features. In *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 1706–1709, Pittsburgh, PA, USA, 2006.

[25] Julian Neri and Philippe Depalle. Fast partial tracking of audio with real-time capability through linear programming. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 326–333, Aveiro, Portugal, 2018.

[26] Reinier Plomp and Willem Johannes Maria Levelt. Tonal consonance and critical bandwidth. *Journal of the Acoustical Society of America*, 38(4):548–560, 1965.

[27] Justin Salamon and Emilia Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1759–1770, 2012.

[28] Frank Scherbaum. On the benefit of larynx-microphone field recordings for the documentation and analysis of polyphonic vocal music. *Proceedings of the International Workshop Folk Music Analysis*, pages 80–87, 2016.

[29] Carl Emil Seashore. *Objective Analysis of Musical Performance*, volume 4 of *Studies in the Psychology of Music*. University of Iowa Press, Iowa City, USA, 1936.

[30] William A. Sethares. Local consonance and the relationship between timbre and scale. *Journal of the Acoustical Society of America*, 94(3):1218–1228, 1993.

[31] Simon Waloschek and Aristotelis Hadjakos. Driftin' down the scale: Dynamic time warping in the presence of pitch drift and transpositions. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 630–636, Paris, France, 2018.

GUITAR TABLATURE ESTIMATION WITH A CONVOLUTIONAL NEURAL NETWORK

Andrew Wiggins, Youngmoo Kim
Drexel University, Dept. of Electrical and Computer Engineering
{awiggins, ykim}@drexel.edu

ABSTRACT

Guitar tablature is a popular notation guitarists use to learn and share music. As it stands, most tablatures are created by an experienced guitarist taking the time and effort to annotate a song. As the process is time consuming and requires expertise, we are interested in automating this task. Previous approaches to automatic tablature transcription break the problem into two steps: 1) polyphonic pitch estimation, followed by 2) tablature fingering arrangement. Using a convolutional neural network (CNN) model, we can jointly solve both steps by learning a mapping directly from audio data to tablature. The model can simultaneously leverage physical playability constraints and differences in string timbres implicit in the data to determine the actual fingerings being used by the guitarist. We propose TabCNN, a CNN for estimating guitar tablature from audio of a solo acoustic guitar performance. We train and test our network using microphone recordings from the GuitarSet dataset [24], and TabCNN outperforms a state-of-the-art multipitch estimation algorithm. We also introduce a set of metrics to evaluate guitar tablature estimation.

1. INTRODUCTION

Given the popularity of the guitar as an instrument for both professional musicians and amateur hobbyists, there have been numerous previous works addressing the problem of automatic guitar transcription. Automatic guitar transcription is a task which aims to generate a symbolic transcription instructing a guitarist how to perform, given an audio recording of a guitar performance. In general, the task of polyphonic transcription is challenging, due to the fact that when multiple different pitches sound at once, their overtones may overlap in the frequency domain, making it hard to tell which pitches are sounding. Despite the narrowed focus on a single instrument, transcription of solo guitar audio remains challenging. Since it has six strings, the guitar can produce up to six pitches at a given time. The guitar is also capable of producing a wide variety of timbres, stemming from differences in guitar models, guitar strings, audio effects, and strumming, plucking, and picking styles.

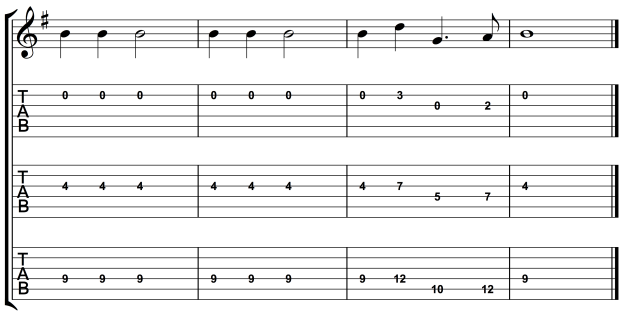


Figure 1. Multiple fingerings (3 bottom staves) can be used to play a given score (top staff) on guitar. These are just 3 of many possible fingerings. In each tablature staff, the horizontal lines represent the 6 guitar strings, and numbers on them indicate the fret on that string to be activated.

Perhaps the greatest challenge in automatic guitar transcription, however, arises from the symbolic representation of guitar music. Rather than using score notation, guitarists commonly use tablature notation to compose, share, and learn music. While music scores display the arrangements of pitches in time, tablature notation additionally indicates which guitar strings and positions along the fretboard were activated to produce the sounding pitches. On guitar, most notes can be played in numerous locations along the fretboard. Figure 1 provides an example of this. These identical pitches played in different locations differ in timbre and can give different characteristics to an overall performance. In order to effectively transcribe a guitar performance from audio, a system needs to estimate both the pitches and fingerings changing over the course of the performance.

Previous works have approached automatic tablature transcription by splitting the problem into two separate steps [5, 6, 25, 26]. First, the systems perform a multipitch estimation on the audio, which determines the set of pitches sounding over the course of the audio. Then, using the estimated pitches, a tablature is arranged, usually by choosing a fingering that maximizes physical ease of play. These methods are designed to output a valid tablature given the detected pitches, but not necessarily the true fingering used by the guitarist during the performance.

In this paper we propose TabCNN [1], a guitar tablature estimation system which outputs the actual fingerings used

[1] Source code: https://github.com/andywiggins/tab-cnn

by the guitarist at the frame-level from audio of a solo performance on a standard 6-string acoustic guitar. The proposed system uses a convolutional neural network (CNN) to learn a direct mapping from audio signal to guitar tablature.

The next section provides background on some related work in automatic guitar transcription and the use of CNNs in music information retrieval. In Section 3 we outline our methodology, including the dataset, preprocessing, and proposed system architecture. In Section 4, we evaluate the proposed system for multipitch estimation and tablature estimation. We further discuss the system performance in Section 5. Finally, in Section 6 we give our conclusions and suggest future work.

2. RELATED WORK

2.1 Automatic Tablature Transcription

Several previous works have addressed the problem of automatic tablature transcription by performing multipitch estimation and then arranging tablature by optimizing the physical ease of play. Burlet and Fujinaga outlined a framework for a guitar transcription web application [5]. Their framework combines a preexisting polyphonic transcription algorithm proposed by Zhou and Reiss [28] with a novel guitar tablature arrangement algorithm that creates a directed acyclic weighted graph of string-fret combinations and finds an optimal path using the A* search algorithm. In [6], Burlet and Hindle utilized the aforementioned tablature arrangement algorithm in conjunction with a novel multipitch estimation algorithm, using deep belief networks to learn framewise pitch estimates from the short-time Fourier transform. Yazawa et al. used latent harmonic allocation (LHA) for multipitch estimation and arranged tablature by filtering the LHA results based on a set of spatial and temporal playability constraints [26]. In [25], the authors applied knowledge of each guitarist's proficiency to further filter the pitches estimated from LHA and generate a sensible tablature arrangement.

Few previous approaches to automatic tablature transcription that we found seek to learn the true fingering used by the guitarist. In [3], Barbancho et al. used peak-picking from the magnitude spectrum to determine fundamentals and partials for candidate pitches and then analyzed the inharmonicity of the partials to determine the most likely string each pitch was played on. The system's performance on "free chords" – i.e., not recognition of predetermined chords, or strictly monophonic playing – is respectable, but is limited to a maximum of 4 pitches sounding simultaneously. In [13], Kehling et al. proposed a system that applies the Blind Harmonic Adaptive Decomposition algorithm proposed in [7] for multipitch estimation. After aggregating framewise pitch estimates into note estimates, their system applies Support Vector Machines to classify various performance parameters, including the guitar string each note was played on. The authors show good performance for this system for notewise multipitch estimation and guitar string estimation, but they do not evaluate the system directly for framewise tablature estimation.

2.2 Related Tasks

There have been a number of works that tackle different problems related to automatic guitar transcription. For guitar chord recognition, Barbancho et al. used a hidden Markov model (HMM) to transcribe guitar chords and fingerings from acoustic features [2]. In [12], Humphrey and Bello approached chord recognition using a convolutional neural network model to output tablature, and this model was trained using a pop music dataset, rather than audio of isolated guitar performances. Hrybyk and Kim used video data in conjunction with audio of solo guitar performances to recognize guitar chords [11].

Regarding the problem of tablature arrangement from symbolic data (not audio), Tuohy and Potter combined a neural network model with a local heuristic-climber to take pieces composed for other instruments and arrange them for guitar [23]. In [18], Mistler used deep neural networks to arrange guitar tablature from sheet music by predicting a fretting cost function and predicting string-fret activations directly.

The problem of arranging solo guitar covers of pop songs was addressed by Ariga et al., who proposed a system which processes pop music audio and generates tablature arrangements that can scale in difficulty [1]. Finally, in the area of automatic music generation, McVicar et al. built an algorithmic composition system to compose tablatures for rhythm and lead guitar parts from an input chord progression [17].

2.3 CNNs for Automatic Music Transcription

While, to our knowledge, the task of guitar tablature estimation has not been approached using convolutional neural networks, CNNs have shown promise for the similar task of automatic piano transcription. In [21], Stigita et al. proposed a hybrid neural network model for piano transcription that combines a CNN for framewise acoustic modelling and a recurrent neural network (RNN)-based music language model. Kelz et al. provided a comparison of various network-based approaches to framewise transcription of piano audio, and the authors argued that the high accuracy and low parameter amount offered by CNNs, compared to other classes of deep neural networks, gives them a "distinct advantage" for piano transcription [14].

The use of CNNs has also been explored for various other tasks within music information retrieval such as musical tempo estimation [20], key classification [15], singing voice detection [19], and instrument classification [9, 10].

3. METHODOLOGY

3.1 Dataset

Since we are interested in learning tablatures for the exact fingerings used by the guitarist, it is important that the dataset contains audio of an actual guitar performance. Previous approaches to guitar transcription utilize audio

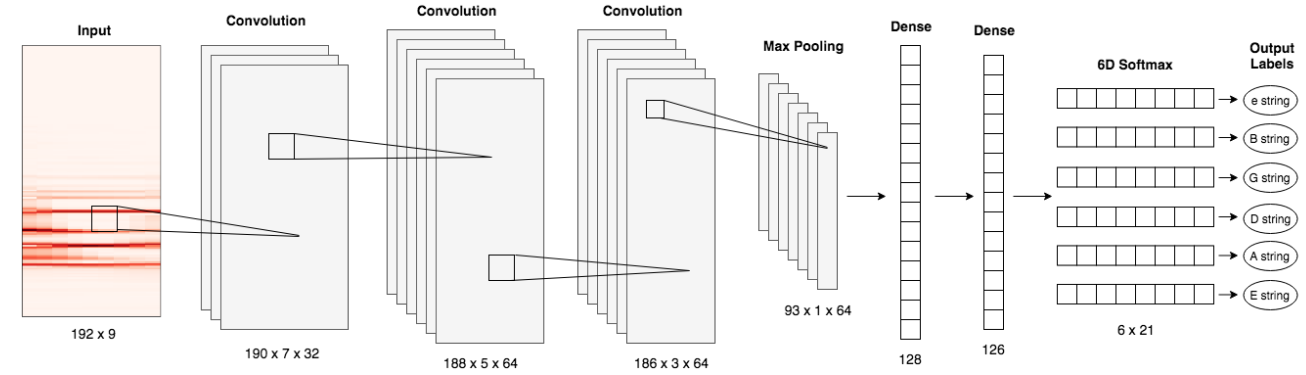


Figure 2. The proposed TabCNN network architecture. The input is a constant-Q spectrogram image of solo acoustic guitar audio. There are a series of 2D convolutional layers followed by a max pooling layer, which learn to extract spatial features relevant for guitar tablature estimation. The dense layers and final softmax function aim to use the learned representation to predict fret-number labels for each of the 6 guitar strings.

data that was automatically generated using MIDI to playback audio samples [5, 6, 25, 26]. As a result, the audio does not represent a performance of specific guitar fingerings. This is problematic for our purposes since there is no ground truth tablature.

We employ the GuitarSet dataset [24], which consists of audio recordings of solo acoustic guitar performances. This dataset was created using a guitar equipped with a hexaphonic pickup. The signals from each individual guitar string were processed separately to produce a frame-level pitch annotation for each of the 6 guitar strings. This dataset enables the ground truth fretting used by the guitarist to be easily accessible. Since the guitar used remained in standard tuning, we can use these pitch estimations to determine the corresponding frets being fingered over the course of the performance. While the hexaphonic signals were used by the GuitarSet authors to create the ground truth annotations, our system uses only the monophonic microphone signal of each song to estimate the tablature.

The GuitarSet contains audio recordings of 360 solo guitar performances, each approximately 30 seconds in length. These performances span every key and cover a variety of styles including bossa nova, rock, and funk. The guitarists were instructed to play two different versions of each song: soloing, which contains mostly single notes, and comping, which means playing chords. Six different guitarists contributed to the dataset, each performing to a provided chord progression, but interpreting it in their own way.

3.2 Audio Preprocessing

During the preprocessing stage, we first downsample the audio from 44100 Hz to 22050 Hz, making the assumption that there is not too much relevant information above 11025 Hz, in order to reduce the dimensionality of the input signals. We normalize each audio clip by its maximum value, to account for any major amplitude differences between clips.

As CNNs are useful in learning spatial features from images, we are interested in transforming the raw audio data into an image representation. The Short-Time Fourier Transform (STFT), commonly used to represent a signal changing over time and frequency, is not a desirable choice for CNNs because of its high dimensionality. Additionally, since the task at hand involves recognizing musical pitches, it would be advantageous to use a representation with a frequency axis that is linearly spaced with respect to pitch. This allows pitch-invariant features to be learned by the network. For these reasons we employ the constant-Q transform (CQT), which greatly reduces the dimension of the frequency axis by linearly spacing the frequency bins with respect to musical pitch.

Motivated by previous work [12], we use a CQT with 192 bins, spanning 8 octaves. This equates to 24 bins per octave, or 2 bins per semitone. We use a hopsize of 512 samples, which corresponds to a frame rate of about 43 frames per second, a rate sufficient for framewise analysis [10]. Using a sliding context window, we generate input images centered around each frame. In our experiments, we observed the best results using a context window of 9 frames. We pad either side of the CQT with zeros so that the beginning and ending frames of each clip can have a full 9-frame-long context window. Thus, input samples are of size 192×9 at each frame.

3.3 Label Preprocessing

To obtain annotations for each individual frame of audio, we sample the stringwise pitch annotations at the same frame rate of approximately 43 frames per second. For each sampled MIDI pitch value, we round it to the nearest integer, which corresponds to finding the nearest musical pitch. Depending on which string the pitch was played on, we subtract the corresponding string's open pitch value. The resulting integers correspond to the frets that were activated to produce the sounding pitches. A value of zero indicates that the string was plucked open (while no frets were being pressed). Since there are 19 frets used on the acoustic guitar, and a string may at times be closed (not produce any sound), this results in 21 different fret classes

that each string may be in during any given frame: open, closed, or any one of the 19 frets. We convert each string's label to a one-hot representation, resulting in label size of 6×21 at each frame.

3.4 Network Architecture

The structure of our neural network model is generally inspired by popular models in computer vision, such as AlexNet [16] and VGGNet [22], which both contain series of convolutions followed by a sub-sampling operation. These architectures terminate in dense layer connections. When using this type of CNN architecture, the expectation is that the early convolutional layers act as a hierarchy of feature extractors, learning spatial filter coefficients that result in feature maps useful for the given classification task. We interpret the final dense layers and output connection to act as a classifier that processes the features to output a final decision or prediction.

The proposed network structure for TabCNN is shown in Figure 2. First, there is a series of three convolutional layers, each with a filter size of 3×3. The first convolutional layer has 32 filters, and the latter two each have 64. These parameters were determined empirically, based on experiments with the validation data. Each convolution is immediately followed by a Rectified Linear Unit (ReLU) activation. Activation functions, in general, allow for complicated, nonlinear mappings to be learned, and the ReLU activation has been shown to train faster [16] and produce better results [8] than alternatives.

The resulting feature maps are subsampled by a 2×2 max pooling layer, which introduces a slight translation invariance in both time and frequency. The structure is then flattened and followed by a dense layer of dimension 128, this size determined empirically in our experiments. After this dense layer, a ReLU activation is applied. This is connected to a second dense layer of dimension 126, which is reshaped to 6×21. The output shape comes from the 6 guitar strings and the 21 different fret classes a string can be assigned. Finally, a softmax activation is applied to each of the 6 rows. As a result, similar to the network described in [12], our model learns to output an estimation of six probability mass functions, which represent the probability of each fret class for each string.

3.5 Training Procedure

For training the model, we design a loss function by viewing the problem as 6 simultaneous multiclass classification problems. For multiclass classification, a common practice is to optimize the categorical cross-entropy between the predictions and the targets [10]. For our loss function, we compute the categorical cross-entropy for each string and sum these values. The loss function is computed as in Eqn (1). We use z_{ij} to denote an activation at frame i on string j that belongs to fret class $C_{z_{ij}}$. The term $p[z_{ij} \in C_{z_{ij}}]$ is the probability output by the network of z_{ij} belonging to class $C_{z_{ij}}$. N is the total number of frames in the mini batch.

$$Loss = -\frac{1}{N} \sum_{j=1}^{6} \sum_{i=1}^{N} \log p[z_{ij} \in C_{z_{ij}}] \tag{1}$$

We train using the ADADELTA optimization algorithm [27], which adapts learning rates for parameters based on a window of previous gradient values. We use an initial learning rate of 1.0 and a mini batch size of 128 training samples. We train for 8 epochs, as we noticed overfitting when training for longer. We also employ dropout regularization to combat overfitting, with a dropout rate of 0.25 applied immediately after the max pooling layer, and a second dropout rate of 0.5 applied after the first dense layer.

4. EVALUATION

To evaluate the proposed system, we first review multipitch estimation metrics from the literature [6] and assess the quality of the system as a polyphonic pitch detector. Then, since there are no standards for evaluating the performance of a tablature estimation system, we introduce a set of metrics to measure performance in estimating tablature for the actual fingering used during the performance. In our evaluations we use 6-fold cross validation, holding out one of the 6 guitarists to test on, while training on the remaining 5. We average our numerical results over the 6 folds of data, testing with a total number of 472,560 frames.

4.1 Multipitch Estimation Metrics

In the following equations, we use Y to denote a binary matrix of size $N \times 44$, where N is the total number of testing frames, and the matrix represents the presence or absence of each pitch for each frame of audio. (The guitar can produce a total of 44 distinct pitches.) Y_{gt} contains the ground truth pitch detections for the testing set, while Y_{pred} contains the predicted pitch detections during testing. We use e to denote a vector of all ones, and $\odot$ to denote element-wise multiplication.

4.1.1 Multipitch Precision

We compute multipitch precision using Eqn (2), which calculates the number of correctly identified pitches divided by the total number of predicted pitches. This metric measures how frequently the pitches that are detected are in fact correct.

$$p_{\text{pitch}} = \frac{e^T (Y_{\text{gt}} \odot Y_{\text{pred}}) e}{e^T Y_{\text{pred}} e} \tag{2}$$

4.1.2 Multipitch Recall

We also compute multipitch recall using Eqn (3), which calculates the number of correctly identified pitches divided by the the total number of ground truth pitches. This metric measures how frequently pitches existent in the signal are are detected by the system.

$$r_{\text{pitch}} = \frac{e^T (Y_{\text{gt}} \odot Y_{\text{pred}}) e}{e^T Y_{\text{gt}} e} \tag{3}$$

System	p_{pitch}		r_{pitch}		f_{pitch}	
TabCNN	**0.900**	±	**0.764**	±	**0.826**	±
	0.016		0.043		0.025	
Deep	0.778	±	0.562	±	0.646	±
Salience	0.092		0.099		0.078	

Table 1. Multipitch estimation metrics for our system, TabCNN, compared against a baseline, the Deep Salience f0-estimation algorithm introduced in [4], from experiments carried out in [24]. For all metrics, we report the mean and standard deviation over the entire dataset.

4.1.3 Multipitch F-measure

Finally, we compute multipitch F-measure using Eqn (4), which is the harmonic mean of multipitch precision and recall. This metric summarizes the system's overall performance for multipitch estimation.

$$f_{\text{pitch}} = \frac{2p_{\text{pitch}}r_{\text{pitch}}}{p_{\text{pitch}} + r_{\text{pitch}}} \tag{4}$$

4.2 Multipitch Estimation Results

The evaluation results of TabCNN for multipitch estimation, alongside a baseline algorithm, are shown in Table 1. For a benchmark, we look to the GuitarSet paper [24], in which the authors employ the Deep Salience multiple-f0 estimation algorithm in [4] to transcribe framewise pitch estimations for the GuitarSet data. [2] Deep Salience, although not designed specifically for guitar transcription, achieves a multipitch F-measure of 0.646. The proposed system TabCNN outperforms these baseline results, achieving a multipitch F-measure of 0.826.

Looking at a recent multipitch system designed specifically to operate on guitar signals, in [6] Burlet and Hindle report a multipitch F-measure of 0.71 before frame smoothing with a Hidden Markov Model (HMM), and an F-measure of 0.77 afterward. Our system's multipitch F-measure of 0.826 without any temporal smoothing is promising. However, these results cannot be compared too closely since the authors were evaluating on a different dataset.

Our results reveal that the proposed system behaves conservatively for multipitch estimation. The multipitch precision being significantly larger than the multipitch recall indicates that the system will more often make an error by missing a detection, rather than reporting a pitch not actually present in the signal. This is a common behavior for multipitch estimation systems, as a pitch can be easily missed if it is able to blend in to the overtones of another pitch present at the same time. We noticed that this often occurs with pitches an octave apart; the higher pitch will often be missed, as it may appear to just be overtones of the lower pitch.

[2] We acknowledge and thank the authors of [24] who have granted us access to the full numerical results from the experiments conducted in the work.

4.3 Tablature Estimation Metrics

We use Z to denote a binary matrix of size $N \times 120$, N being the total number of testing frames. The dimension of 120 arises from all possible sounding string-fret combinations on guitar (6×20). This matrix represents the presence or absence of each string-fret combination for each frame of audio. Z_{gt} contains the ground truth tablature detections for the testing set, while Z_{pred} contains the predicted tablature detections during testing. Again, we use e to denote a vector of all ones and $\odot$ to denote element-wise multiplication.

4.3.1 Tablature Precision

We define tablature precision, which calculates the number of correctly identified string-fret combinations divided by the total number of predicted string-fret combinations. This metric measures how frequently the tablature detected is in fact correct.

$$p_{\text{tab}} = \frac{e^T(Z_{\text{gt}} \odot Z_{\text{pred}})e}{e^T Z_{\text{pred}} e} \tag{5}$$

4.3.2 Tablature Recall

We also introduce tablature recall, which calculates the number of correctly identified string-fret combinations divided by the the total number of ground truth string-fret combinations. This metric measures how frequently tablature existent in the signal is detected by the system.

$$r_{\text{tab}} = \frac{e^T(Z_{\text{gt}} \odot Z_{\text{pred}})e}{e^T Z_{\text{gt}} e} \tag{6}$$

4.3.3 Tablature F-measure

We define multipitch F-measure, which is the harmonic mean of tablature precision and recall. This metric summarizes the system's overall performance for tablature estimation.

$$f_{\text{tab}} = \frac{2p_{\text{tab}}r_{\text{tab}}}{p_{\text{tab}} + r_{\text{tab}}} \tag{7}$$

4.3.4 Tablature Disambiguation Rate

Finally, we introduce a tablature disambiguation rate (TDR) which is computed by dividing the total number of correctly identified string-fret combinations by the total number of of correctly identified pitches. This metric measures how frequently pitches that are correctly identified are assigned the correct tablature.

$$TDR = \frac{e^T(Z_{\text{gt}} \odot Z_{\text{pred}})e}{e^T(Y_{\text{gt}} \odot Y_{\text{pred}})e} \tag{8}$$

4.4 Tablature Estimation Results

The tablature estimation evaluation results for TabCNN are shown in Table 2. While, to our knowledge, there are no prior approaches to directly compare these metrics to, the results are respectable, as each tablature metric is not too far below its multipitch counterpart. As in multipitch estimation, the proposed system achieves a higher tablature

System	p_{tab}	r_{tab}	f_{tab}	TDR
TabCNN	**0.809** $\pm$ 0.029	**0.696** $\pm$ 0.061	**0.748** $\pm$ 0.047	**0.899** $\pm$ 0.033

Table 2. Tablature estimation results for the proposed system, TabCNN, using the metrics we introduce to measure performance in fingering prediction. For all metrics, we report the mean and standard deviation over the entire dataset.

precision than tablature recall. This implies that errors made by the system are more often due to missed string-fret combinations than predicting fingerings not present in the signal. The TDR of 0.899 indicates that over 89% of correctly identified pitches are assigned the correct fingering. This value is promising given that the majority of pitches playable on guitar can be played in multiple locations.

5. DISCUSSION

Viewing the predicted tablature alongside the ground truth [3] , the system appears to output correct string-fret activations a good amount of the time, with the occasional error. To better understand these results, Figure 3 contains examples of 3 common types of errors made by the proposed system.

False alarms occur when a string-fret combination that is predicted is not actually present in the input signal. This type of error negatively impacts the tablature precision metric. In Figure 3 (a), a false alarm occurs when an F3 pitch is mistakenly detected on the 3rd fret of the D-string. This specific error likely happened because the overtones of the F2 pitch, which is present, could be mistaken for the presence of the F3 pitch an octave up. Also, the predicted fingering is a chord shape commonly used by guitarists called a "power chord."

Missed detections occur when the system fails to detect a string-fret combination that is present in the input signal. These errors hurt the tablature recall score. In Figure 3 (b), the presence of the A♭4 pitch on the 4th fret of the e-string is missed by the system. This type of error could be due to the note having been played quietly or having mostly faded out by the current frame, but still technically being active according to the ground truth. Also, this specific error can be attributed to octave confusion, since the A♭4 could easily be mistaken for overtones of A♭3 note, which was activated on the 6th fret of the D-string.

Finally, miss-frettings occur when a pitch is correctly detected, but it is assigned the incorrect string-fret combination. This is essentially a simultaneous false alarm and missed detection, so both the tablature precision and tablature recall are negatively affected. However, the TDR metric most directly represents how often this third type of error is avoided. In Figure 3 (c), the A♭4 pitch is detected as an activation on the 4th fret of the e-string. However, the pitch was actually played on the 9th fret of the B-string. In this is specific case, there were no other pitches simultaneously present to cause confusion. Instead, this error is likely due to the system's failure to predict the correct guitar string based on timbre.

[3] As supplementary materials, video demonstrations showing predicted and ground truth tablature synced with input audio are available at: https://github.com/andywiggins/tab-cnn

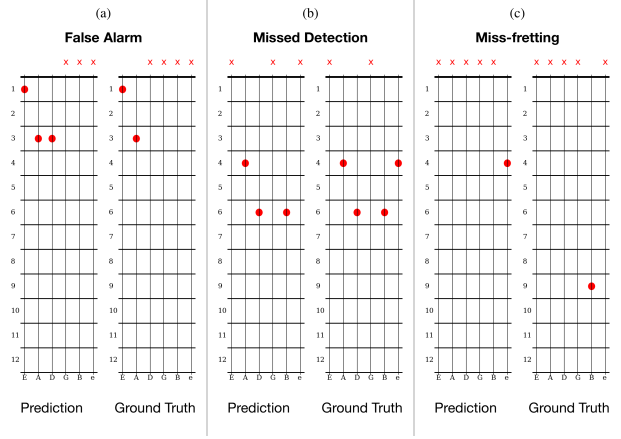


Figure 3. Fretboard diagrams showing examples of 3 common error types made by TabCNN. In each diagram, the vertical lines are guitar strings, and the horizontal lines are the frets. The circles indicate positions of the guitarist's fingers on the fretboard. An "X" over a string means that string is not sounding.

Overall, we observed that errors often occur at the beginning or end of a note. A strategy for improving estimation performance could be the incorporation of a temporal smoothing algorithm.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed TabCNN, a convolutional neural network (CNN) approach to tablature estimation from audio of a solo acoustic guitar performance. Our system performs competitively in guitar multipitch estimation, while providing the advantage of additionally predicting the true fingering used by the guitarist. The guitar community, which frequently shares music in tablature form, can benefit from a system that automates the arduous task of transcribing tablature, while retaining nuance from the original performance, in the form of the exact fretboard positions being used.

Looking ahead to future work, the approach in this paper could be integrated into an end-to-end tablature transcription system. A temporal smoothing method could be added to aggregate these framewise tablature estimations into a note-level transcription. Additionally, we hope that the tablature estimation metrics introduced in this paper can serve to evaluate future guitar transcription approaches that aim to predict the guitarist fingering.

7. REFERENCES

[1] Shunya Ariga, Satoru Fukayama, and Masataka Goto. Song2guitar: A difficulty-aware arrangement system

for generating guitar solo covers from polyphonic audio of popular music. In *Proceedings of the 18th International Conference on Music Information Retrieval (ISMIR), Suzhou, China*, 2017.

[2] Ana M Barbancho, Anssi Klapuri, Lorenzo J Tardón, and Isabel Barbancho. Automatic transcription of guitar chords and fingering from audio. *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, 20(3):915–921, 2012.

[3] Isabel Barbancho, Lorenzo J Tardon, Simone Sammartino, and Ana M Barbancho. Inharmonicity-based method for the automatic generation of guitar tablature. *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, 20(6):1857–1868, 2012.

[4] Rachel M Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan P Bello. Deep salience representations for f0 estimation in polyphonic music. In *Proceedings of the 18th International Conference on Music Information Retrieval (ISMIR), Suzhou, China*, 2017.

[5] Gregory Burlet and Ichiro Fujinaga. Robotaba guitar tablature transcription framework. In *Proceedings of the 14th International Conference on Music Information Retrieval (ISMIR), Curitiba, Brazil*, 2013.

[6] Gregory Burlet and Abram Hindle. Isolated guitar transcription using a deep belief network. *PeerJ Computer Science*, 3:e109, 2017.

[7] Benoit Fuentes, Roland Badeau, and Gaël Richard. Blind harmonic adaptive decomposition applied to supervised source separation. In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 2654–2658. IEEE, 2012.

[8] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[9] Juan S Gómez, Jakob Abeßer, and Estefanía Cano. Jazz solo instrument classification with convolutional neural networks, source separation, and transfer learning. In *Proceedings of the 19th International Conference on Music Information Retrieval (ISMIR), Paris, France*, 2018.

[10] Yoonchang Han, Jaehun Kim, Kyogu Lee, Yoonchang Han, Jaehun Kim, and Kyogu Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 25(1):208–221, 2017.

[11] Alex Hrybyk and Youngmoo Kim. Combined audio and video analysis for guitar chord identification. *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR), Utrecht, Netherlands*, 2010.

[12] Eric J Humphrey and Juan P Bello. From music audio to chord tablature: Teaching deep convolutional networks to play guitar. In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6974–6978. IEEE, 2014.

[13] Christian Kehling, Jakob Abeßer, Christian Dittmar, and Gerald Schuller. Automatic tablature transcription of electric guitar recordings by estimation of score- and instrument-related parameters. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 219–226, 2014.

[14] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. *arXiv preprint arXiv:1612.05153*, 2016.

[15] Filip Korzeniowski and Gerhard Widmer. Genre-agnostic key classification with convolutional neural networks. In *Proceedings of the 19th International Conference on Music Information Retrieval (ISMIR), Paris, France*, 2018.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] Matt McVicar, Satoru Fukayama, and Masataka Goto. Autoguitartab: computer-aided composition of rhythm and lead guitar parts in the tablature space. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(7):1105–1117, 2015.

[18] Elias Mistler. Generating guitar tablatures with neural networks. *Master of Science Dissertation, The University of Edinburgh*, 2017.

[19] Jan Schlüter and Bernhard Lehner. Zero-mean convolutions for level-invariant singing voice detection. In *Proceedings of the 19th International Conference on Music Information Retrieval (ISMIR), Paris, France*, 2018.

[20] Hendrik Schreiber and Meinard Müller. A single-step approach to musical tempo estimation using a convolutional neural network. In *Proceedings of the 19th International Conference on Music Information Retrieval (ISMIR), Paris, France*, 2018.

[21] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, 24(5):927–939, 2016.

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[23] Daniel R Tuohy, Walter D Potter, and Artificial Intelligence Center. An evolved neural network/hc hybrid for tablature creation in ga-based guitar arranging. In *International Computer Music Conference Proceedings*, 2006.

[24] Qingyang Xi, Rachel M Bittner, Johan Pauwels, Xuzhou Ye, and Juan P Bello. Guitarset: A dataset for guitar transcription. In *Proceedings of the 19th International Conference on Music Information Retrieval (ISMIR), Paris, France*, 2018.

[25] Kazuki Yazawa, Katsutoshi Itoyama, and Hiroshi G Okuno. Automatic transcription of guitar tablature from audio signals in accordance with player's proficiency. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3122–3126. IEEE, 2014.

[26] Kazuki Yazawa, Daichi Sakaue, Kohei Nagira, Katsutoshi Itoyama, and Hiroshi G Okuno. Audio-based guitar tablature transcription using multipitch analysis and playability constraints. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 196–200. IEEE, 2013.

[27] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[28] Ruohua Zhou and Joshua D Reiss. A real-time polyphonic music transcription system. *Proceedings of the 4th Music Information Retrieval Evaluation eXchange (MIREX)*, 2008.

Session C

LEARNING A JOINT EMBEDDING SPACE OF MONOPHONIC AND MIXED MUSIC SIGNALS FOR SINGING VOICE

Kyungyun Lee **Juhan Nam**
Graduate School of Culture Technology, KAIST
{kyungyun.lee, juhannam}@kaist.ac.kr

ABSTRACT

Previous approaches in singer identification have used one of monophonic vocal tracks or mixed tracks containing multiple instruments, leaving a semantic gap between these two domains of audio. In this paper, we present a system to learn a joint embedding space of monophonic and mixed tracks for singing voice. We use a metric learning method, which ensures that tracks from both domains of the same singer are mapped closer to each other than those of different singers. We train the system on a large synthetic dataset generated by music mashup to reflect real-world music recordings. Our approach opens up new possibilities for cross-domain tasks, e.g., given a monophonic track of a singer as a query, retrieving mixed tracks sung by the same singer from the database. Also, it requires no additional vocal enhancement steps such as source separation. We show the effectiveness of our system for singer identification and query-by-singer in both the in-domain and cross-domain tasks.

1. INTRODUCTION

Singing voice is often at the center of attention in popular music. We can easily observe large public interest in singing voice and singers through the popularity of karaoke industry and singing-oriented television shows. A recent study also showed that some of the most salient components of music are singers (vocals, voice) and lyrics [5]. Therefore, extracting information relevant to singing voice, i.e., to singers, from music signals, is an important area of research in music information retrieval (MIR) [9, 11]. The relevant tasks include singing voice detection [16], singing melody extraction [14, 28], singer identification [12, 21], and similarity-based music retrieval [8, 22].

Modern singer information processing systems have been designed to work with only one of monophonic or mixed music signals [15, 21, 34]. Then, given both types of signals for analysis, we question whether the system can extract information relevant to singing voice that is transferable between between monophonic and mixed tracks. In our experiment, we observe that systems trained with only one type of signals do not perform well, when tested with another type. To address this limitation, we introduce a new problem of *cross-domain* singer identification (singer-ID) and similarity-based retrieval, in which we regard monophonic and mixed music signals as two different audio domains. Cross-domain problems have been explored in computer vision and recommender systems, for example, image retrieval from user sketches to real images [29] and user preference modeling from movies to books [6]. In MIR, information transfer between monophonic and mixed tracks can open up new possibilities for singer-based retrieval systems. Some examples are: 1) given a user's vocal recording in a karaoke application, finding popular singers who sound similar to the user, and 2) given a studio vocal track of a singer, retrieving all tracks (monophonic and mixed) relevant to the singer from a large music database.

To learn a joint feature representation of data from both monophonic and mixed tracks, we adopt a metric learning method, which forces tracks from the same singer to be mapped closer to each other than those from others (Section 3.2). To acquire sufficient training data, we create a synthetic dataset by performing a simple music mashup on two public datasets: vocal recordings from DAMP [30] and background tracks from musdb18 [25] (Section 3.1.1). We present experiments to demonstrate that our system is able to extract singer-relevant information from both monophonic and mixed music signals, and share the information between the two domains (Section 4). Source code, trained models, example audios and detailed information about the dataset are available [1] .

2. RELATED WORK

Cross-domain systems have not yet been examined regarding singing voice analysis. Nonetheless, a common challenge in singer information processing systems is to extract singing voice characteristics from music signals in the presence of background accompaniment music. The most direct way to obtain vocal information is to use monophonic vocal tracks. Recently, Wang et al. [34] trained a siamese neural network on monophonic recordings from a subset of the DAMP dataset. Their model scored higher on singer classification but lower on song classification compared to a baseline model. This implies that the model was able to learn singing voice characteristics, rather than the

[1] http://github.com/kyungyunlee/mono2mixed-singer

content of a music piece, such as its melody or lyrics. However, since music of our interest is often mixed tracks, this approach has limitations.

Several works have handled mixed audio signals by enhancing vocal signals through source separation or melody enhancement [8, 15, 21]. Given recent advances in source separation [32], this approach may bring improved results for most singing voice analysis systems. Another common choice is using audio features that represent human voice or singing voice, such as mel-frequency cepstral coefficients (MFCCs) [2, 15]. With the success of deep neural networks, it is even possible to learn appropriate features from more general audio representations, i.e., short-time Fourier transform (STFT) or even raw audio. We take this last approach and train our model to be a feature extractor for a given input audio represented by a mel-spectrogram.

Depending on the target task, background music can be helpful. An example is singer recognition in popular music [19]. This is because singing style is often dependent on the genre or mood of the music, and singers tend to perform in similar genres throughout their careers. However, our work focuses on learning the actual characteristics of singing voice, independent from background music.

3. METHODS

In this section, we describe the data generation pipeline, model configuration and training strategy for learning a joint representation of monophonic and mixed tracks for singing voice.

3.1 Data generation

For training cross-domain singer-ID and retrieval systems, a sufficiently large number of monophonic and mixed track pairs per singer is needed. Existing singing voice datasets, such as MIR-1K [10], iKala [3] and Kara1K [1], provide the monophonic and mixed track pairs, but they have a small number of singers or only a few tracks per singer. An alternative option may be to perform singing voice detection (SVD) and vocal source separation on a large dataset, but the audio quality can be degraded.

In this work, we choose to utilize the DAMP dataset, which contains vocal-only recordings from mobile phones of around 3,500 users from the Smule karaoke app (there are 10 full-length songs per user). This serves as the main ingredient to generate our synthetic singer dataset. As a preprocessing step, we perform a simple energy-based SVD to remove silent segments. Then, 1000 singers are chosen for training stage and additional 300 singers are put aside for testing. The original DAMP dataset processed with SVD, *DAMP-Vocal*, is used as the monophonic dataset; the synthesized mixed track dataset, *DAMP-Mash* (detailed in section 3.1.1), is used as the mixed track dataset in this work.

3.1.1 Mashup: DAMP and musdb18

A music mashup is a way of creating music by carefully mixing two or more tracks from several different pre-recorded songs. Inspired by such work, we automatically generate a synthetic singer dataset, called DAMP-Mash, by combining vocal recordings from the DAMP dataset with background tracks from the musdb18 dataset. Instead of random mixing, we build a pipeline (Figure 1) to identify the "mashability" [4] between tracks. Our mashability criteria requires 3-second long vocal and background tracks to have the same tempo and key. Once the two segments pass the mashability test, they are mixed at their nearest beat location. Before mixing, we adjust the loudness by balancing the root-mean-square energy between both segments.

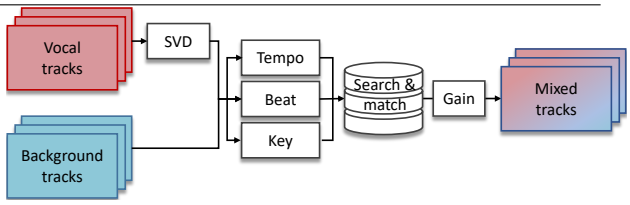


Figure 1: Mashup pipeline to generate synthetic dataset, *DAMP-Mash*.

Tempo detection and beat tracking are performed at track-level using librosa 0.6.2 [20]. On the other hand, key is determined locally at 3-second long segments by using the Krumhansl-Schmuckler key finding algorithm [13] on chromagrams. As a result, vocal segments within the same song end up being mixed with multiple different background tracks. Thus, we view our synthetic dataset as being genre-independent. This mashup pipeline can be also regarded as a data augmentation technique.

3.2 Model

3.2.1 Skeleton model

A 5-layer 1-D convolutional neural network (CNN) is the skeleton of larger networks used in this paper. First four convolutional layers have 128 filters of size 3, each followed by a maxpooling layer of size 3. The last convolutional layer consists of 256 filters of size 1 to output a final embedding vector of 256 dimensions. All convolution operations are done on the temporal dimension only. The final embedding vector will be used to represent input audio and to perform tasks described in Section 4. Batch normalization and Leaky ReLU [17] are applied to all convolutional layers, and dropout of 50% is applied after the last convolutional layer to prevent overfitting.

The input is a 3-second long audio of at least 70% singing voice frames. The sample rate of audio is 22050 Hz and we compute an STFT using a 1024-sample long Hanning window with 50% overlap. We then convert it to a mel-spectrogram with 128 bins and apply logarithmic compression on the magnitude. As a result, the input shape is 129 frames by 128 bins.

3.2.2 Embedding model

The outcome of metric learning is a mapping function from inputs to output vectors in an embedding space, where inputs of the same class are closer to each other than those

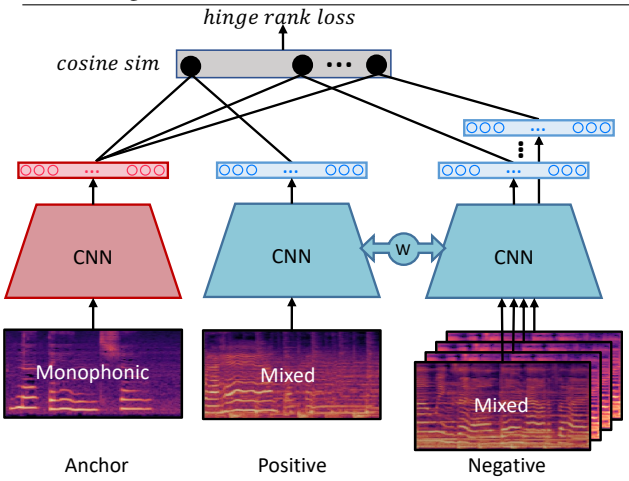


Figure 2: Configuration of `CROSS` model. The anchor network (left) is for modeling monophonic tracks; the rest is for modeling mixed tracks.

from different classes. Specifically, we build our model upon a triplet network, which consists of three potentially weight-sharing networks that takes three inputs: anchor, positive (same class as the anchor) and negative (different class as the anchor) items. This architecture can be extended to take multiple negative items [31] to overcome the limitation of learning from only one negative example. For model configuration, we closely follow the work of [24], using 4 negative items. We also use a type of margin-based ranking loss, called hinge rank loss, with cosine similarity as our metric [7].

Thus, the loss function for a given set of anchor (p_i), positive (p_+) and negative (p_-) feature vectors is:

$$loss(p_i, p) = \sum_{p_-} max[0, \alpha - S(p_i, p_+) + S(p_i, p_-)] \quad (1)$$

where S is a similarity score:

$$S(p_i, p_j) = cos(p_i, p_j) = \frac{p_i \cdot p_j}{||p_i|| \cdot ||p_j||} \quad (2)$$

and α indicates the margin, which is fixed to 0.1 after performing a grid search on values between 0.01 and 1.0. Negative tracks are selected through negative sampling among tracks that do not belong to the singer of the anchor item. We tested a more difficult negative sampling strategy of selecting the four highest scoring items among twenty randomly chosen negative samples, but the model showed minor improvement with an increase in computation time. Investigation on negative sampling is left as our future work.

We choose metric learning for three main reasons. First, by giving a higher similarity score to any pair of tracks performed by the same singer, the model can learn to identify the singer from a track regardless of it being monophonic or mixed. Thus, it is especially suitable for training cross-domain systems. Second, using singer identity as the only ground truth to measure similarity between two tracks will force the model to focus only on singing voice. Since DAMP-Mash is genre-invariant, the only common component in two tracks is going to be related to singing voice. Thus, we may expect the model to perform a feature-level source separation on mixed tracks. Lastly, the model can be trained on a larger number of singers without increasing the number of parameters. On the other hand, a classification model that uses a softmax layer will need to increase the output layer size to match the number of training singers [24].

We explore our ideas with three models, which differ in the type of data used for training:

- `MONO`: all inputs are monophonic tracks
- `MIXED`: all inputs are mixed tracks
- `CROSS`: anchors are monophonic, while positive and negative items are mixed tracks (Figure 2)

Our main idea in this work is reflected in the `CROSS` model, for which the hinge rank loss ensures that the cosine similarity between monophonic and mixed tracks from the same singer is scored higher than that from different singers. `MONO` and `MIXED` models are reference models for comparison.

While networks within `MONO` and `MIXED` model share weights, in `CROSS` model, the anchor network and the rest do not share weights. Thus, it yields two separate feed-forward networks, each designed specifically for its corresponding domain (Figure 2). As a result, depending on the domain of an input audio, one of the two networks is used as the feature extractor. Each network is configured with the skeleton model described in Section 3.2.1.

3.2.3 Pre-training via classification

Metric learning is known for its difficulty in optimization [35, 36]. To alleviate this problem, we train a classification model and use it to initiate the learning of the embedding models. The classification model has one linear layer added to the skeleton model (Section 3.2.1) and predicts the correct singer with a softmax probability. Instead of fully training it, we remove the last output layer after 30 epochs and use it to continue the training in a metric learning style. We do not freeze any layers.

4. EXPERIMENTS & EVALUATION

4.1 Test scenarios

Two main tasks for evaluation are singer identification and query-by-singer. In both tasks, a music signal to be analyzed (source) is queried to a collection of data (target) to retrieve desired information. Depending on the domain of source and target data, we design three test scenarios:

- *Mono2Mono*: both source and target data are *monophonic* (in-domain)
- *Mix2Mix*: both source and target data are *mixed* (in-domain)
- *Mono2Mix*: source data is *monophonic*, but the target data is *mixed* (cross-domain)

Each task is evaluated on all three test scenarios.

4.2 Task 1: Singer identification

Dataset : We select 300 singers unseen from the training stage for evaluation. For each singer, we use 6 tracks for building singer models and set aside 4 tracks as query tracks, resulting in 1200 queries. Depending on the domain of source and target data, DAMP-Vocal (monophonic) and DAMP-Mash (mixed) dataset are used accordingly.

Description : As in [23, 27], singer identification is to determine the correct singer of the query track among the 300 candidate singer models. All queries and singer models are represented as 256 dimensional feature vectors; a track vector is an average of 20 feature vectors computed from 3-second long segment of the same track and a singer model is an average of 6 track vectors from the same singer. We made predictions by computing cosine similarity (2) between the query track and all the singer models. Then, the singer with the highest score is chosen.

For our baseline, we train a Gaussian Mixture Model-Universal Background Model (GMM-UBM), which is commonly used in speaker recognition systems [26]. Each singer model is adapted through maximum a posteriori (MAP) estimation from a single singer-independent background model. All models are composed of 256 components with MFCCs of dimension 13 as input. We train 2 GMM-UBMs, one with monophonic tracks for *Mono2Mono* and the other with mixed tracks for *Mix2Mix*.

We report both top-1 and top-5 classification accuracy. They represent the proportion of correct guesses out of 1200 queries in total. Top-5 accuracy is calculated by considering a prediction as being correct if the ground truth singer appears within the top 5 highest scoring singers.

4.3 Task 2: Query-by-singer

Dataset : As in the singer recognition task (Section 4.2), same 300 singers are used for evaluation. 6 tracks from each singer are selected to build a collection of 1800 tracks to represent a search database and 4 tracks are used as test queries.

Description : Given a query track, the task is to retrieve tracks that are performed by the same singer among the track database. We compute the similarity (Equation (2)) between the query track and all the tracks in the database, and rank them based on their similarity scores. This can be applied to singer-based music recommender systems to discover singers with similar singing voice characteristics.

We report *precision* and *recall-at-k* as well as mean average precision (mAP) score, where k is set to 5 to resemble music retrieval systems. Given a query track performed by singer A, *precision-at-k* (Pr@k) refers to the proportion of tracks that are performed by A and are recommended among k items; *recall-at-k* (R@k) refers to the proportion of tracks that are performed by A and are recommended out of all the tracks performed by A (6 tracks in total) in the database. Unlike Pr@k and R@k, mAP takes into account the actual order of the recommended tracks. Thus, it

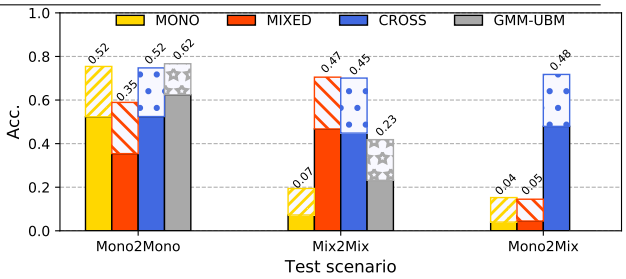


Figure 3: Singer identification results for `MONO`, `MIXED` and `CROSS` on different test scenarios. The solid section points to the top-1 accuracy (also written above each bar) and the hatched section points to the top-5 accuracy.

is useful for music recommender systems, where it is important that relevant items are not only retrieved, but also with higher confidence than false positive items.

4.4 Results

In Figure 3 and Figure 4, we observe a large performance variation for `MONO` and `MIXED` models across different test scenarios. Both models perform well in *Mono2Mono* and *Mix2Mix*, respectively. However, their performances drop significantly in other scenarios, especially for *Mono2Mix*. This is expected, because these models have not been trained to handle cross-domain scenarios.

On the other hand, `CROSS` model performs well on all three test scenarios, benefiting from two jointly trained networks that can each handle monophonic and mixed tracks. We see that it is the only model that is able to match and compare the singer identity between tracks from different domains. Also, its performances on *Mono2Mono* and *Mix2Mix* are on par with the `MONO` and `MIXED` models. This is a useful observation, since training only the `CROSS` model can still give good performance on all three test scenarios, avoiding the effort of training separate models for each scenario. Note that the baseline model, GMM-UBM, shows the best performance in *Mono2Mono*, but not so well in *Mix2Mix*. Result for *Mono2Mix* is omitted, since it is close to random prediction. When there is no background music, GMM-UBM with MFCCs are efficient in characterizing singing voice.

As mentioned in Section 3.2.3, we show the effect of using a pre-trained network on singer identification task (Figure 5). `CROSS` model (blue stars) shows the largest performance improvement compared to the other two models. We assume that comparing the singer identity between monophonic and mixed track is more difficult than comparing between tracks of the same domain. Therefore, a pre-trained model, which learned to somewhat identify singers from an input audio, serves as a hint to focus on signals relevant to singing voice. Using a pre-trained model not only improved the accuracy, but also accelerated the learning process.

Regarding background music as noise and singing voice as the signal, signal-to-noise ratio (SNR) has a large impact on the performance of singing voice analysis systems [16]. We change the SNR of the test data and show

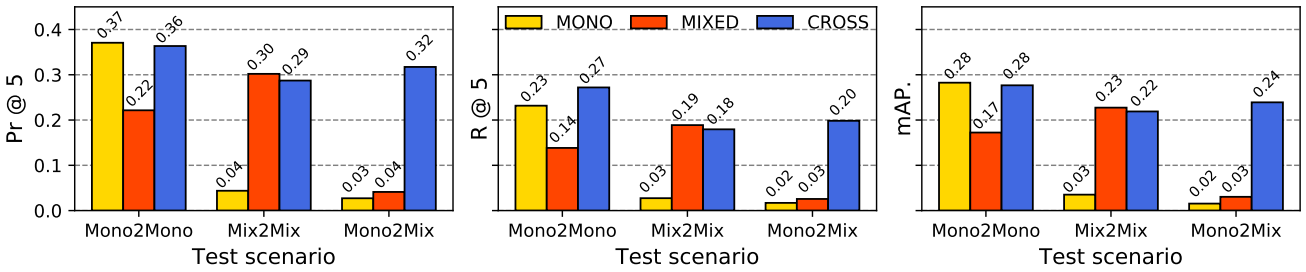


Figure 4: Query-by-singer result for MONO, MIXED and CROSS models. *precision-at-k* (left), *recall-at-k* (center) and mean average precision score(right) are shown. Each number above the bar refers to corresponding model's score.

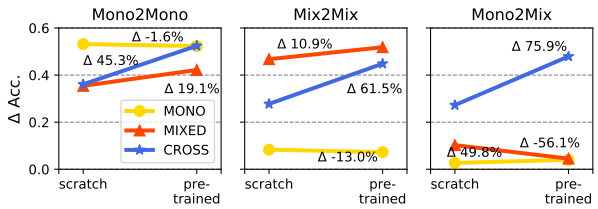


Figure 5: Top-1 accuracy improvement on singer identification when using a pre-trained network. The numbers on the line indicates the percentage of improvement.

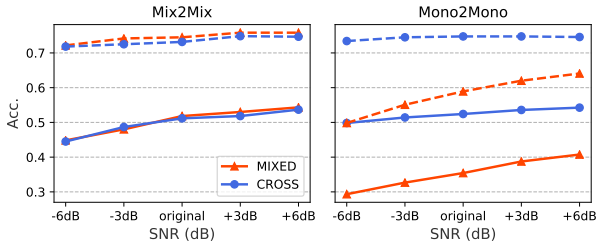


Figure 6: Result of MIXED and CROSS model for singer recognition on varying SNR. Solid line indicates top-1 and dotted line indicates top-5 accuracy.

results on singer recognition task for MIXED and CROSS models in Figure 6. Since *Mono2Mono* deals with only monophonic tracks, the change in performance exhibited in *Mono2Mono* (right) is due to the overall loudness of the track, not SNR. Therefore, as the performance change on *Mix2Mix* (left) shows a similar trend across different SNR, it implies that models trained on DAMP-Mash dataset is able to identify singing voice in more difficult conditions. This is a great benefit for singing voice analysis systems.

4.5 Evaluation on Popular Music Recordings

As our system is trained with a synthetic dataset, we evaluate it on popular music recordings to ensure that the trained system can also generalize to real-world data.

Dataset : Million Song Datatset (MSD) contains 1,000,000 tracks and 44,745 artists from popular music recordings. As done in [23], we filter the dataset to select artists with substantial vocal tracks using singing voice detection (SVD). This dataset is referred to as *MSD-Singer* [2] . For comparison, we train a model, named POP, on 1000 artists from MSD-Singer dataset. We used 17 30-second long tracks for each artist for training. 500 singers, unseen from the training stage, are used for evaluation. 15 tracks of each singer are used for building singer models and 5 tracks are used as query tracks.

	MIXED	CROSS	POP	Random
Acc.	0.291	0.282	0.393	0.002
Top-5 Acc.	0.511	0.491	0.664	0.01

Table 1: Accuracy result on singer recognition on dataset of popular music recording, MSD-Singerdataset

Description : The task is equivalent to singer identification in Section 4.2, only with a different dataset. The result from the POP model is the upper bound, as it is trained and tested on MSD-Singer dataset; meanwhile, MIXED and CROSS models are trained on DAMP-Mash and DAMP-Vocal dataset.

Result: The result shown in Table 1 compares MIXED and CROSS models with POP model and a random classifier. It shows that even though our models are trained only with the synthetic dataset, they are also able to identify singing voice in popular music. Therefore, we can confirm that DAMP-Mash dataset is able to represent the popular music to some degree and that our models are able to generalize to real-world recordings. We believe that the results will improve with a better automatic mashup pipeline. Training the CROSS model on source separated MSD-Singer dataset is also left as a future work.

5. EMBEDDING SPACE VISUALIZATION

We visualize the embedding space learned by the MIXED and CROSS models to understand how they each process monophonic and mixed tracks. From DAMP-Voice and DAMP-Mash dataset, we select 25 singers unseen from the training stage and highlight 10 with colors for better visualization. 20 tracks are plotted for each singer: 10 monophonic vocal tracks and their corresponding mixed tracks. After feature extraction, we reduce the dimension of each

[2] Details provided at http://github.com/kyungyunlee/MSD-singer

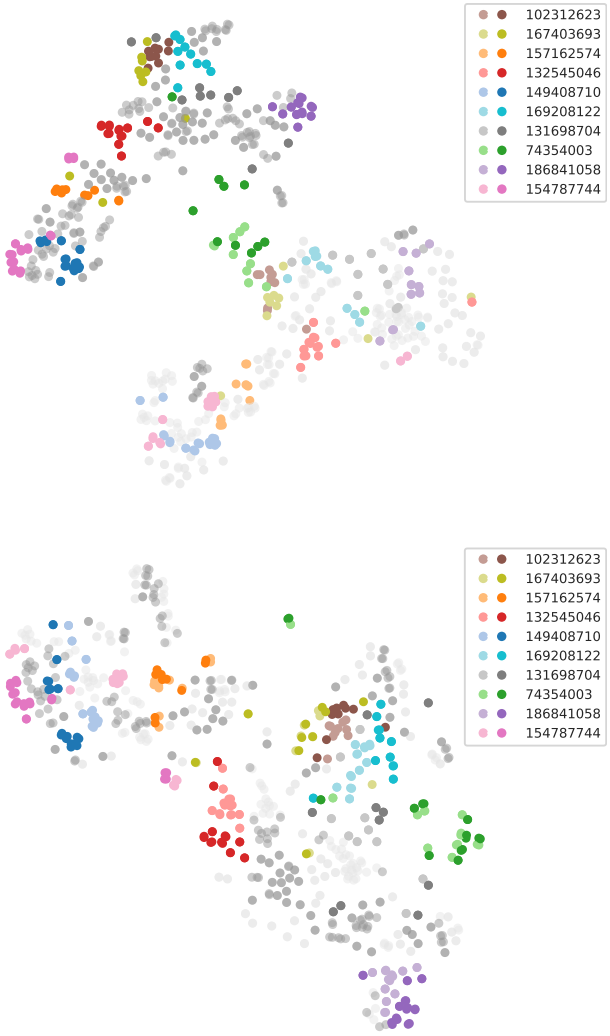


Figure 7: Singer embedding space from `MIXED` model (top) and `CROSS` model (bottom). The label numbers are player IDs from the DAMP dataset. The colors on the left column refers to monophonic vocal tracks; the right column refers to mixed tracks. Best viewed in color.

feature vector from 256 to 2 dimensions using t-distributed stochastic neighbor embedding (t-SNE) [18]. Each dot on the embedding space represents a track. For visualization, we use a paired color palette and assign lighter color to monophonic tracks. Since both monophonic and mixed tracks are from the same singer, an ideal embedding space will show clusters of 20 tracks for each singer.

Figure 7 shows the embedding space learned from the `MIXED` model. There is a noticeable gap between the features of monophonic tracks and that of mixed tracks, which means that the model differentiates monophonic and mixed tracks, rather than finding similar singing voice. Still, we can see that the model is able to cluster tracks from the same singer within the same domain. However, in Figure 7, the monophonic and mixed track features of the same singer are mapped close to each other. This explains why the `CROSS` model shows good performance on cross-domain tasks. We can observe that it is able to transfer singer information across two domains.

6. MOTIVATION FOR FUTURE WORK

Improvement on music mashup: Our mashup pipeline has a large room for improvement. Besides errors produced from existing algorithms, such as key detection, more efforts can be put towards mixing two tracks with a good balance as in real-world recordings. A good automatic mashup system can benefit many areas of research in MIR. The creativity and limitless choices of techniques that can be applied to generate a mashup imply that a large amount of multitrack dataset can be generated for many tasks of interest.

From track to singer modeling: In this work, we use an average of several track-level feature vectors to build singer models. However, in case of singers with highly varying vocal characteristic between different tracks and taking into account the "album effect", averaging may not always be the best choice. Exploring GMMs with multiple mixtures or principal component analysis (PCA) can be an interesting future direction.

Going beyond singing voice: Although we have focused on singing voice, our methods can be tested with tasks involving other instruments, such as multiple instrument recognition. The same mashup technique can be applied to create a dataset, by replacing the monophonic vocal tracks with any instrument of interest. Data generation with mashup may yield better results for instrument recognition in real-world recordings compared to the method proposed in [33], where only two monophonic instrument tracks are used to create a random mix.

7. CONCLUSION

In this paper, we introduced a new problem of cross-domain singer identification and singer-based music retrieval to allow information transfer between monophonic and mixed tracks. Through data generation using music mashup, we were able to train an embedding model to output a joint representation for singing voice from tracks regardless of their domain. We evaluated on three different test scenarios, which include both in-domain and cross-domain cases. A huge advantage of `CROSS` model is that it performs well not only on the cross-domain scenario, but also on commonly observed in-domain scenarios. Therefore, by training only the `CROSS` model, it yields two models, one for each domain. Additional evaluation on varying SNR and on popular music dataset demonstrated that the model is robust to background music and can also be generalized beyond our synthetic dataset.

To conclude, we believe that cross-domain systems can enable many interesting applications related to singing voice, as well as in MIR. Specifically, our future interests include improving the quality of the mashup dataset and performing comparisons between singing voices of karaoke users and that of popular singers.

8. ACKNOWLEDGEMENTS

We thank Keunwoo Choi for valuable comments and reviews. This work was supported by Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Science, ICT & Future Planning (2015R1C1A1A02036962), and by NAVER Corp.

9. REFERENCES

[1] Yann Bayle, Ladislav Maršík, Martin Rusek, Matthias Robine, Pierre Hanna, Katerina Slaninová, Jan Martinovic, and Jaroslav Pokornỳ. Kara1k: a karaoke dataset for cover song identification and singing voice analysis. In *IEEE International Symposium on Multimedia (ISM)*, pages 177–184, 2017.

[2] Wei Cai, Qiang Li, and Xin Guan. Automatic singer identification based on auditory features. In *2011 Seventh International Conference on Natural Computation*, volume 3, pages 1624–1628, 2011.

[3] Tak-Shing Chan, Tzu-Chun Yeh, Zhe-Cheng Fan, Hung-Wei Chen, Li Su, Yi-Hsuan Yang, and Roger Jang. Vocal activity informed singing voice separation with the iKALA dataset. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 718–722. IEEE, 2015.

[4] Matthew EP Davies, Philippe Hamel, Kazuyoshi Yoshii, and Masataka Goto. Automashupper: Automatic creation of multi-song music mashups. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(12):1726–1737, 2014.

[5] Andrew Demetriou, Andreas Jansson, Aparna Kumar, and R Bittner. Vocals in music matter: The relevance of vocals in the minds of listeners. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 514–520, 2018.

[6] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, pages 278–288, 2015.

[7] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, pages 2121–2129, 2013.

[8] Hiromasa Fujihara, Tetsuro Kitahara, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. Singer identification based on accompaniment sound reduction and reliable frame selection. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, pages 329–336, 2005.

[9] Masataka Goto. Singing information processing. In *Proceedings of the 12th IEEE International Conference on Signal Processing (ICSP)*, volume 10, pages 2431–2438, 2014.

[10] Chao-Ling Hsu and Jyh-Shing Roger Jang. On the improvement of singing voice separation for monaural recordings using the mir-1k dataset. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(2):310–319, 2010.

[11] Eric J Humphrey, Sravana Reddy, Prem Seetharaman, Aparna Kumar, Rachel M Bittner, Andrew Demetriou, Sankalp Gulati, Andreas Jansson, Tristan Jehan, Bernhard Lehner, et al. An introduction to signal processing for singing-voice analysis: High notes in the effort to automate the understanding of vocals in music. *IEEE Signal Processing Magazine*, 36(1):82–94, 2019.

[12] Youngmu Kim and Brian Whitman. Singer identification in popular music recordings using voice coding features. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, 2002.

[13] Carol L. Krumhansl. *Cognitive Foundations of Musical Pitch*. Oxford psychology series. Oxford University Press, USA, 1990.

[14] Sangeun Kum and Juhan Nam. Joint detection and classification of singing voice melody using convolutional recurrent neural networks. *Applied Sciences*, 9(7), 2019.

[15] Mathieu Lagrange, Alexey Ozerov, and Emmanuel Vincent. Robust singer identification in polyphonic music using melody enhancement and uncertainty-based learning. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, 2012.

[16] Kyungyun Lee, Keunwoo Choi, and Juhan Nam. Revisiting singing voice detection: a quantitative review and the future outlook. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[17] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

[18] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[19] Namunu Chinthaka Maddage, Changsheng Xu, and Ye Wang. Singer identification based on vocal and instrumental models. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 375–378, 2004.

[20] Brian McFee, Matt McVicar, Stefan Balke, Carl Thomé, Vincent Lostanlen, Colin Raffel, Dana Lee, Oriol Nieto, Eric Battenberg, Dan Ellis, Ryuichi Yamamoto, Josh Moore, WZY, Rachel Bittner, Keunwoo Choi, Pius Friesch, Fabian-Robert Stöter, Matt Vollrath, Siddhartha Kumar, nehz, Simon Waloschek, Seth, Rimvydas Naktinis, Douglas Repetto, Curtis "Fjord" Hawthorne, CJ Carr, João Felipe Santos, JackieWu, Erik, and Adrian Holovaty. librosa/librosa: 0.6.2, August 2018.

[21] Annamaria Mesaros, Tuomas Virtanen, and Anssi Klapuri. Singer identification in polyphonic music using vocal separation and pattern recognition methods. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, pages 375–378, 2007.

[22] Tomoyasu Nakano, Kazuyoshi Yoshii, and Masataka Goto. Vocal timbre analysis using latent dirichlet allocation and cross-gender vocal timbre similarity. In *Acoustics, Speech and Signal Processing, 2014. ICASSP 2014. IEEE International Conference on*, pages 5202–5206, 2014.

[23] Jiyoung Park, Donghyun Kim, Jongpil Lee, Sangeun Kum, and Juhan Nam. A hybrid of deep audio feature and i-vector for artist recognition. In *Joint Workshop on Machine Learning for Music, International Conference on Machine Learning*, 2018.

[24] Jiyoung Park, Jongpil Lee, Jangyeon Park, Jung-Woo Ha, and Juhan Nam. Representation learning of music using artist labels. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[25] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. The MUSDB18 corpus for music separation, December 2017.

[26] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn. Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1-3):19–41, 2000.

[27] Jimena Royo-Letelier, Romain Hennequin, Viet-Anh Tran, and Manuel Moussallam. Disambiguating music artists at scale with audio metric learning. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

[28] Justin Salamon, Emilia Gómez, Daniel PW Ellis, and Gaël Richard. Melody extraction from polyphonic music signals: Approaches, applications, and challenges. *IEEE Signal Processing Magazine*, 31(2):118–134, 2014.

[29] Abhinav Shrivastava, Tomasz Malisiewicz, Abhinav Gupta, and Alexei A Efros. Data-driven visual similarity for cross-domain image matching. In *ACM Transactions on Graphics (ToG)*, volume 30, page 154, 2011.

[30] Jeffrey C Smith. Correlation analyses of encoded music performance. 2013.

[31] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, 2016.

[32] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

[33] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Learning from between-class examples for deep sound recognition. In *International Conference on Learning Representations (ICLR)*, 2018.

[34] Cheng-i Wang and George Tzanetakis. Singing style investigation by residual siamese convolutional neural networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 116–120, 2018.

[35] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2593–2601, 2017.

[36] Xiaofan Zhang, Feng Zhou, Yuanqing Lin, and Shaoting Zhang. Embedding label structures for fine-grained feature representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1114–1123, 2016.

AUGMENTING MUSIC LISTENING EXPERIENCES ON VOICE ASSISTANTS

Morteza Behrooz[1] **Sarah Mennicken**[2] **Jennifer Thom**[2]
Rohit Kumar **Henriette Cramer**[2]
[1] University of California Santa Cruz, Santa Cruz, USA
[2] Spotify USA, Somerville, MA, USA
morteza@ucsc.edu, sarahm@spotify.com

ABSTRACT

Voice interfaces have rapidly gained popularity, introducing the opportunity for new ways to explore new interaction paradigms for music. However, most interactions with music in current consumer voice devices are still relatively transactional; primarily allowing for keyword-based commands and basic content playback controls. They are less likely to contextualize content or support content discovery beyond what users think to ask for. We present an approach to dynamically augment the voice-based music experience with background information using story generation techniques. Our findings indicate that augmentation can have positive effects on voice-based music experiences, given the right user context and mindset.

1. INTRODUCTION

Voice-enabled devices, such as "smart speakers" like Amazon's Echo, Apple's HomePod, Google Home, or Sonos One, have reached the mainstream. In particular, listening to music is a popular use case for such devices [23, 25]. Finding music to listen to and discovering music on these devices can be a challenge as the interactions supported by voice-enabled speakers are relatively limited by current interaction models.

Prior research suggests listeners employ music search to learn and explore about new content to consume. Listeners seek background information to stay informed about their favorite artists, genres, and songs, and use it as a relationship builder with others [19]. This exploratory mindset, however, is relatively rare on music streaming apps because catalog-based entity search does not support this user need well [13]. Augmenting listening experiences and conversational interactions have the potential to support these exploratory user goals but leveraging them for a good user experience remains a challenge.

Learning about background information is sometimes a part of the listening experience itself. Often, such information is presented together with the music playback to contextualize the content. For example, user interfaces of several music streaming services, such as Apple Music, Pandora, and Spotify, include a section for additional information beyond basic track metadata for artists, albums, and playlists. Sometimes, songs are contextualized further by displaying the lyrics, stories, or background information associated with certain parts of the songs (e.g., "Behind the Lyrics" feature on Spotify [27]).

In this paper, we provide a method for how voice-based content consumption can be automatically augmented with background information and present the development and study of a prototype inspired by story generation methods.

We make the following contributions:

- Introduction of a type of content augmentation to contextualize voice-based content consumption with background information in Section 3.
- Detailed design of an approach taking playlists as input and utilizing weighted graphs to generate textual music augmentations, inspired by story generation in Section 3.
- Identification of best practices for using augmentation and conversation in voice-based music consumption in Section 5.

2. RELATED WORK

2.1 Listener Information Needs and Music Search

When listeners search for music, they have multiple information needs that they may be trying to fulfill. These user needs help to shape how listeners approach their music search goals. For instance, listeners may be in the mindset of looking for something specific or they may be in the mindset where they are open to multiple types of music-related information. Prior research has suggested that users of a streaming music service have distinct mindsets when they are searching for music [13]. In a focused mindset, users have one particular item in mind. Catalog, entity-based search interfaces favor this particular mindset and queries that align with the structure of available metadata. In an analysis of Google Answers queries, Bainbridge et al. [3] found that users typically (81.3% of the

time) expressed needs through bibliographic queries, using performer, title of work, or date of recording. Li et al. [17] also observed that typed searches on a streaming music platform are typically focused, suggesting that the modality and design of the current feature supported this type of mindset.

Listeners also have broader information needs that are not met by catalog-based entity searches commonly supported in online music services. Lee et al. [15] observed that people use cloud music services that store listeners' music libraries to listen to music that they were unfamiliar with, suggesting that music discovery and exploration is an important user need. In addition, listeners indicated they search for information about the artists and music for learning purposes [14]. Users of a streaming music platform, however, tended not to use the search feature to deeply learn about a specific type of music and left the platform to fulfill that need [13].

2.2 Voice Assistants and Music Consumption

Voice-enabled speakers currently allow music listeners to search for content (e.g., by saying "play Jazz" or "play Time by Pink Floyd") and control the music playback (e.g., play/pause/skip and volume controls). In fact, these basic playback controls form the most common category of user commands [25]. While many of these speakers can be used in conjunction with a secondary device that has a graphical user interface (GUI), voice interaction is increasingly becoming a primary modality for consuming music [25], which increases the importance of evolving and improving the music experience through voice. Notably though, the voice-only smart speaker experience does not offer much in the way of discovery or background information, and such lack of contextualization and grounding can reduce music discovery and listener's emotional investment [26].

Our work focuses on contextualizing the voice-based music experience with relevant background information. This idea shares similarities to music radio shows, where the hosts provide relevant information about the content they play and add other talking points in between songs. In [4], radio's interaction of speech and content is framed as a special kind of narrative, in which the DJ or radio host is the narrator. One of the main challenges in creating an experience like radio shows is maintaining the "flow" of the music, and balancing the spoken words and songs, as this is one of the main skills of the radio hosts [2]. Our user study seeks to learn more about how to achieve a balance between this flow and providing background information.

2.3 Story Generation

Story generation is the problem of automatically selecting a sequence of events that meet a certain criteria and can be narrated as a story [18]. Story generation and our approach to augment the music listening experience share the goal to generate sequences of textual content given specific constraints. While there are many different approaches to generate stories [12, 20, 28], ours is similar to planning-based approaches which also commonly use graph representations to map the space of story events and the possible constraints of a valid or optimal progression of the storyline. In [18] such constraints are reflected by logical precedence rules, while our method utilizes edge weights and pathfinding to extract a preferred storyline. Similar to PlotShot [9] we apply this a graph-based approach to generate a sequence of text for a given form of input media. Inspired by these different approaches, we take playlists as input and utilize graphs with edge weights that denote content preferences to generate textual music augmentations.

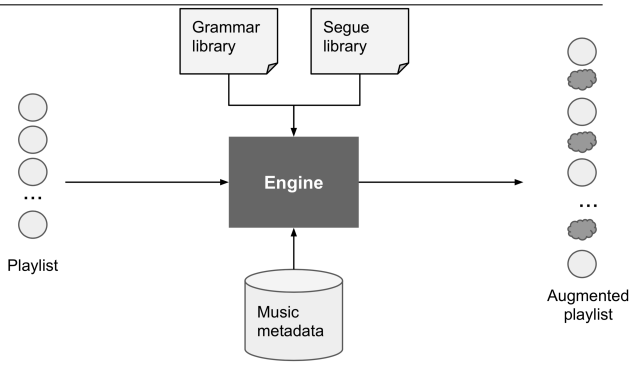


Figure 1. The system architecture of our prototype.

3. APPROACH AND PROTOTYPE

Our prototype takes ordered playlists as input, finds relevant background information and relationships for the songs it contains, and chooses a subset of that information for being used in the output (illustrated in Fig. 1). We call every piece of information that comes in between two consecutive songs a *segue*. Every segue describes a predefined property, such as some information or characteristic, of the next song or a relationship between the current and the next song.

While our approach is not limited to playlists as input, we decided to use them as a starting point given that songs in a playlist typically contain more variety in the metadata as opposed to an individual artist's album. Moreover, songs in a playlist often have an implicit reason for having been grouped together (e.g., being of the same genre, suiting a specific mood or situation [21], artist similarity, etc.) We kept the original order of playlist songs to preserve possible semantic reasons behind curation by playlist creators.

3.0.1 Music Metadata

Our prototype uses a set of metadata and background information about songs, artists, and albums. Table 1 shows some sample entries of qualitative facts about songs and artists consisting of short extractions from publicly available sources of background information (e.g., Wikipedia).

3.0.2 Segue Library and Grammar Library

Table 2 contains a few examples for segues. Each segue has a natural language generation (NLG) template resulting in a "segue text" when realized. Our segue library contains 21 segues. The authoring effort for creating new

Table 1. Songs' metadata examples.

Property	Applicable Entities	Example
Name	Artist, Album, Song	Drake, Scorpion, Wild Thoughts
Genre	Album, Song	Hip Hop
Qualitative fact	Artist, Song	Rihanna's real name is Robyn Rihanna Fenty.

segues simply depends on the complexity of the segue logic.

Inspired by the story generation concept of grammars [7], we defined a simple construct in our system to allow prioritizing authored sequences of segues that are presumed to be interesting. For instance, by preferring a sequence of `ArtistFact`, `ArtistOriginJump`, and `ArtistFact`, an augmentation can focus on the background of songs and their artists. Grammars are an instrument for professional authorship and editorial opinion to be reflected in the system.

3.1 Generating a Sequence of Segues

First, our prototype accesses available metadata about songs, artists, and albums appearing in the playlist. Then it finds all the matching segues for every two consecutive songs which results in a list of segue options for each such position. For the entire playlist, we get a list of these lists, which we call the *story possibility space*. Given that the choice of a segue at each position in this space is independent of other positions, the story possibility space forms a graph and the search problem for finding a sequence of augmentations becomes a problem of finding the best path in this graph. To do so, we use a set of heuristics and preferences which are reflected in a *weighting function*. These scores are assigned as weights to the edges that represent those transitions in our graph.

$$\begin{aligned} weight(s_1, s_2) = {} & diff(s_1, s_2) \\ & + s_1^{pref} + s_2^{pref} \\ & - lengthiness + silence\_reward \\ & + playlist\_reward \\ & + positional\_preference \end{aligned}$$

Several variables enable weighting absolute and relative preferences. $diff(s_1, s_2)$ enables avoiding repetition between consecutive segues. Static "segue preference scores" s_i^{pref} give specific segues authored preference. For example, pointing out a change of genre between two consecutive songs might be more interesting than simply stating title and artist of the next song. Terse responses are often preferred in conversational interactions [10], hence $lengthiness$ punishes a segue if it has a long text and $silence\_reward$ rewards a graph edge if the previous segue is long but the next segue is `NullSegue`. $playlist\_reward$ represents that some segues fit better to a specific type of playlist, such as ArtistQualFact in artist-focused playlists. $positional\_preference$ is used for segues that make only sense at a specific part of a playlist. For example, a playlist introduction with a short authored description only makes sense at the beginning.

Given a weighted graph, we first look for and choose any possible grammar matches. A grammar is a match if there exists a path in a sub-graph of the story possibility space, where the sequence of nodes in that path matches the grammar's sequence of segue types. Edge weights do not have a role in finding a grammar match. If two grammars overlap, we choose the path representing one of them at random.

For the portions of the story possibility space where no grammar match is found, we use the edge weights to find the best path, one with the heaviest sum of weights. If a given portion of the overall graph that needs pathfinding is larger than 5 playlist positions, we find the path step by steps in windows of size 5. In doing so, we ensure that each such window does not contain any segue types that exist in the previous window, and hence avoiding local repetition of segue types.

To exemplify conversational interactions, we identify possible interaction points in which we could trigger a short dialog and let the user response determine which segue option comes next. We do so by checking against simple logic definitions, e.g., if there are specific types of segues in the next list of segue options (see Table 3.)

After the full graph path is determined, we use the realized segue text of the segues in the chosen path, and insert these segue texts in between the songs. An example excerpt of an augmented playlist is shown in Table 4. Our prototype can generate augmentations for any given playlist as long as it has access to the metadata for the songs in that playlist. For our evaluation, we decided to focus on three popular types of playlists to start with those based on an artist, a genre, or listener popularity.

4. EVALUATION

To better understand how our method of adding contextual information to smart speaker experiences affects music listening, we conducted a two-phased study within Spotify.

In phase 1, we gathered feedback from two professional writers who are familiar with the music domain to elicit expert feedback on the content of the segues. They received the written output of our prototype generated for one representative example of each playlist type: artist, genre, and listener popularity. While we invited them to provide any type of feedback, we specifically asked them to share their views on the contents of individual segues and describe how they would approach writing similar content from a professional perspective as writers. After they returned their comments, we conducted a semi-structured interview with both of the writers which took about 45 minutes.

In phase 2, we conducted an internal evaluation with nine Spotify employees (four female, five male) from various parts of the organization to identify potential future improvements and establish a first understanding of user needs. Participants were in their early 20s to late 40s from non-technical functions (such as design, marketing, or operations) and located in various locations across the United

Table 2. Examples for segues, their logic description, and samples for their realized text.

Segue Type	Logic Description	NLG template	Realized text
`NullSegue`	Always a match regardless of the songs.	N/A	N/A
`MundaneSegue`	Always a match regardless of the songs.	Next song is `next_song_name` by `next_song_artist_name`.	Next song is Time by Pink Floyd.
`ArtistOriginJump`	Musical origin of the previous song's artist is different than the next one's.	From `prev_city` where `prev_artist_name`'s musical origins are, to `next_city` where `next_artist_name`'s are.	From Los Angeles where Tupac's musical origins are, to New York City, where Biggie's are.
`SameYearSameArtist`	Previous and next song share the same artist and release year.	Just like the last song, the next song is from `next_song_release_year` by `next_song_artist_name`.	Just like the last song, the next song is from 2007 by Rihanna.

Table 3. Examples for conversational augmentations.

Voice Prompt	User Response	Voice Response
From when do you think this last song was?	Correct	That's right. But the next song, called Shook Ones, Pt. II takes us into a different era. All the way to 1995. (`DifferentEraSegue`)
	Wrong	Actually, it's from 2007. The next song called Shook Ones, Pt. II and [...]. (`DifferentEraSegue`)
Question! Are you more interested in the artist's background or the genre?	Genre	The genre of the upcoming song is called "Latin Trap". (`NextGenreSegue`)
	Artist	Next song is by Cardi B. Here's a fun fact about their biography... (`ArtistQualFact`)

Table 4. Example excerpt of an augmented playlist.

•••

Juicy by The Notorious B.I.G.

Here's The Next Episode by Dr. Dre and Snoop Dogg.

The Next Episode by Dr. Dre

Now switching from the 2001 (Explicit Version) album to one called The Best of 2Pac - Pt. 1: Thug.

California Love by 2Pac

The last song was from 2007. The next song called Shook Ones, Pt. II takes us into a different era. All the way to 1995.

Shook Ones, Pt. II by Mobb Deep

Just like the last song, this song was released in 1995.

Gangsta's Paradise by Coolio

The last and the upcoming song both are described as dark groovy.

•••

States. Each session included a semi-structured interview in which we asked participants about their previous experience with voice assistants and whether or how they look for additional content around music. Each participant was asked to listen to a demo audio file for one of our three playlist types. After answering a short questionnaire, they also interacted with our envisioned conversational experience in a short Wizard-of-Oz (WoZ) demo where an experimenter controlled which content to play. Each playlist type was presented to three users who were randomly assigned to a condition.

The demo consisted of ten shortened songs (first and last 15 seconds) and ten segues (one intro segue, nine transition segues) which were generated using our proposed method and then read by a text-to-speech (TTS) engine. Overall, they had a duration of 5:30-5:50 minutes. The short WoZ section to convey the conversational experience covered three songs only, but between the songs the TTS voice prompted the participant with a potential question such as *"Question! Are you more interested in the artist's background or the genre?"*. Depending on the answer the experimenter chose the next audio file to play to continue the experience. Table 3 shows two examples.

We recorded and transcribed all sessions. Two of the co-authors went independently through the transcripts, first categorizing them for their relevance to the stated research questions and then doing an affinity analysis [5] moving relevant quotes between the high-level categories to derive our findings.

5. FINDINGS

We identified various factors that influence the perception and usefulness of including contextual information in music listening experiences.

5.1 Addressing Listener Needs and Contexts

Music is consumed in vastly different situations, playing a different role for listener's needs in each one. We found that listeners' perceived usefulness of the voice-based augmentations heavily depends on the situation and its unique needs.

Augmentations enable music discovery and education. Augmentations are well received when listeners are in an exploratory mindset. Our participants expressed special interest in voice augmentations to learn about content that is new to them. P4 said: *"[Talking about a playlist containing new songs] I'm like 'Wait, what band is this?' [...] 'What other songs can I listen to from them.' "* P6 described their interest in being able to learn about (niche) genres through such augmentations: *"I feel like metal would work really well for this because a lot of bands have a lot of history behind them [...] it's the opposite of trendy [...] people are still listening to music that was written and performed 20 years ago consistently."*

Similarly, editor 2 saw them as a way of discovering less-known artists by providing information about them: *"[When choosing music automatically] you might end up*

skewing the information toward [...] the top-selling artists of all time; yet obviously there are hugely influential artists that have not sold a lot of records but have impacted other artists and bands." Lastly, P1 brought up the need to identify the right occasions for adding information: *"I like that it's just another way to get to know an artist that you already like and I would potentially like if it was getting to know an artist that you don't know. What I wouldn't like is if it's in between."* This highlights a potential for leveraging listener's level of affinity for an artist they already know, or the predicted level of affinity for a new artist, in determining the quantity or focus of the augmentations.

Activities determine needs for and appropriateness of augmentations. Music often supports a specific listener activity. We found that activities with low cognitive load, such as doing chores or cooking, were commonly mentioned as appropriate contexts for voice-based augmentations. P8 said: *"The perfect experience [is] if I'm at home doing something fun like cooking or something not fun like cleaning."* Activities that require a higher level of focus but that listeners consciously choose to support with background music were perceived to be less suitable. Participants mentioned several examples where the music is serving such an activity-supporting focus like working out, studying, or relaxing and felt that any addition to the music could get into the way of that primary activity. *"I need [the music] to keep the motivation going, keep the music going."* (P8)

5.2 Selection of Appropriate Content for Augmentations

The next category of our findings relates to the content of the augmentation and what it focuses on.

Personalizing augmentations improves the experience. The level of affinity with an artist or genre varies significantly across listeners, and the same is true for the level of familiarity with background information. For example, using a sub-genre to describe a song might be very interesting to someone familiar with the general genre, but vague and uninteresting to someone who is only a casual listener of that type of content. P3 said: *"[...] a high, medium, low, [or] novice/expert setting [would be good], because I'm not an expert on this, so I don't understand [some of the segues]."* Similarly, P1 saw an opportunity to point out to them if they are listening to an artist for the first time: *"Say it's the first time I've listened to an artist, I think it would be cool to learn more about that artist."*

Another frequently mentioned interest for personalization was to allow the listener to adjust the topics that the augmentations focus on (e.g., artist life or genre information). P9 said: *"If I could somehow customize like what's being said by the voice to choose like facts or historical whatever, I think that'd be cool."* Editors had similar views. Editor 2 said: *"we've got one end of the spectrum there is music nerds. They've already put their hands in the air and said, 'Please give me more as much as you can."'* The same editor then drew a parallel between customization of content and augmentation: *"can I add another layer of personalization to this which is, please make [the augmentation] minimal [or] please tell me as much as you can about this artist or this genre."*

Augmentations could explain recommendations or present relevant news. Our study subjects mentioned other types of information that would be useful for them to hear. Music listeners increasingly delegate their choice of music content to streaming services, which use various algorithmic and machine learning methods to choose songs that they believe the listener might enjoy. However, listeners usually do not get any explanation for why a particular set of content is chosen for them. P4 said: *"It kind of guides you to know how they're piecing together this playlist for you. It's like, 'this is why we're playing this song for you',"* and P3 mentioned: *"A lot of times for [automatically generated playlists], I'm like why do I have this song, it would be great if [the voice] could tell me."* Alluding to the same point, Editor 1 noted: *"With just the bare information the name, the title, and to give more information and background obviously [one can] provide a much deeper experience for users and give users the reason why they should continue listening."*

Contextual needs of music listeners often extend to their awareness of the current happenings in the music world. Most prominently, our participants expressed interest in hearing about tours and relevant news headlines. P1 said: *"If they were on tour in my area, that's something I'd want to know,"* and P5 mentioned: *"There's a lot of news always with musicians, whether it's a controversy or other things [...] if you had some of that, like why is this song popular right now or what's going on with this song."* Editor 1 brought up the same point, and discussed the following as an example: *"Let's say [an artist] passes away [...], and you insert a little nugget of information to inform people about that. And then the next song is [by the same artist]. I mean [...] that might make it even more important for maybe someone to listen."*

5.3 Appropriate Presentation of Augmentations

Our last category of findings offers insights about the delivery and presentation of the augmentation content.

TTS voice needs to be trustworthy, high quality and fitting. The synthesized voice in which the segues are presented to the listener was one of the most common topics brought up by our participants; most prominently, the quality of the TTS voice as described by P5: *"With the DJ kind of idea, I think the sound of the thing makes a big difference; so [...] that computery voice takes me out of the moment"*. Despite the quality of the TTS, participants seemed to establish a connection with the agent behind the voice, and explicitly expressed a preference for knowing or at least being able to trust the agent. For instance, P2 said: *"Using someone's voice who is an authority on the genre*

or playlist [is better] [...] there's a difference between that voice telling me little tidbits and somebody like [reference to a Jazz musician]."

Participants mentioned they would like specific properties of the voice, such as gender and accent, to be personalized, either based on the current content or their general preferences. For instance, P4 said: *"I like [it] when people have the Google or Waze, the driving apps, and you can change the accent."* P5 noted: *"I think it would be cool if it was kind of genre-based [...] yea if it's tied to genre or playlist type of thing."* Editor 1 pointed out to voice's gender as well, saying: *"It'd be really jarring to hear like a very male voice [on] Ani DiFranco or Riot grrrl playlists or a very feminist playlist"*.

Augmentations should not be frequent. Participants expressed a preference for segues that connect the previous and the next songs (e.g., by highlighting similarities or differences) over segues that focus solely on the next song. For example, P3 said: *"I like this [...] it tells me a little bit about what I just listened to [...] and then it sets me up into what the next song is going to be,"* and P5 mentioned: *"I [liked] that some of them attempt to link the previous song to the next song."* While semantic continuity is valuable, the frequency of augmentations should not be too high, and segues should not come in between every two songs. We included a representation of an intentional skip (`NullSegue` in Table 2), but it formed either zero or just one out of the 10 generated segues that each participant experienced. Five of our participants (P1, P2, P4, P5, and P8) believed the augmentation was too frequent. P1, for instance, said: *"I definitely in no scenario want [to hear the segues] after every single song"*. Lee et al. [16] found that different user personas have a varying desire for engagement when interacting with music information retrieval systems, which needs to be taken into additional consideration when designing such augmented listening experiences.

Participants enjoyed the conversational augmentations. Our conversational augmentations showcased the ability to ask about the music that is being played, and this was well received by our participants. Most of them (seven out of nine) counted the conversational demo as more fun and interesting than the non-conversational case. When probed on the reasons, participants frequently pointed out the ability to interact. P4, for instance, said: *"You kind of feel like there's this other entity that you're having a conversation with."* In another example, P7 said: *"I think I like this better [than the non-conversational demo]. It was more fun [because of] the interaction aspect of it."* However, two of the participants (P1 and P3) could not imagine themselves using the conversational experience in any situation and generally disliked it. Both participants attributed this dislike to usually preferring a "leaned-back" music consumption mindset, as P3 said: *"I don't want it asking me questions. I actually hated it. It wasn't lean back and was trying to get me to interact..."*

6. LIMITATIONS AND FUTURE WORK

Our results indicate that augmenting voice-based music consumption with background information addresses some of the listener needs that are commonly ignored in current experiences [1, 6]. But similar to how different listening situations affect musical preferences [22], we need to investigate situation-specific preferences for augmentations to understand when music listening is a passive [8], flow-like [11] experience which should not be interrupted.

Our augmentations did not have a "narrative coherence" [24], i.e., a coherent story about a particular topic. In the music context, such narratives could be based on a variety of topics, such as recent events, genres, or artists, all of which were mentioned by our study participants as well. For instance, a dynamically generated augmentation about the history of a genre could focus on songs that represent the turning points of it or have other musical significance. Access to more metadata and large semantic models that capture music-related relationships between various entities can help a story generator in achieving this goal.

In terms of presentation, our evaluation suggests that the quality of the TTS engine seems to be particularly important for music listening experiences. We suspect that the imperfections of the TTS might be more apparent due to a general focus on the audio quality, both for music and voice output. In other use cases for voice assistants, the focus is often more on retrieving the requested information; however, this hypothesis requires further research. Changing the voice's accent or gender, based on explicit listener preference, was stated as an interest by several participants; doing so automatically, such as based on a listener model or audio content, is not only very difficult, but also poses the risk of reinforcing stereotypes of societal and cultural associations for certain types of music.

To minimize negative effects of breaking the audio flow of the music experience, a smoother transition between augmentations and music content is needed. For example, by matching audio properties of the augmentation with those of the surrounding music content, similarly to the techniques used by radio show hosts to match the nearby songs in their ending and beginning [2].

7. CONCLUSIONS

Consuming music via voice assistants is currently a limited experience. While enabling transactional requests for catalog search and basic playback controls, listeners may miss out on context or history of the consumed content. To better support an exploratory mindset for discovering music, we introduce a method that uses story generation techniques to augment the voice-based music experience with relevant background information in between the songs. Our results indicate that adding contextual information to voice-based music interactions can improve smart speaker experience and meet listener needs for music discovery and background information. However, there are also limitations to when the augmentations are desirable based on playback context and listener's mindset and activities.

8. ACKNOWLEDGEMENTS

We thank Keunwoo Choi and Aparna Kumar for their valuable feedback on this paper. Dedicated to Mila Kim, a constant presence during this project.

9. REFERENCES

[1] Music Ally. Everybody's talkin': Smart speakers & their impact on music consumption. http://musically.com/wp-content/uploads/2018/03/SmartSpeakersFinal.pdf, 2018. Accessed: 2018-07-20.

[2] Anupriya Ankolekar, Thomas Sandholm, and Louis Lei Yu. Evaluating mobile music experiences: Radio on-the-go. In *International Conference on Mobile Computing, Applications, and Services*, pages 56–73. Springer, 2018.

[3] David Bainbridge, Sally Jo Cunningham, and J. Stephen Downie. How people describe their music information needs: A grounded theory analysis of music queries. In *ISMIR*, pages 221–222, 2003.

[4] Jody Berland. Radio space and industrial time: music formats, local narratives and technological mediation 1. *Popular Music*, 9(2):179–192, 1990.

[5] Hugh Beyer and Karen Holtzblatt. *Contextual design: defining customer-centered systems*. Elsevier, 1997.

[6] Billboard. From mood playlists to metadata: How smart speakers are the next frontier – and challenge – for the music business. https://www.billboard.com/articles/business/8263197/smart-speaker-challenges-music-business, 2018. Accessed: 2018-07-20.

[7] William F Brewer and Edward H Lichtenstein. Event schemas, story schemas, and story grammars. *Center for the Study of Reading Technical Report; no. 197*, 1980.

[8] Steven Brown, Michael J Martinez, and Lawrence M Parsons. Passive music listening spontaneously engages limbic and paralimbic systems. *Neuroreport*, 15(13):2033–2037, 2004.

[9] Rogelio E Cardona-Rivera and Boyang Li. Plotshot: Generating discourse-constrained stories around photos. In *Proceedings of the 12th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2016.

[10] Ana Paula Chaves and Marco Aurelio Gerosa. Single or multiple conversational agents?: An interactional coherence comparison. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 191. ACM, 2018.

[11] Frank M Diaz. Mindfulness, attention, and flow during music listening: An empirical investigation. *Psychology of Music*, 41(1):42–58, 2013.

[12] Belén Díaz-Agudo, Pablo Gervás, and Federico Peinado. A case based reasoning approach to story plot generation. In *European Conference on Case-Based Reasoning*, pages 142–156. Springer, 2004.

[13] Christine Hosey, Lara Vujovic, Brian St. Thomas, Jean Garcia-Gathright, and Jennifer Thom. Just give me what I want: How people use and evaluate music search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2019.

[14] Jin Ha Lee, Hyerim Cho, and Yea-Seul Kim. Users' music information needs and behaviors: Design implications for music information retrieval systems. *Journal of the association for information science and technology*, 67(6):1301–1330, 2016.

[15] Jin Ha Lee, Yea-Seul Kim, and Chris Hubbles. A look at the cloud from both sides now: An analysis of cloud music service usage. In *ISMIR*, pages 299–305, 2016.

[16] Jin Ha Lee and Rachel Price. Understanding users of commercial music services through personas: Design implications. In *ISMIR*, pages 476–482, 2015.

[17] Ang Li, Jennifer Thom, Praveen Chandar, Christine Hosey, Brian St. Thomas, and Jean Garcia-Gathright. Search mindsets: Understanding focused and non-focused information seeking in music search. In *Proceedings of the 30th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 2019.

[18] Boyang Li, Stephen Lee-Urban, George Johnston, and Mark Riedl. Story generation with crowdsourced plot graphs. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[19] Adam J Lonsdale and Adrian C North. Why do we listen to music? a uses and gratifications analysis. *British Journal of Psychology*, 102(1):108–134, 2011.

[20] Lara J Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark O Riedl. Event representations for automated story generation with deep neural nets. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[21] Esa Nettamo, Mikko Nirhamo, and Jonna Häkkilä. A cross-cultural study of mobile music: retrieval, management and consumption. In *Proceedings of the 18th Australia conference on Computer-Human Interaction: Design: Activities, Artefacts and Environments*, pages 87–94. ACM, 2006.

[22] Adrian C North and David J Hargreaves. Situational influences on reported musical preference. *Psychomusicology: A Journal of Research in Music Cognition*, 15(1-2):30, 1996.

[23] Edison Research NPR. The smart audio report. https://www.nationalpublicmedia.com/smart-audio-report/, 2017. Accessed: 2018-07-20.

[24] Mark O Riedl. A comparison of interactive narrative system approaches using human improvisational actors. In *Proceedings of the intelligent narrative technologies III workshop*, page 16. ACM, 2010.

[25] Alex Sciuto, Arnita Saini, Jodi Forlizzi, and Jason I Hong. Hey alexa, what's up?: A mixed-methods studies of in-home conversational agent usage. In *Proceedings of the Designing Interactive Systems Conference*, pages 857–868. ACM, 2018.

[26] Elliot Jay Stocks. Music consumption in the era of smart speakers. https://medium.com/@elliotjaystocks/music-consumption-in-the-era-of-smart-speakers-b88d04a18746, 2017. Accessed: 2018-07-20.

[27] The Verge. Spotify and genius are collaborating on info-rich behind the lyrics playlists. https://www.theverge.com/2016/1/12/10750990/spotify-genius-behind-the-lyrics-playlists-iphone, 2016. Accessed: 2018-07-25.

[28] R Michael Young, Stephen G Ware, Brad A Cassell, and Justus Robertson. Plans and planning in narrative generation: a review of plan-based approaches to the generation of story, discourse and interactivity in narratives. *Sprache und Datenverarbeitung, Special Issue on Formal and Computational Models of Narrative*, 37(1-2):41–64, 2013.

COUPLED RECURRENT MODELS FOR POLYPHONIC MUSIC COMPOSITION

John Thickstun[1] **Zaid Harchaoui**[2] **Dean P. Foster**[3] **Sham M. Kakade**[1,2]
[1] Allen School of Computer Science and Engineering, University of Washington, Seattle, WA, USA
[2] Department of Statistics, University of Washington, Seattle, WA, USA
[3] Amazon, NY, USA
{thickstn,sham}@cs.washington.edu, zaid@uw.edu, foster@amazon.com

ABSTRACT

This paper introduces a novel recurrent model for music composition that is tailored to the structure of polyphonic music. We propose an efficient new conditional probabilistic factorization of musical scores, viewing a score as a collection of concurrent, coupled sequences: i.e. voices. To model the conditional distributions, we borrow ideas from both convolutional and recurrent neural models; we argue that these ideas are natural for capturing music's pitch invariances, temporal structure, and polyphony. We train models for single-voice and multi-voice composition on 2,300 scores from the KernScores dataset.

1. INTRODUCTION

In this work we will think of a musical score as a sample from an unknown probability distribution. Our aim is to learn an approximation of this distribution, and to compose new scores by sampling from this approximation. For a broad survey of approaches to automatic music composition, see [9]; for a more targeted survey of classical probabilistic approaches, see [3]. We note the success of parameterized, probabilistic generative models in domains where problem structure can be exploited by models: convolutions in image generation, or autoregressive models in language modeling. This work examines autoregressive models of scores (Section 3): how to evaluate these models, how to build the structure of music into parameterized models, and the effectiveness of these modeling choices.

We study the impact of structural modeling assumptions via a cross-entropy measure (Section 4). It is reasonable to question whether cross-entropy is a good surrogate measure for the subjective quality of sampled compositions. In theory, a sufficiently low cross-entropy indicates a good approximation of the target distribution and therefore must correspond to high-quality samples. In practice, we observe of other generative modeling tasks that learned models do achieve sufficiently low cross-entropy to produce qualitatively good samples [4, 19, 29]. Studying the cross-entropy allows us to explore many models with various structural assumptions (Section 5). Finally, we provide a qualitative evaluation of samples from our best model to demonstrate that these models have sufficiently small cross-entropy for samples to exhibit a degree of subjective quality (Section 6). Supplementary material including appendices, compositional samples, and code for the experiments is available online. [1]

2. RELATED WORKS

In this work, we consider both single-voice models and multi-voice, polyphonic models. Early probabilistic models of music focused on single-voice, monophonic melodies. The first application of neural networks to melody composition was proposed by [28]. This work prompted followup [18] using an alternative data representation inspired by pitch geometry ideas [25]; the relative pitch and note-embedding schemes considered in the present work can be seen as a data-driven approach to capturing some of these geometric concepts. For recent work on monophonic composition, see [12, 23, 26].

Work on polyphonic music composition is considerably more recent. Early precurors include [15], which considers two-voice composition, and [5], which proposes an expert system to harmonize 4-voice Bach chorales. The harmonization task became popular, along with the Bach chorales dataset [1]. Multiple voice polyphony is directly addressed in [16], albeit using a simplified preprocessed encoding of scores that throws away duration information.

Maybe the first work with a fair claim to consider polyphonic music in full generality is [2]. This paper proposes a coarse discrete temporal factorization of musical scores (for a discussion of this raster factorization and others, see Section 3) and examines the cross-entropy of a variety of neural models on several music datasets (including the Bach chorales). Many subsequent works on polyphonic models use the dataset, encoding, and quantitative metrics introduced in [2], notably [30] and [13]. We also note recent, impressive work on the closely related problem of modeling expressive musical performances [11, 20].

Many recent works focus exclusively on the Bach

[1] http://homes.cs.washington.edu/~thickstn/ismir2019composition/

chorales dataset [7, 10, 17]. The works [7, 17] evaluate their models using qualitative large-scale user studies. The system proposed in [7] optimizes a pseudo-likelihood, so its quantitative losses cannot be directly compared to generative cross-entropies. The generative model proposed in [17] could in principle report cross entropies, but this work also focuses on a qualitative study. Quantitative cross-entropy metrics on the chorales are analyzed in [10]. Both [7] and [10] propose non-sequential Gibbs-sampling schemes for generation, in contrast to the ancestral samplers used in [17] and in the present work.

3. FACTORING THE DISTRIBUTION OVER SCORES

Polyphonic scores consist of notes and other features of variable length that overlap each other in quasi-continuous time. Scores contain a vast heterogenous collection of information, much of which we will not attempt to model: time signatures, tempi, dynamics, etc. We will therefore give a working definition of a score that captures the pitch, rhythmic, and voicing information we plan to model. We define a score of length T beats as a continuous-time, matrix-valued sequence $\mathbf{x}$, where $\mathbf{x}_t \in \{0,1\}^{V\times 2P}$ for each time $t \in [0,T]$. Specifically, for each voice $v \in \{1,\dots,V\}$ and each pitch $p \in \{1,\dots,P\}$ we set

$$\mathbf{x}_{t,v,p} = 1 \quad \text{iff pitch } p \text{ is on at time } t \text{ in voice } v, \quad (1)$$

$$\mathbf{x}_{t,v,P+p} = 1 \quad \text{iff pitch } p \text{ begins at time } t \text{ in voice } v. \quad (2)$$

Both "note" bits (1) and "onset" bits (2) are required to represent a score, expressing the distinction between a sequence of repeated notes of the same pitch and a single sustained note; see Appendix C for further discussion.

Let q denote the (unknown) probability distribution over scores $\mathbf{x}$. Score are high dimensional objects, of which we have limited samples (2,300 – see Section 4). Rather than directly model q, we will serialize $\mathbf{x}$, factor q according to this serialization, and model the resulting conditional distributions $q(\cdot|\text{history})$. There are many possible ways to factor q; in the remainder of this section we review the popular raster factorization, and propose a new sequential factorization based on voices.

Raster score factorization. Many previous works factor a score via rasterization. If we sample a score $\mathbf{x}$ at constant intervals Δ and impose an order on parts and notes, we can factor the distribution q over scores as $q(\mathbf{x}) =$

$$\prod_{k=1}^{T/\Delta}\prod_{v=1}^{V}\prod_{p=1}^{2P} q(\mathbf{x}_{k\Delta,v,p}|\mathbf{x}_{1:k\Delta},\mathbf{x}_{k\Delta,1:v},\mathbf{x}_{k\Delta,v,1:p}). \quad (3)$$

Throughout this work, a slice $1\!:\!i$ is inclusive of the first index 1 but does not include the final index i.

This factorization generates music in sequential Δ-slices of time. Some prior works directly model the (high-dimensional) distribution $\mathbf{x}_{k\Delta}$; this approach was pioneered by [2], using NADE to model the conditional distributions $q(\mathbf{x}_{k\Delta}|\mathbf{x}_{1:k\Delta})$. Others impose further order on notes (and voicings, if they choose to model them) and factor the distribution into binary conditionals as in (3). Notes are typically ordered based on pitch, either low-to-high [7] or high-to-low [17].

Sequential voice factorization. Putting full scores aside for now, consider factoring a single voice v, i.e. a slice $\mathbf{x}_{1:T,v,1:2P}$ of a score. By definition, a Kern-Scores voice is homophonic in the sense that its rhythms proceed in lock-step: a voice consists of a sequence of notes, chords, or rests, and no notes are sustained across a change point.[2] Instead of generating raster time slices, suppose we run-length encode the durations between change points in v. We denote these change points by $c_0^v,\dots,c_{L_v}^v$ where L_v is the number of change points in voice v. Let D be the number of unique distance between change points, and define a run-length encoded voice $\mathbf{r} \in \left(\{0,1\}^D \oplus \{0,1\}^N\right)^{L_v}$. At each index $k \in \{1,\dots,L_v\}$, $\mathbf{r}_k = (\mathbf{r}_{k,0},\mathbf{r}_{k,1})$ with $\mathbf{r}_{k,0} \in \{0,1\}^D$ and $\mathbf{r}_{k,1} \in \{0,1\}^N$ such that

$$\mathbf{r}_{k,0} = \mathbf{1}_{d_k} \quad \text{where } d_k = \frac{c_{k+1}^v - c_k^v}{\Delta} \in \mathbb{N},$$

$$\mathbf{r}_{k,1,p} = 1 \quad \text{iff pitch } p \text{ begins at time } c_k^v \text{ in voice } v.$$

The durations d_k correspond to note-values (quarter-note, eighth-note, dotted-half, etc.). We proceed to factor the voice sequentially as $p(\mathbf{r}) =$

$$\prod_{k=1}^{L_v} q(\mathbf{r}_{k,0}|\mathbf{r}_{1:k}) \prod_{p=1}^{P} q(\mathbf{r}_{k,1,p}|\mathbf{r}_{1:k},\mathbf{r}_{k,0},\mathbf{r}_{k,1,1:p}). \quad (4)$$

Sequential score factorization. We now consider a sequential factorization that interlaces predictions in the score's constituent voices. The idea is to predict voices sequentially as we did in the previous section, but we must now choose the order across voices in which we make predictions. The rule we choose is to make a prediction in the voice that has advanced least far in time, breaking ties by the arbitrary numerical order assigned to voices (ties happen quite frequently: for example, at the beginning of a score when all parts have advanced 0 beats). This ensures that all voices are generated in near lock-step; generation in any particular voice never advances more than one note-value ahead of any other voice.

Mathematically, we can describe this factorization as follows. First, we impose a total order on change points c_k^v across voices by the rule $c_k^v < c_{k'}^u$ for all v,u if $k<k'$ and $c_k^v < c_k^u$ if $v<u$. Define $L \equiv \sum_{v=1}^{V} L_v$. For index $i \in \{1,\dots,L\}$ let α_i and β_i denote the index and voice of the corresponding change point according the the total ordering on change points. We define a sequentially encoded score $\mathbf{s} \in (\{0,1\}^D \oplus \{0,1\}^N)^L$ by

$$\mathbf{s}_{k,0} = \mathbf{1}_{d_k} \quad \text{where } d_k = \frac{c_{\alpha_k+1}^{\beta_k} - c_{\alpha_k}^{\beta_k}}{\Delta} \in \mathbb{N},$$

$$\mathbf{s}_{k,1,p} = 1 \quad \text{iff pitch } p \text{ begins in voice } \beta_k \text{ at time } c_{\alpha_k}^{\beta_k}.$$

And we factor the distribution sequentially by $q(\mathbf{s}) =$

$$\prod_{k=1}^{L} q(\mathbf{s}_{k,0}|\mathbf{s}_{1:k}) \prod_{p=1}^{P} q(\mathbf{s}_{k,1,p}|\mathbf{s}_{1:k},\mathbf{s}_{k,0},\mathbf{s}_{k,1,1:p}). \quad (5)$$

[2] For polyphonic instruments like the piano, we must adopt a more refined definition of a voice than "notes assigned to a particular instrument;" see Appendix B for details.

4. DATASET AND EVALUATION

Dataset. The models presented in this paper are trained on KernScores data [24], a collection of early modern, classical, and romantic era digital scores assembled by musicologists and researchers associated with Stanford's CCARH. [3] The dataset consists of over 2,300 scores containing approximately 2.8 million note labels. Tables 1 and 2 give a sense of the contents of the dataset.

We contrast this dataset's Humdrum encoding with the MIDI encoded datasets used by most works discussed in this paper. [4] MIDI was designed as a protocol for communicating digital performances, rather than digital scores. This is exemplified by the MAPS [6] and MAESTRO [8] datasets, which consist of symbolic MIDI data aligned to expressive performances. While this data is symbolic, it cannot be interpreted as scores because it is unaligned to a grid of beats and does not encode note-values (quarter-note, eighth-note, etc). Some MIDI datasets are aligned to a grid of beats, for example MusicNet [27]. But heuristics are still necessary to interpret this data as visual scores. For example, many MIDI files encode "staccatto" articulations by shortening the length of notes, thwarting simple rules that identify note-values based on length.

Evaluation. Let $\hat{q}$ be an estimate of the unknown probability distribution over scores q. We want to measure the quality of $\hat{q}$ by its cross-entropy to q. Because the entropy of a score grows with its length T, we will consider a cross-entropy rate. By convention, we measure time in units of beats, so the cross-entropy rate has units of bits per beat.

Defining cross-entropy for a continuous-time process generally requires some care. But for music, defining the cross-entropy on an appropriate discretization will suffice. Musical notes begin and end at rational fractions of the beat, and therefore we can find a common denominator d of all change points in the support of the distribution q (for our dataset $d = 48$). For a score of length T beats, we partition the interval $[0, T]$ into constant subintervals of length $\Delta \equiv 1/d$ and define a rate-adjusted, discretized cross-entropy

$$H_{\mathcal{P}}(q||\hat{q}) \equiv \mathop{\mathbb{E}}_{\mathbf{x}\sim q}\left[-\frac{1}{T\Delta}\log \hat{q}(\mathbf{x}_0, \mathbf{x}_\Delta, \mathbf{x}_{2\Delta}, \ldots, \mathbf{x}_T)\right]. \tag{6}$$

Proposition 1 in Appendix F shows that we can think of Δ as the resolution of the score process, in the sense that any further refinement of the discretization d yields no further contributions to the cross entropy.

Definition 6 is independent of any choice about how we factor q: it is a cross entropy measure of the joint distribution over a full score. As we discussed in Section 3, there are many ways to factor a generative model of scores. These choices lend themselves to different natural cross-entropies, each with their own units. By measuring in units of bits per beat at the process resolution Δ as defined by Definition 6, we can compare results under different factorizations.

[3] http://kern.ccarh.org/

[4] A notable exception is [16], which uses data derived from the KernScores collection considered here.

Computational cost. Raster models are expensive to train and evaluate on rhythmically diverse music. A raster model must be discretized at the process resolution Δ to generate a score with precise rhythmic detail. The process resolution Δ of a corpus containing both triplets and sixty-fourth notes is $\Delta = 3 \times 16 = 48$ positions per beat. Corpora with quintuplet patterns require a further factor of 5, resulting in $\Delta = 240$. To generate a score from a raster factorization requires Δ predictions per beat; to ease the computational burden of prediction, when the raster approach is taken scores are typically discretizing at either 1 or 2 positions per beat [2]. Unfortunately, this discretization well above the process resolution leads to dramatic rhythmically simplification of scores (see Appendix C).

In contrast, a sequential factorization such as (4) or (5) requires predictions proportional to the average number of notes per beat, while maintaining the rhythmic detail of a score. The KernScores single-voice corpus averages ≈ 1.25 notes per beat, requiring 1.25 predictions per beat for sequential factorization versus Δ predictions per beat for raster factorization. The KernScores multi-voice corpus averages ≈ 5 notes per beat, requiring 5 predictions per beat for sequential factorization, an order of magnitude less than the $\Delta \approx 50$ predictions per beat required for raster prediction.

5. MODELS AND WEIGHT-SHARING

Modeling voices allows us to think of the polyphonic composition problem as a collection of correlated single-voice composition problems. Learning the marginal distribution over a single voice v is similar in spirit to classical monophonic tasks. Learning the distribution over KernScores voices generalizes this classical task to allow for chords: formally, a monophonic sequence would require the vector $\mathbf{r}_{k,1} \in \{0, 1\}^N$ described in Section 3 to be one-hot, whereas our our dataset includes voices where this vector is multi-hot, expressing intervals and chords (e.g. chords in the left hand of a piano, or double-stops for a violin).

We will explore two modeling tasks. First we consider a single-voice prediction task: learn the marginal distribution over a voice v, estimating the conditionals that appear in the factorization (4). Results on this task are summarized in Table 3. Second we consider a multi-voice prediction task: learn the joint distribution over scores, estimating the conditionals that appear in the factorization (5). Results on this task are summarized in Table 4.

5.1 Representation

Like our choice of factorization, we are faced with many options for encoding the history of a score for prediction. Some of the same computational and modeling considerations apply to both the choice of a factorization and the choice of a history encoding, but these are not inherently connected decisions. For the single-voice task, we use the encoding $\mathbf{r}$ introduced to define the sequential voice factorization in Section 3.

For the polyphonic task, we also encode history using a run-length encoding. Let $c_1, \ldots, c_K$ denote change points in the full score $\mathbf{x}$, let $d_j^v \equiv (c_{j+1}^v - c_j^v)/\Delta \in \mathbb{N}$, and define

Bach	Beethoven	Chopin	Scarlatti	Early	Joplin	Mozart	Hummel	Haydn
191,374	476,989	57,096	58,222	1,325,660	43,707	269,513	3,389	392,998

Table 1. Notes in the KernScores dataset, partitioned by composer. The "Early" collection consists of Renaissance vocal music; a plurality of the Early music is composed by Josquin.

Vocal	String Quartet	Piano
1,412,552	820,152	586,244

Table 2. Notes in the KernScores dataset, partitioned by ensemble type.

a sequence $\mathbf{e} \in (\{0,1\}^{D+1} \oplus \{0,1\}^{P})^{K \times V}$ where

$\mathbf{e}_{k,v,0,0:D} = \mathbf{1}_{d_j^p}$ iff $c_k = c_j^v$ for some c_j^v in voice v,

$\mathbf{e}_{k,v,0,D} = 1$ iff c_k is not a change point in voice v,

$\mathbf{e}_{k,v,1,p} = 1$ iff pitch p begins in voice v at time c_k.

This is not the fully serialized encoding **s** used to define a score factorization (for discussion of a fully sequential representation, see [20]). At each time step k for which any voice exhibits a change point, we make an entry in **e** for every voice; we refer to $\mathbf{e}_k$ as a frame. This requires us to augment our alphabet of duration symbols D with a special continuation symbol that indicates no change (comparable to the onset bits in the encoding **x**). An advantage of this representation over sequential or raster representations is that more history can be encoded with shorter sequences.

For a fixed voice v, let $\tilde{\mathbf{r}} \equiv \mathbf{e}_{:,v}$ be a single-voice slice of the score history. Observe that $\tilde{\mathbf{r}} \neq \mathbf{r}$, where **r** is the run-length encoding used for the single-voice task. The slices $\tilde{\mathbf{r}}$ are spaced out with aforementioned continuation symbols where there are change points in other voices. With the single-voice encoding **r**, simple linear filters can be learned that are sensitive to particular rhythmic sequences: e.g. groups of four eighth notes, or three triplet-quarter notes. This is not the case for $\tilde{\mathbf{r}}$; rhythmic patterns can be somewhat-arbitrarily broken up by continuation symbols.

These observations might lead us to consider raster encodings for multi-voice history, which restore the effectiveness of simple linear filters at the cost of increasing the dimensionality of the history encoding. We find that recurrent networks for the single-voice task are unhampered when retrained on $\tilde{\mathbf{r}}$: compare experiments 21 and 22 in Table 3. Performance falls slightly when learning on $\tilde{\mathbf{r}}$, but this is to be expected because history interspersed with continuations is effectively a shorter-length history.

For both the single-voice and multi-voice tasks, we truncate the history at a fixed number of frames prior to the prediction time. We explore several history lengths in the experiments and observe diminishing improvement in quantitative results for windows beyond the range of 10-20 frames of **e**: see experiment group (1,2,6,7) in Table 4.

5.2 Single-voice models

Using factorization (4), we explore fully connected, convolutional, and recurrent models for learning the conditional distributions $q(\mathbf{r}_{k,0}|\mathbf{r}_{1:k})$ over note-values and $q(\mathbf{r}_{k,1,n}|\mathbf{r}_{1:k}, \mathbf{r}_{k,0}, \mathbf{r}_{k,1,1:p})$ over pitches. We build separate models to estimate $\mathbf{r}_{k,0}$ and $\mathbf{r}_{k,1,p}$, with respective losses Loss_t and Loss_n. In the remainder of this section, we consider opportunities to exploit structure in music by sharing weights in our models. Quantitative results for single-voice models are summarized in Table 3, with additional details available in Appendix A.

Autoregressive modeling. To build a generative model over conditionally stationary sequential data, it often makes sense to make the autoregressive assumption $q(\mathbf{r}_k|\mathbf{r}_{1:k}) = q(\mathbf{r}_{k'}|\mathbf{r}_{1:k'})$ for all $k, k' \in \mathbb{N}$. We can then learn a single conditional approximation $\hat{q}(\mathbf{r}_k|\mathbf{r}_{1:k})$ and share model parameters across all time translations.

Scores are not quite conditionally stationary; the distribution of notes and rhythms varies substantially depending on the sub-position within a beat. To address this, we follow the lead of [13] and [7] and augment our history tensor with a one-hot location feature vector ℓ that indicates the subdivision of the beat for which we are presently making predictions. [5] Compare the loss of duration models (Loss_t) with and without these features in experiment pairs (3,4), (6,7), (10,11), (12,13), and (15,16).

Relative pitch. We can perform a similar weight-sharing scheme with pitches as we did with time. Instead of building an individual predictor for each pitch conditioned on the notes in the history tensor, we adopt an idea proposed in [13]: build a single predictor that conditions on a shifted version of the history tensor centered around the note we want to predict. Convolving this predictor over the pitch axis of the history tensor lets us make a prediction at each note location, as visualized by Figure 1.

As with time, the distribution over notes is not quite conditionally stationary. For example, a truly relative predictor would generate notes uniformly across the note-class axis, whereas the actual distribution of notes concentrates around middle C. Therefore we augment our history tensor with a one-hot feature vector $\mathbf{1}_p$ that indicates the pitch p for which we are making a prediction. This allows us to take full advantage of all available information when making a prediction, while borrowing strength from shared harmonic patterns in different keys or octaves. We compare absolute pitch-indexed classifiers ($\mathbf{lin}_p$) to a single, relative pitch classifier (**lin**) in Table 3: compare the loss of pitch models (Loss_p) in experiment groups (2,3,4), (5,6,7), (8,9,10), (11,12,13), and (15,16).

Relative pitch models serve a similar purpose to key-signature normalization [17] or data augmentation via transposition [7]. Building this invariance into the model

[5] Location can always be computed from a full history. But we truncate the history, so this information is lost unless it is explicitly reintroduced.

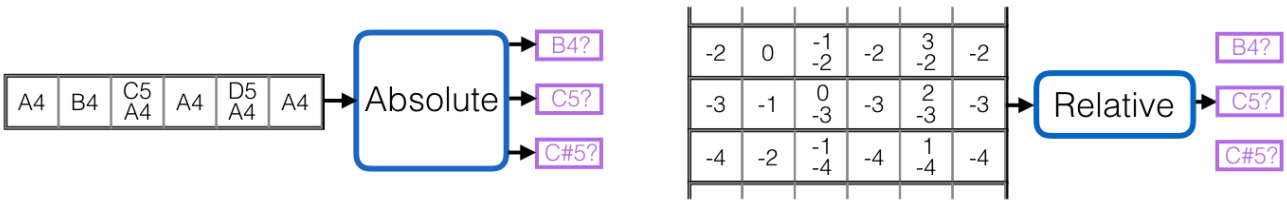


Figure 1. Left: an absolute pitch predictor learns individual classifiers for each pitch-class. Right: a relative pitch predictor learns a single classifier and translates the data along the frequency axis to center it around the pitch to be predicted. Whereas the absolute predictor decides whether C5 is on given the previous note was A4, the relative predictor decides whether the note under consideration is on given the previous note was 3 steps below it.

offers an alternative approach, avoiding data preprocessing or the introduction of hyper-parameters. We find that training with transpositions in the range ± 5 semi-tones yields no performance increase for relative pitch models.

Pitch embeddings. Borrowing the concept of a word embedding from natural language processing, we consider learned embeddings $\mathbf{c}$ of the pitch vectors $\mathbf{r}_{k,1}$ ($\mathbf{e}_{k,v,1}$ for the multi-voice models). For recurrent models, we do not see performance benefits to learning these embeddings: compare experiments 20 and 21 in Table 3. However, we do find that we can learn compact embeddings (16 dimensions for the experiments presented in this paper) without sacrificing performance, and using these embeddings reduces computational cost. We also find that using a 12 dimensional fixed embedding of pitches $\mathbf{f}$, in which we quotient each pitch class by octave, reduces overfitting for the rhythmic model while preserving predictive accuracy.

5.3 Multi-voice models

Using the factorization (5), we now explore ways to capture correlations between the voices and model the conditional distributions $q(\mathbf{s}_{k,0}|\mathbf{s}_{1:k})$ over note-values and $q(\mathbf{s}_{k,1,p}|\mathbf{s}_{1:k}, \mathbf{s}_{k,0}, \mathbf{s}_{k,1,1:p})$ over notes. We build separate models to estimate $\mathbf{r}_{k,0}$ and $\mathbf{r}_{k,1,p}$, with losses Loss_t and Loss_p in Table 4 respectively. The same structural observations that we made about scores for the single-voice models apply to multi-voice modeling; all multi-voice models considered in this paper use the three weight-sharing schemes considered for single-voice models. We explore an additional weight-sharing opportunity below for the multi-voice task: voice decomposition.

The effectiveness of recurrent models for the single-voice modeling task, and the representational considerations in Section 5.1, motivate us to consider extensions of the recurrent architecture to capture structure in the multi-voice setting. One natural extension of the standard recurrent neural network to model multiple, concurrent voices is a hierarchical architecture, illustrated in Figure 2:

$$
\begin{aligned}
h_{k,v}(\mathbf{e}) &\equiv \mathbf{a}\left(W_v^\top h_{k-1,v}(\mathbf{e}) + W_e^\top \mathbf{c}\left(\mathbf{e}_{k,v}\right)\right), \\
g_k(\mathbf{e}) &\equiv \mathbf{a}\left(W_g^\top g_{k-1}(\mathbf{e}) + W_{hv}^\top \sum_u h_{k,u}(\mathbf{e})\right).
\end{aligned} \tag{7}
$$

The first equation is a standard recurrent network that builds a state estimate $h_{k,v}$ of a voice v at time index k based on transition weights W_v, an input embedding $\mathbf{c}$, input weights W_e, and non-linear activation $\mathbf{a}$ (we use a ReLU activation). We integrate the state of each voice (weights W_{hv}) into a global state g_k given the previous global state g_{k-1} (weights W_g). Because voice order is arbitrary in our dataset, we sum (i.e. pool) over their states before feeding them into the global network. At each time k, we use the learned state of each voice together with the global state to make a note-value prediction: $\hat{\mathbf{s}}_{k,0} = \mathbf{lin}(h_{k,\beta_k}(\mathbf{e}), g_k(\mathbf{e}))$, where $\mathbf{lin}$ is a log-linear classifier. We make pitch predictions $\mathbf{s}_{k,1,p} \in \{0, 1\}$ using the same architecture. We learn a single, relative-pitch classifier for $\mathbf{s}_{k,1,p} \in \{0, 1\}$ in all multi-voice experiments

#	History	Arch	Loc?	Relative?	Pitch?	Embed?	Loss
1	$\mathbf{r}_{(1)}$	bias	no	no	no	no	10.07
2	$\mathbf{r}_{(1)}$	linear	no	no	no	no	8.05
3	$\mathbf{r}_{(1)}$	linear	no	yes	no	no	6.29
4	$\mathbf{r}_{(1)}$	linear	yes	yes	yes	no	6.12
5	$\mathbf{r}_{(1)}$	fc	no	no	no	no	5.92
6	$\mathbf{r}_{(1)}$	fc	no	yes	no	no	6.05
7	$\mathbf{r}_{(1)}$	fc	yes	yes	yes	no	5.70
8	$\mathbf{r}_{(5)}$	linear	no	no	no	no	7.91
9	$\mathbf{r}_{(5)}$	linear	no	yes	no	no	5.76
10	$\mathbf{r}_{(5)}$	linear	yes	yes	yes	no	5.63
11	$\mathbf{r}_{(5)}$	fc	no	no	no	no	4.90
12	$\mathbf{r}_{(5)}$	fc	no	yes	no	no	4.80
13	$\mathbf{r}_{(5)}$	fc	yes	yes	yes	no	4.69
14	$\mathbf{r}_{(5)}$	fc	yes	yes	yes	yes	4.63
15	$\mathbf{r}_{(10)}$	linear	no	yes	no	no	7.88
16	$\mathbf{r}_{(10)}$	linear	yes	yes	yes	no	5.53
17	$\mathbf{r}_{(10)}$	fc	yes	yes	yes	yes	4.55
19	$\mathbf{r}_{(10)}$	cnn	yes	yes	yes	yes	4.42
20	$\mathbf{r}_{(10)}$	rnn	yes	yes	yes	no	4.37
21	$\mathbf{r}_{(10)}$	rnn	yes	yes	yes	yes	4.36
22	$\tilde{\mathbf{r}}_{(10)}$	rnn	yes	yes	yes	yes	4.52

Table 3. Single-voice results. We define $\mathbf{r}_{(m)} \equiv \mathbf{r}_{k-m:k}$ (a truncated history of length m); $\tilde{\mathbf{r}}_{(m)}$ is defined likewise, based on the alternate encoding $\tilde{\mathbf{r}}$ discussed in Section 5.1, Representation. The Relative flag indicates the use of a relative-pitch classifier, and the Loc, Pitch, and Embed flags indicate the use of location features, pitch features, and pitch embeddings, discussed in Section 5.2. For additional details of these experiments, see Appendix A.

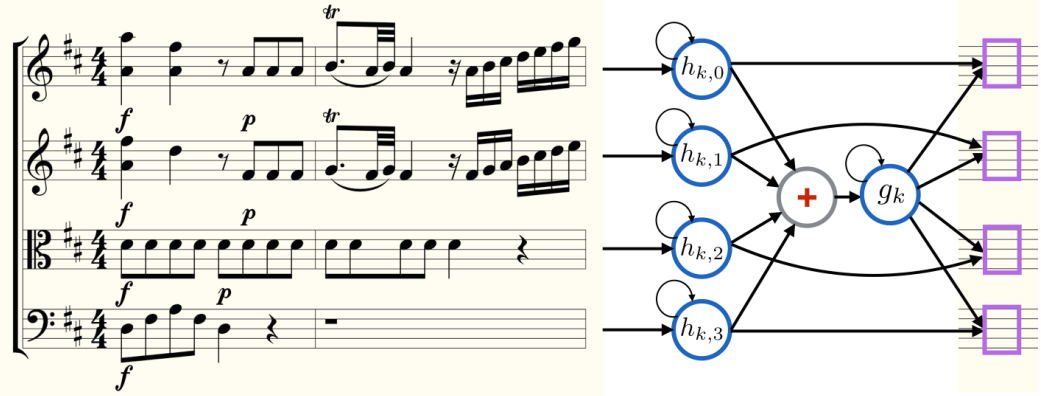


Figure 2. Coupled state estimation of Mozart's string quartet number 2 in D Major, K155, movement 1, from measure 1, rendered by the Verovio Humdrum Viewer. A recurrent network models the state $h_{k,v}$ of each voice v at step k, based on the previous state $h_{k-1,v}$ and the current content of the voice. Another recurrent network models of the global state g_k of the score at step k based on the previous global state g_{k-1} and a sum of the current states of each voice. Subsequent notes (purple) in each voice are predicted using features of the global state and the state of the relevant voice. See Equations 7 for a precise mathematical description of this model.

(section 5.2, Relative pitch). We do not share weights between the note-value and pitch models.

Voice decomposition. Decomposing a score into multiple voices presents us with an opportunity to share weights between voice models by learning a single set of weights W_v in equation (7), rather than learning unique voice-indexed weights W_{v_i} for each voice v_i. Indeed, because voice indices are arbitrary, the weights W_{v_i} will converge to the same values for all i; sharing a single set of weights W_v accelerates learning by enforcing this property. All score models presented in Table 4 share these weights.

#	History (voice/global)	Architecture	Loss (total)	Loss_t (time)	Loss_n (notes)
1	3 / 3	hierarchical	14.05	5.65	8.40
2	5 / 5	hierarchical	13.40	5.35	8.04
6	10 / 10	hierarchical	12.87	5.12	7.75
7	20 / 20	hierarchical	12.78	5.01	7.76
8	10	independent	18.63	6.56	12.08

Table 4. Multi-voice results. The "hierarchical" architecture is defined by equations (7). Voice and global history refer to the number of time steps used to construct the states $h_{k,v}$ and g_k respectively. Experiment 8 is a baseline where the voice models are completely decoupled (equivalent to single-voice Experiment 22 in Table 5; the average number of voices per score is 4.12). Results are reported on non-piano test set data (see Appendix B for discussion of piano data). For additional experiments and ablations, see Appendix A.

6. CONCLUSION

To gain insight into the quality of samples from our models, we recruited twenty study participants to listen to a variety of audio clips, each synthesized from either a real composition or from sampled output of Experiment 6 in Table 4. For each clip, participants were asked to judge whether the clip was written by a computer or by a human composer, following a procedure comparable to [21]. The clips varied in length, from 10 frames of a sample **e** (2-4 seconds; the length of history conditioned on by the model) to 50 frames (10-20 seconds). Participants become more confident in their judgements of the longer clips, but even among the longest clips (around 20 seconds) participants often identified an artificial clip as a human composition. Results are presented in Table 5; see Appendix E for further study details.

Clip Length	10	20	30	40	50
Average	5.3	5.7	6.6	6.7	6.8

Table 5. Qualitative evaluation of the 10-frame hierarchical model: Experiment 6 in Table 4. Twenty participant were asked to judge 50 audio clips each, with lengths varying from 10 to 50 frames. The scores indicate participants' average correct discriminations out of 10: 5.0 would indicate random guessing; 10.0 would indicate perfect discrimination.

These results superficially suggest that we have done well in modeling the short-term structure of the dataset (we make no claims to have captured long-term structure; indeed, the truncated history input to our models precludes this). But it is not clear that humans are good–or should be good–at recognizing plausible local structures in music without context. See [14, 22] for criticism of musical Turing tests like the one presented here. It is also unclear how to use such studies to make fine-grained comparisons between models (as we have done quantitatively throughout this paper). It is not even clear how to prompt a user to discriminate between such models. Therefore we re-emphasize the interpretation of this qualitative evaluation, proposed in Section 1, as a perceptual grounding of the quantitative evaluation considered throughout this work.

7. ACKNOWLEDGEMENTS

We thank Lydia Hamessley and Sreeram Kannan for sharing valuable insights. This work was supported by NSF Grants DGE-1256082, CCF-1740551, the Washington Research Foundation for innovation in Data-intensive Discovery, and the CIFAR program "Learning in Machines and Brains." We also thank NVIDIA for their donation of a GPU.

8. REFERENCES

[1] Moray Allan and Christopher K. I. Williams. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *Advances in Neural Information Processing Systems*, 2006.

[2] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *International Conference on Machine Learning*, 2012.

[3] Darrell Conklin. Music generation from statistical models. In *Proceedings of the AISB 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pages 30–35, 2003.

[4] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Language modeling with longer-term dependency. 2018.

[5] Kemal Ebcioğlu. An expert system for harmonizing four-part chorales. *Computer Music Journal*, 1988.

[6] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 2010.

[7] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. Deepbach: a steerable model for bach chorales generation. *International Conference on Machine Learning*, 2017.

[8] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the maestro dataset. *arXiv preprint arXiv:1810.12247*, 2018.

[9] Dorien Herremans, Ching-Hua Chuan, and Elaine Chew. A functional taxonomy of music generation systems. *ACM Computing Surveys (CSUR)*, 50(5):69, 2017.

[10] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. *International Society for Music Information Retrieval Conference*, 2017.

[11] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. 2019.

[12] Natasha Jaques, Shixiang Gu, Richard E. Turner, and Douglas Eck. Tuning recurrent neural networks with reinforcement learning. *International Conference on Learning Representations Workshop*, 2017.

[13] Daniel D. Johnson. Generating polyphonic music using tied parallel networks. *International Conference on Evolutionary and Biologically Inspired Music and Art*, 2017.

[14] Anna Jordanous. A standardised procedure for evaluating creative systems: Computational creativity evaluation based on what it is to be creative. *Cognitive Computation*, 4(3):246–279, 2012.

[15] Teuvo Kohonen. A self-learning musical grammar, or 'associative memory of the second kind'. *International Joint Conference on Neural Networks*, 1989.

[16] Victor Lavrenko and Jeremy Pickens. Polyphonic music modeling with random fields. *ACM International Conference on Multimedia*, 2003.

[17] Feynman Liang, Mark Gotham, Matthew Johnson, and Jamie Shotton. Automatic stylistic composition of bach chorales with deep lstm. *International Society for Music Information Retrieval Conference*, 2017.

[18] Michael C. Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, 1994.

[19] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[20] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. This time with feeling: Learning expressive musical performance. *arXiv preprint arXiv:1808.03715*, 2018.

[21] Marcus Pearce and Geraint Wiggins. Towards a framework for the evaluation of machine compositions. In *Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pages 22–32, 2001.

[22] Marcus T Pearce and Geraint A Wiggins. Evaluating cognitive models of musical composition. In *Proceedings of the 4th international joint workshop on computational creativity*, pages 73–80. Goldsmiths, University of London, 2007.

[23] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. *arXiv preprint arXiv:1803.05428*, 2018.

[24] Craig Stuart Sapp. Online database of scores in the humdrum file format. *International Society for Music Information Retrieval Conference*, 2005.

[25] Roger N. Shepard. Geometrical approximations to the structure of musical pitch. *Psychological Review*, 1982.

[26] Bob L. Sturm, Joao Felipe Santos, Oded Ben-Tal, and Iryna Korshunova. Music transcription modelling and composition using deep learning. *Conference on Computer Simulation of Musical Creativity*, 2016.

[27] John Thickstun, Zaid Harchaoui, and Sham M. Kakade. Learning features of music from scratch. In *International Conference on Learning Representations (ICLR)*, 2017.

[28] Peter M. Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 1989.

[29] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *SSW*, 125, 2016.

[30] Raunaq Vohra, Kratarth Goel, and J. K. Sahoo. Modeling temporal dependencies in data using a dbn-lstm. *IEEE International Conference Data Science and Advanced Analytics*, 2015.

HIT SONG PREDICTION: LEVERAGING LOW- AND HIGH-LEVEL AUDIO FEATURES

Eva Zangerle, Ramona Huber, Michael Vötter
University of Innsbruck, Austria
firstname.lastname@uibk.ac.at

Yi-Hsuan Yang
Academia Sinica, Taipei, Taiwan
yang@citi.sinica.edu.tw

ABSTRACT

Assessing the potential success of a given song based on its acoustic characteristics is an important task in the music industry. This task has mostly been approached from an internal perspective, utilizing audio descriptors to predict the success of a given song, where either low- or high-level audio features have been utilized separately. In this work, we aim to jointly exploit low- and high-level audio features and model the prediction as a regression task. Particularly, we make use of a wide and deep neural network architecture that allows for jointly exploiting low- and high-level features. Furthermore, we enrich the set of features with information about the release year of tracks. We evaluate our approach based on the Million Song Dataset and characterize a song as a hit if it is contained in the Billboard Hot 100 at any point in time. Our findings suggest that the proposed approach is able to outperform baseline approaches as well as approaches utilizing low- or high-level features individually. Furthermore, we find that incorporating the release year as well as features describing the mood and vocals of a song improve prediction results.

1. INTRODUCTION

The task of predicting hit songs aims to infer the potential (commercial) success of a given song, possibly before the release of the song [18]. This is particularly interesting for the music industry as it allows to find potentially successful songs, promising songwriters and composers, to allocate budget for promotion, and to identify key elements that are pivotal for the success of a song. A natural next step would be the automatic generation of musical pieces which actually exhibit these features that have been shown to be crucial for success. To this end, the hit song prediction task has been tackled from two perspectives [12, 23]: an *internal* perspective, which relies solely on (musical) features extracted from the audio, and an *external* perspective, which models aspects of the musical ecosystem, for instance by incorporating social media or market data.

In this work, we take on the internal perspective, focusing on audio descriptors of a song to predict its success. While this might not capture all the aspects that are relevant for the musical success of songs (e.g., social media trends and events [25], psychological issues [18] or social influence [9, 20]), we believe that it is still an important problem that can be approached on its own and possibly enriched with external information at a later stage.

Approaches focusing on internal factors have mostly modeled hit prediction as a classification or regression problem solved by traditional approaches or, more recently, deep learning [23, 24]. The features used to characterize songs range from low-level Mel-Frequency Cepstral Coefficients (MFCC) [6], melodic features [8], temporal features [10], lyrics features [21] to high-level audio features describing e.g., the danceability of songs [7, 17]. While both of these feature types have been individually shown to contribute to hit prediction, they have yet to be exploited jointly for this task.

Recently, Demetriou *et al*. [5] have investigated the most influential features when it comes to users liking or disliking a song in a user study. They have shown that the most significant features of a song are its ability to evoke emotions, vocals of the singer, beat and rhythm and the lyrics. Along these lines, we are particularly interested in investigating whether these features are also influential in the task of hit song prediction. Interiano *et al*. [12] have shown that audio descriptors relevant for the success of a song change over time and that musical fashion is rather short-lived, rendering it hard to exploit past data to predict future trends. Despite approaches to predict the release year of songs based on acoustic features [2] and the use of temporally weighted regression methods to account for changing features over time [7], this fact has not yet been explicitly explored for hit song prediction. Incorporating release year information into our hit song prediction approach to reflect the dynamics of success on the music market is another distinguishing feature of this work.

Consequently, we shed light on the following two research questions (RQ) in this study:

RQ1: How can we predict hit songs based on acoustic features extracted from the song's audio in a deep learning scenario?

RQ2: Which role do individual features (or groups of features) and the release year of a song play in this task?

To answer these research questions, we model the prediction of hit songs as a regression task. We extract low-

and high-level features from the audio of each track and feed these into a deep neural network architecture, where low-level features are fed into the deep part of the network to distill dense representations thereof, whereas high-level features are fed into the wide part of the network to be utilized directly. This also allows feeding the release year of a song into the network as a high-level feature. Utilizing the dense computed dense representations of low-level features in combination with high-level features, we subsequently compute a regression task to predict a track's peak ranking position.

The contribution of this paper lies in the following aspects: (i) we present a novel regression approach towards hit song prediction using neural networks which combines wide (high-level) and deep (low-level) acoustic features; (ii) we show that mood and vocals (the features identified as being crucial when it comes to liking and disliking a song [5]) are also of high relevance for the hit prediction task; (iii) we show that adding the release year as a high-level feature allows for further improvements, implying that contextualizing the song temporally is important due to the short-lived trends in music [12]; (iv) this is the first work that utilizes solely data from the public domain for this task. For reproducibility and to encourage follow-up research along this line, we also make public the data underlying our experiments [1] .

The remainder of this paper is structured as follows. Section 2 provides information about the dataset underlying our analyses. Section 3 details our proposed approach towards the prediction of hit songs. Section 4 presents the experiments we conducted and the results obtained. Section 6 concludes this paper and discusses future work.

2. DATASET

We base our experiments on the widely used and freely available Million Song Dataset (MSD) [2], which contains one million songs that are representative for western commercial music released between 1922 and 2011. The dataset contains release year information for 515,576 of the MSD songs [2]. As we are interested in the impact of the release year information on hit song prediction quality, we constrain our dataset to those songs that we can obtain the release year information for. In contrast to previous studies on hit song prediction, our dataset fully stems from the public domain. Please refer to Table 1 for an overview of the datasets utilized in existing work and their availability.

To extract low- and high-level audio features for every song, we rely on representative 30 seconds samples for each of the songs in the Million Song Dataset. We make use of the Essentia framework [3] to extract low- and high-level features from the audio (cf. Section 3.1 for details) and dropped all songs in the MSD where we could not determine all those features.

Moreover, our approach requires distinguishing between hits and non-hits of musical success [15]. Along the lines of previous research [13, 21], we define a song as successful if it is featured in the weekly Billboard Hot 100 [2] at least once. Therefore, we crawl the Billboard Hot 100 from the according website for the years 1954 until 2018. To find songs in the Billboard Hot 100 matching the songs contained in the Million Song Dataset, we compare both the artist name and track title for each song pair in the two sets and only consider exact matches as hit songs. After that, we dropped duplicates (determined based on artist name and track title). This provides us with a set of 5,832 hit songs and hence, positive samples for which we extract their highest rank in the charts. For negative samples (and hence, non-hits) sampled from the MSD, it is important to ensure that they are not accidentally hits. Hence, we used to following procedure based on the set of songs for which we have release year information and Essentia features. Firstly, we compute a fuzzy matching ratio [3] between all MSD songs and the set of hit songs by concatenating the artist name and track title with a delimiter and selecting the best matching pairs thereof. Based on this matching procedure, we gather a pool of non-hits where the title fuzzy matching ratio is less or equal than 40. We determined this threshold by preliminary experiments and manually inspecting results. We only consider the title ratio here as it is possible that an artist has multiple further songs, that we nevertheless aim to include in our set of possible non-hits and hence, we do not include artist similarity in this computation. The resulting dataset contains a substantially higher number of non-hits (89,235) than hits (5,832), hence it is highly imbalanced (6.1% positive vs. 93.9% negative instances). To overcome this imbalance, we decided to randomly draw 5,832 samples from the pool of non-hits to get a balanced dataset for our experiments.

Paper	Data	PD	AV
[23, 24]	KKBOX listening data, audio	no	no
[6]	in-house audio database, UK, US, AUS charts	no	no
[8]	in-house audio database, UK charts	no	no
[21]	lyrics features, Billboard charts	no	no
[7, 17]	Echonest features, UK charts	yes	no
[18]	HiFind database of music	no	no
this	MSD dataset, Essentia features	yes	yes

Table 1. Datasets utilized for internal hit song prediction. Notation: PD—dataset stems from public domain, AV—dataset is publicly available.

3. HIT SONG PREDICTION

In this section, we detail our approach on hit-song prediction. We first present the features utilized to characterize songs and then detail the neural network-based approach.

3.1 Song Features

Previous research in the field of hit song prediction has relied on utilizing either low- or high-level features of songs.

[1] https://doi.org/10.5281/zenodo.3258042

[2] https://www.billboard.com/charts/hot-100

[3] The ratio of matching tokens between the two strings is based on Levenshtein distance as implemented by Python's fuzzywuzzy library.

Category	Features
mood	acoustic, aggressive, electronic, happy, party, relaxed, sad [14]; Hu and Downie's 5 clusters of mood [11]
genre	blues, classic, country, disco, hip-hop, jazz, metal, pop, reggae, rock [22]
voice	voice, instrumental, female voice, male voice
rhythm/beat	bpm, beats count, bpm histogram, beats loudness, beats loudness band ratio, onset rate, danceability
chords	chords strength, chords change rate, chords number rate, chords key, chords scale, harmonic pitch class profile, tuning strength and frequency

Table 2. Feature categories and the Essentia features each category contains.

Low-level features allow capturing acoustic descriptors like loudness, dynamics, and spectral shape of a signal, rhythm descriptors or tonal information [19]. In contrast, high-level features are computed from low-level feature models and capture abstract concepts such as mood, genres, vocals or music type [19]. In this work, we aim to combine those two types of features. The intuition here is that while low-level features allow for a detailed description of the acoustic characteristics of a song, high-level features complement this detailed view with abstract concepts such as mood or danceability, resulting in a more holistic description of a song.

Based on the dataset presented in Section 2, we propose to extract low- and high-level features based on a given MP3 file of a song containing a 30 seconds preview. Particularly, we make use of the Essentia toolkit [19], a well-established and widely used extraction library for audio descriptors. For the extraction of low-level features, we rely on Essentia's pre-compiled extractors [4], which provide a variety of spectral, time-domain, rhythm, and tonal descriptors. This provides us with 40 basic features (e.g., MFCCs, dissonance or silence rate), 11 rhythm features (e.g., beats per minute or onset-rate) and 13 tonal features (e.g., key or harmonic pitch class profiles) that serve as low-level input for our task.

For high-level features, we again rely on Essentia and utilize the provided pre-trained high-level classification models [5] to compute high-level features based on the low-level features previously extracted. These features include musical genre, mood, timbre, vocals/voice, or danceability.

In this work, we hypothesize that features identified as salient in users liking/disliking a song [5] are also relevant for the task of hit song prediction. To assess the relative importance of these different features, we rely on the categories of features proposed by Demetriou *et al*. [5] and perform a matching between Demetriou's categories of features and our dataset's features. As our approach is based on internal features only, we are not able to match all of Demetriou's categories (e.g., lyrics). We argue that this is still a valid approach as this work is focused on internal aspects of a song. We hence make use of the following feature categories: mood, genre, voice, rhythm/beat, and chords. The first three contain solely high-level features computed by Essentia, whereas the latter two stem from both Essentia's low- and high-level features. Table 2 shows the assignment of individual low- and high-level features to those categories. As previous research has shown that musical fashion and trends are highly dynamic and short-lived [12], we are also interested in the impact of information about the release year of a song. The idea here is that providing temporal context in terms of the release year of a song can contribute to improved prediction performance as the characterization and embedding of the song is improved. Hence, we extract the release year information for each song from the Million Song Dataset and treat it as a high-level feature.

3.2 Regression Wide and Deep Network

The core idea of our approach is to combine low- and high-level acoustic features to characterize tracks as those two types of features capture different aspects and characteristics of a track (on different levels of abstraction). Given the differences between these two feature types in terms of the amount of features, complexity, and diversity, we aim to reflect this in the architecture of the neural network used for hit prediction. Therefore, we utilize a network architecture inspired by the structural concept of the Wide and Deep network architecture by Cheng *et al*. [4]. While our proposed solution is in fact quite different from the original model [6], we believe that the distinction and notion of deep and wide features describes our scenario well. Hence, we will nevertheless use this notion of wide and deep features and the corresponding network parts.

Figure 1 presents an overview of the proposed network architecture. This architecture allows training a wide linear model alongside a deep neural network while distinguishing two types of features: wide features can be regarded as abstract, high-level features that can directly be used for further computation, whereas deep features in the deep part of the network are used to learn dense, lower-dimensional representations of input features. In our scenario, low-level features can be considered deep features, whereas high-level features are wide features. Based on the wide features and the computed dense representations of the deep features, we aim to perform a regression task for predicting the peak position a song will reach in the charts. This can also be used to distinguish hits and non-hits by using any position larger than 100 as a threshold value. As for the implementation of the deep part of the network, the goal here

[4] http://essentia.upf.edu/documentation/extractors_out_of_box.html, music 1.0 extractor of Essentia v.2.1.-beta2 was used.

[5] http://essentia.upf.edu/documentation/streaming_extractor_music.html

[6] The original wide and deep approach was designed for a recommendation scenario, where the wide part is used to model user-item co-occurrences and the deep part is used to learn low-dimensional latent descriptors of queries and items.

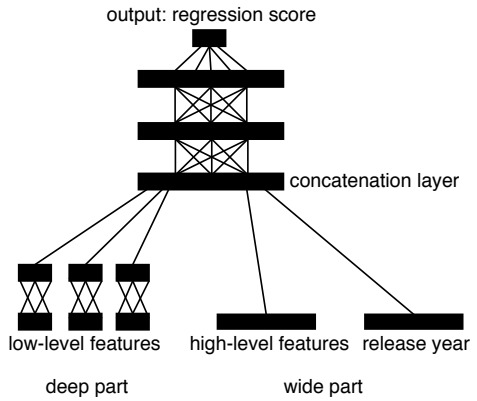


Figure 1. The wide and deep network architecture employed for hit song prediction.

is to use the sparse low-level features as input and to compute meaningful dense representations of audio descriptors to be processed further.

Low-level features comprise a variety of different feature formats: e.g., aggregations of frame-based features across the song or per-frame values for a set of frequency bands—rendering a sparse, complex set of feature representations, where also the computed individual values stem from a broad range. Practically speaking, our approach has to be able to cope with nested input features of varying size, complexity and value ranges. We chose to flatten these input arrays into one-dimensional arrays that can be fed into the deep part of the network.

The purpose of the deep part of the network is to aggregate feature vectors to a one-dimensional representation of the original input features, which are subsequently fed into the regression part of the network. We create multiple groups of low-level features corresponding to the feature categories and their components presented in Table 2. Each group is then fed into a single feature aggregation block (FAB) in the deep part of the network. We chose to model each FAB as a dense layer utilizing a sigmoid activation function as this has been shown to be effective for feature selection in deep neural networks [16]. The input size is chosen to fit the number of values in the feature group and the output size is one. This results in a single value per feature group which is the newly computed higher-level representation of this group. The resulting features computed by the FABs in the deep part of the network are subsequently merged with the features that we feed into the wide part of the network. This concatenation (merge) layer is followed by two dense layers with batch normalization and a ReLU activation function. These two dense layers have the same size as the concatenation layer. To ultimately compute the final result of the regression task, we add another dense layer with output size one and no activation function to ensure that the computed result is in the desired range of possible ranking positions (1–150).

Each high-level feature is represented by classes, where each class is assigned a probability value (range $[0, 1]$). In the special case of two complementing classes such as danceable and not danceable, we chose to only use one of these two probabilities (in the above example, the probability of a song being danceable) to model this feature. We use the resulting values as input for the wide part of the neural network. Feeding categorical values such as the tonal key into the network is realized by previously converting them to a one-hot encoded vector representation. Further, it should be mentioned that we normalized all input (feature) values to the range $[0, 1]$ using a min-max-scaler. Feeding the release year into the neural network is realized by adding another high-level feature (normalized to $[0, 1]$).

4. EXPERIMENTS AND RESULTS

Here, we first present the experimental setup used and secondly, we present and discuss the results obtained.

4.1 Experimental Setup

We base our experiments on the dataset presented in Section 2 and experiment with two different regression tasks: predicting the highest rank of a song in the Billboard Hot 100. For non-hits, we set the highest rank achieved to 150. We chose to use a ranking of 150 to describe non-hits to make the difference between hits and non-hits in terms of ranking more explicit based on preliminary experiments. Due to the high imbalance of hit and non-hit instances in the dataset, we chose to randomly downsample the negative class to achieve balanced classes [7]. Subsequently, we applied five fold cross validation on the remaining instances.

We trained the proposed network architectures with mean squared error (MSE) as loss function and a batch size of 32. The neural network was implemented based on Tensorflow [1], utilizing Keras. As optimizer, we used the adaptive learning rate optimization algorithm, Adam. As for the number of epochs used for training the network, we experimented with values between 10 and 200. All input data is scaled to $[0, 1]$. As we experimented with a wide variety of different setups, training epochs, etc., we utilized a grid search approach to determine the best configuration and present the best obtained results in Section 4.2. Naturally, the underlying network was trained and optimized individually for each input feature set. For the evaluation and comparison of the proposed regression approaches, we use root mean squared error (RMSE) and the mean absolute error (MAE). To also derive a measure of how well these approaches perform when it comes to actually predicting hit songs, we also present the accuracy values for each approach. These were computed based on the results of the regression computation and classifying all tracks with a predicted ranking of less than 100 as hits and a predicted ranking larger than 100 as non-hits. However, we consider this two-class classification an easier task than the regression task based on the actual ranking. Here, we argue that the accuracy evaluation allows us to get an intuition on how

[7] Manual inspection showed that the release year distribution of the test- and training datasets are comparable.

well the regression results may be used to generally distinguish hits from non-hits.

To assess the relative importance of feature classes, we base our evaluation on the following classes, combinations thereof and the combination of individual features stemming from those classes:

- **LL**: basic low-level acoustic features as presented in Section 3.1.
- **LL-filtered**: a subset of the low-level feature set that we have identified as highly relevant in our preliminary feature selection experiments [8]. We argue that pre-selecting a smaller feature set contributes to both runtime and performance (cf. Section 4.2 for results).
- **chords**: chords features as presented in Table 2, extracted from low-level Essentia features.
- **rhythm**: rhythm and beat features as presented in Table 2, extracted from low-level Essentia features.
- **HL**: all high-level acoustic features, information about the release year of the track, including the following sub-categories: **voice**, **mood**, **genre** and **release year** (cf. Table 2 for details on the contained features).

Please note that depending on the feature set utilized, we adapt the way we utilize the neural network accordingly—i.e., for the low-level feature set, we utilize the deep part of the proposed neural network only, whereas, for the high-level feature set, we utilize the wide part of the network only and for any combination of high- and low-level features, we exploit both parts of the full wide and deep network.

We propose to conduct two experiments to answer our research questions: Experiment 1 aims to assess the performance of the proposed wide and deep network architecture. Therefore, we utilize the proposed low-level features in the deep part of the network and the proposed high-level features in the wide part of the network aiming to show that the proposed architecture achieves superior results than (i) a linear regression baseline as well as (ii) utilizing the two parts of the network individually, relying solely on either low- or high-level features. Based on the results of Experiment 1, Experiment 2 aims to investigate the relative importance of individual feature subsets in the wide and deep neural network. To do so, we experiment with different feature sets (low- and high-level) and compare their prediction performance.

As baselines to compare our approach to, we chose to utilize traditional linear regression [9], which we apply to the same feature sets.

[8] Feature set comprises: dissonance, spectral features (centroid, spread, skewness, kurtosis, flatness db, flux, rolloff, decrease, energy), low energy ratio, avg. loudness, barkbands, erbbands, melbands, MFCCs and HFCs.

[9] We experimented with a number of linear regression algorithms (e.g., ridge, lasso or elastic net regularization), where linear regression obtained the best results.

Approach	RMSE	MAE	Acc.
HL (wide)	57.11	48.50	72.08%
LL-filtered+chords+rhythm (deep)	63.94	54.15	65.50%
LL, chords, rhythm (deep)	60.82	52.09	66.94%
HL+LL-filtered+chords+rhythm (wide + deep)	56.05	45.12	74.23%
HL+LL+chords+rhythm (wide + deep)	**55.45**	**43.84**	**75.04%**
HL (baseline)	58.10	50.38	71.01%
LL-filtered+chords+rhythm (baseline)	223.20	57.68	65.97%
LL+chords+rhythm (baseline)	8.54×10^{9}	7.92×10^{6}	68.47%
HL+LL-filtered+chords+rhythm (baseline)	504.90	52.98	72.56%
HL+LL+chords+rhythm (baseline)	6.4110^{9}	5.95×7^{9}	73.91%

Table 3. Results for highest rank prediction on full feature sets. Both the values of RMSE and MAE are the lower the better; the best results are printed in bold font.

4.2 Results and Discussion

In the following, we discuss the findings of the two experiments conducted.

Experiment 1 aimed to investigate the performance of the wide and deep parts of the network individually but also combined in a full wide and deep architecture. Here, we deliberately include all low- and high-level feature sets proposed. Table 3 depicts the results of this experiment. As can be seen, the proposed wide and deep network approach outperforms the baseline approaches across all evaluation measures. This approach achieves the lowest RMSE and MAE values of 55.45 and 43.84 when relying on all low-level features. Using the filtered set of low-level features reaches an RMSE of 56.05 and an MAE of 45.21. When inspecting the results of the network-based approaches that utilize solely either low- or high-level features (which we also consider as representative baseline methods), we observe that utilizing solely high-level features provides us with reasonable results, suggesting that high-level features indeed capture the abstract characteristics of songs well. In contrast, utilizing only low-level features achieves higher RMSE and MAE values. These observations our initial hypothesis as we find that the combination of low- and high-level features is indeed able to substantially outperform approaches utilizing these feature sets individually. The linear regression baseline approaches in the bottom half of the table achieve the best results when utilizing solely high-level features (MAE of 50.38).

When inspecting the accuracy evaluation, we can observe that the highest accuracy value of 75.04% is achieved by the proposed network approach, again utilizing both high- and low-level features. Interestingly, for the linear regression baselines, while RMSE and MAE values are substantially higher than our proposed approach, we can observe that accuracy values are within a reasonable margin, albeit still lower than the proposed wide and deep approach. We lead this discrepancy back to the fact that we assign non-hits a rank of 150. The observed error measures suggest that the predicted ranks computed by linear regression are very high, leading to such high error margins. This is particularly the case when utilizing the full set of low-level features, holding a substantially higher set of features and hence, posing a more complex regression task. However, given the reasonable accuracy results and

the fact that the regression model indeed seems to capture the distinction between hit- and non-hit songs well (with a wide margin between predicted rankings for hits and non-hits) and hence, can be considered a reasonable baseline. To conclude and to answer RQ1, our experiments show the proposed wide and deep neural network-based approach combining low- and high-level features is a suitable approach towards hit song prediction.

Experiment 2 aimed to analyze the relative importance of individual features and classes thereof. Therefore, we evaluated different feature sets in the proposed wide and deep network. As the LL basic features have shown to outperform the filtered set of basic low-level features, we restrain the results presented here to the full low-level feature set. Please note that due to space constraints, we only list the best performing and informative configurations and their obtained results.

For low-level features (including chords and rhythm features), we hardly find differences in their performance (across all combinations with high-level features). Differences in RMSE and MAE between different feature variations are very subtle and do not show a clear pattern regarding best performing features. We conclude that neither chords nor rhythm features are particularly pivotal for hit song prediction. Hence, in the following, we restrain the presented results to the full set of low-level features (LL-filtered, chords, rhythm). Table 4 depicts the results of these analyses.

For high-level features, we can observe that year information profoundly contributes to the prediction performance, improving every experiment by 12–13%, when added to set of high-level features. This confirms our hypothesis that due to short-lived fashion and trends in the music industry, embedding songs in their temporal context by adding release year information allows modeling these dynamics efficiently for hit song prediction. Furthermore, we can observe that—along the lines of Demetriou *et al*. [5]—voice, mood, and genre features are also important for this task. Our experiments show that the combination of high-level features improves RMSE and MAE; the best results are obtained when utilizing low-level, rhythm, and chords features in combination with release year, voice, mood and genre features (hence, the full feature set). Inspecting the performance of single HL features (such as e.g., mood) in combination with low-level features shows that year has the highest impact on the evaluation measures, with genre, mood and voice leading to higher error measures. Combining those high-level features, however, allows to substantially increase performance in all evaluated measures. While the differences between these different feature sets are partly subtle, the patterns detected are stable across all our experiments. To answer RQ2, we find that the release year information is the most important high-level feature. Our experiments also show that voice and mood descriptors contribute to the hit prediction task, which is in line with previous findings regarding salient features in regards to whether people like or dislike a song.

Features LL	Features HL	RMSE	MAE	Acc.
LL, rhythm, chords	year, voice, mood, genre	**55.45**	**43.84**	**75.04%**
LL, rhythm, chords	year, genre	55.93	45.80	73.84%
LL, rhythm, chords	year, mood	57.12	45.66	73.55%
LL, rhythm, chords	year, voice	56.63	46.04	72.04%
LL, rhythm, chords	genre	64.14	52.84	65.11%
LL, rhythm, chords	mood	61.77	52.82	67.92%
LL, rhythm, chords	voice	61.18	52.50	68.00%
LL, rhythm, chords	year	57.51	46.35	72.29%
LL, rhythm, chords	year, mood, voice	56.22	45.53	74.46%
LL, rhythm, chords	year, genre, mood	57.35	45.38	73.63%
LL, rhythm, chords	year, genre, voice	56.06	45.66	73.60%

Table 4. Results for highest rank prediction on feature sets (best results are printed in bold font).

5. LIMITATIONS

We acknowledge that our dataset and our definition of a successful song are biased towards western, commercial music. While we believe that this approach is legitimate, it remains to be shown that our approach can be extended to other types of music and possibly other characterizations of success. However, we believe that due to using audio features, the approach taken is generalizable. Another limitation is the prevalent problem of class imbalance among hits and non-hits as the current setting does not reflect the real distribution of classes. We aim to experiment with unbalanced distributions between hits and non-hits as part of our future work to perform the evaluation in scenario that captures the real-world distribution better. Furthermore, our approach takes an internal perspective based on the audio signal to characterize the track and to predict its success. Here, we have to acknowledge that this model naturally does not include any external factors such as information about the artist (e.g., whether he/she has been on the charts before), marketing strategies of music labels or the relation with special events (e.g., songs being played at Super Bowl).

6. CONCLUSION

In this paper, we have presented a novel approach for the task of hit song prediction. Particularly, we propose to combine low- and high-level audio features of songs in a deep neural network that distinguishes low- and high-level features to account for their particularities. Our experiments on the Million Song Dataset suggest that the combination of these two types of features in the proposed network architecture can indeed improve the prediction performance. Furthermore, we find that incorporating the release year of songs into the wide part of the network allows for temporally contextualizing songs and hence, reflecting musical trends and fashions. In addition, we can show that mood and voice are salient features for this task. Future work includes experimenting with more complex network architectures to allow for improved feature selection and the computation of latent features within the network as well as analyzing and utilizing those features that distinguish hits from non-hits.

7. REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *Proc. USENIX Symposium on Operating Systems Design and Implementation*, volume 16, pages 265–283, 2016.

[2] Thierry Bertin-Mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proc. International Society for Music Information Retrieval Conference*, 2011.

[3] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez Gutiérrez, Sankalp Gulati, Perfecto Herrera Boyer, Oscar Mayor, Gerard Roma Trepat, Justin Salamon, José Ricardo Zapata González, and Xavier Serra. Essentia: An audio analysis library for music information retrieval. In *Proc. International Society for Music Information Retrieval Conference*, 2013.

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proc. ACM Workshop on Deep Learning for Recommender Systems*, pages 7–10, 2016.

[5] Andrew Demetriou, Andreas Jansson, Aparna Kumar, and Rachel Bittner. Vocals in music matter: The relevance of vocals in the minds of listeners. In *Proc. International Society for Music Information Retrieval Conference*, 2018.

[6] Ruth Dhanaraj and Beth Logan. Automatic prediction of hit songs. In *Proc. International Society for Music Information Retrieval Conference*, 2005.

[7] Jianyu Fan and Michael Casey. Study of chinese and uk hit songs prediction. In *Proc. International Symposium on Computer Music Multidisciplinary Research*, pages 640–652, 2013.

[8] Klaus Frieler, K Jakubowski, and Daniel Müllensiefen. Is it the song and not the singer? Hit song prediction using structural features of melodies. *Yearbook of Music Psychology*, pages 41–54, 2015.

[9] Dorien Herremans and Tom Bergmans. Hit song prediction based on early adopter data and audio features. In *Late Breaking Demo at International Society for Music Information Retrieval Conference*, 2017.

[10] Dorien Herremans, David Martens, and Kenneth Sörensen. Dance hit song prediction. *Journal of New Music Research*, 43(3):291–302, 2014.

[11] Xiao Hu and J Stephen Downie. Exploring mood metadata: Relationships with genre, artist and usage metadata. In *Proc. International Society for Music Information Retrieval Conference*, pages 67–72, 2007.

[12] Myra Interiano, Kamyar Kazemi, Lijia Wang, Jienian Yang, Zhaoxia Yu, and Natalia L. Komarova. Musical trends and predictability of success in contemporary songs in and out of the top charts. *Royal Society Open Science*, 5(5), 2018.

[13] Yekyung Kim, Bongwon Suh, and Kyogu Lee. #nowplaying the future billboard: Mining music listening behaviors of twitter users for hit song prediction. In *Proc. International Workshop on Social Media Retrieval and Analysis*, pages 51–56. ACM, 2014.

[14] Cyril Laurier, Owen Meyers, Joan Serra, Martin Blech, and Perfecto Herrera. Music mood annotator design and integration. In *Proc. IEEE Workshop on Content-Based Multimedia Indexing*, pages 156–161, 2009.

[15] J. Lee and J. Lee. Music popularity: Metrics, characteristics, and audio-based prediction. *IEEE Transactions on Multimedia*, 20(11):3173–3182, 2018.

[16] Yifeng Li, Chih-Yu Chen, and Wyeth W Wasserman. Deep feature selection: theory and application to identify enhancers and promoters. *Journal of Computational Biology*, 23(5):322–336, 2016.

[17] Yizhao Ni, Raul Santos-Rodriguez, Matt Mcvicar, and Tijl De Bie. Hit song science once again a science? In *Proc. International Workshop on Machine Learning and Music*, pages 2–3, 2011.

[18] François Pachet. Hit Song Science. In *Music Data Mining*, pages 305–326. Chapman & Hall/CRC Press Boca Raton, FL, 1st edition, 2012.

[19] Alastair Porter, Dmitry Bogdanov, Robert Kaye, Roman Tsukanov, and Xavier Serra. AcousticBrainz: a community platform for gathering music information obtained from audio. In *Proc. International Society for Music Information Retrieval Conference*, 2015.

[20] Matthew J Salganik, Peter Sheridan Dodds, and Duncan J Watts. Experimental study of inequality and unpredictability in an artificial cultural market. *Science*, 311(5762):854–856, 2006.

[21] Abhishek Singhi and Daniel G Brown. Hit song detection using lyric features alone. *Proc. International Society for Music Information Retrieval Conference*, 2014.

[22] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.

[23] Li-Chia Yang, Szu-Yu Chou, Jen-Yu Liu, Yi-Hsuan Yang, and Yi-An Chen. Revisiting the problem of audio-based hit song prediction using convolutional neural networks. In *Proc. IEEE International Conference Acoustics, Speech and Signal Processing*, pages 621–625, 2017.

[24] Lang-Chi Yu, Yi-Hsuan Yang, Yun-Ning Hung, and Yi-An Chen. Hit song prediction for pop music by siamese cnn with ranking loss. *arXiv preprint arXiv:1710.10814*, 2017.

[25] Eva Zangerle, Martin Pichl, Benedikt Hupfauf, and Günther Specht. Can microblogs predict music charts? an analysis of the relationship between #nowplaying tweets and music charts. In *Proc. International Society for Music Information Retrieval Conference*, pages 365–371, 2016.

DA-TACOS: A DATASET FOR COVER SONG IDENTIFICATION AND UNDERSTANDING

Furkan Yesiler[1] **Chris Tralie**[2] **Albin Correya**[1] **Diego F. Silva**[3]
Philip Tovstogan[1] **Emilia Gómez**[1 4] **Xavier Serra**[1]
[1] Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
[2] Department of Mathematics And Computer Science, Ursinus College, USA
[3] Departamento de Computação, Universidade Federal de São Carlos, Brazil
[4] Joint Research Centre, European Commission, Seville, Spain
furkan.yesiler@upf.edu, ctralie@alumni.princeton.edu

ABSTRACT

This paper focuses on Cover Song Identification (CSI), an important research challenge in content-based Music Information Retrieval (MIR). Although the task itself is interesting and challenging for both academia and industry scenarios, there are a number of limitations for the advancement of current approaches. We specifically address two of them in the present study. First, the number of publicly available datasets for this task is limited, and there is no publicly available benchmark set that is widely used among researchers for comparative algorithm evaluation. Second, most of the algorithms are not publicly shared and reproducible, limiting the comparison of approaches. To overcome these limitations we propose Da-TACOS, a DaTAset for COver Song Identification and Understanding, and two frameworks for feature extraction and benchmarking to facilitate reproducibility. Da-TACOS contains 25K songs represented by unique editorial metadata plus 9 low- and mid-level features pre-computed with open source libraries, and is divided into two subsets. The Cover Analysis subset contains audio features (e.g. key, tempo) that can serve to study how musical characteristics vary for cover songs. The Benchmark subset contains the set of features that have been frequently used in CSI research, e.g. chroma, MFCC, beat onsets etc. Moreover, we provide initial benchmarking results of a selected number of state-of-the-art CSI algorithms using our dataset, and for reproducibility, we share a GitHub repository containing the feature extraction and benchmarking frameworks.

1. INTRODUCTION

Cover songs play an important role in the history of recorded music. Weinstein [42] argues that cover songs are peculiar to rock music, and some iconic early rock bands, like The Beatles, The Rolling Stones and Led Zeppelin, recorded cover songs at the beginning of their careers. Artists from other genres eventually followed this trend of reinterpreting recorded musical works. More recently, audio and video online streaming platforms have given rise to a great volume of fan versions of numerous original songs, including so-called "Youtube covers". Cataloguing and tracking cover versions of songs are important both from a historical and a legal standpoint, since there is sometimes a fine line between creative license and plagiarism [22]. However, this task often requires automation via content-based MIR strategies due to the explosion of recordings across many repositories.

Automatic Cover Song Identification (CSI) systems must contend with the myriad changes of musical facets that can occur among versions. While cover songs may share some musical characteristics, such as melody, harmony or chord progression, they are not identical musical works. According to Serrà [28], one can categorize musical transformations between cover versions into 8 main groups: timbre (due to production techniques and/or due to instrumentation), tempo, timing, structure, key, harmonization, lyrics and noise. Given this, the vast majority of CSI systems focus solely on the tonal content [3, 8, 31, 33, 35], a characteristic thought to be least altered between a song and its cover versions. Such systems work on top of features which are invariant to these transformations, incorporating techniques such as beat-synchronous features [12] to control for changes in tempo, or Optimal Transposition Index (OTI) [29] to control for changes in key.

In spite of Serrà's taxonomy and intuition about what makes a cover, to our knowledge, there are no large-scale studies quantifying the extent to which the aforementioned musical attributes change among cover versions. Furthermore, a variety of CSI algorithms have been designed under different assumptions about what makes a cover, each with different goals and trade-offs in mind, but the community lacks a large-scale open source dataset to compare their performance; the largest benchmark set to date is the SecondHandSongs dataset (SHS), a subset of the Million

Song Dataset [4], with 18,196 songs but this contains proprietary features. Motivated by both of these problems, we propose a new dataset, which we call "Da-TACOS"; a DaTAset for COver Song Identification and Understanding (Section 3). This dataset consists of 25,000 songs with a variety of audio descriptors, including low level features such as frame level HPCPs/MFCCs, beat onsets, and higher level information such as key, tempo, and audio tags. We then split our collection of songs into two subsets. The *Cover Analysis subset* is used to quantify what makes a cover by looking at changes in key, local onset deviation, tempo, rhythm, and audio tag descriptions (Section 4). In our analyses, we also introduce some tools not previously seen in the MIR community, such as ShapeDNA [25] and topological time series analysis [26]. The *Benchmark subset* is used for detailed benchmarking of a set of representative CSI algorithms selected across years of work on this topic (Section 5). After we set the stage with our preliminary experiments, we expect this dataset will enable researchers to continue to explore both the *what is a cover* question and *benchmarking* in more detail.

2. RELATED WORK

Most CSI systems have 3 main building blocks [23]: feature extraction, feature post-processing, and similarity estimation. An extensive review of the traditional cover song identification systems can be found in Serrà et al. [30]. In this section, we present a brief overview of these building blocks, the datasets for this task, and the observed limitations of current approaches.

Traditional CSI systems begin with low level feature extraction from the audio. The most common audio descriptors used in those systems are Pitch Class Profiles (PCP), or Chroma features, which represent the tonal content of songs via the octave-folded energies for each of the 12 pitch classes used in Western music theory. The task at hand informs which strategy is chosen, and to improve robustness, different variants of Chroma features [31, 34] were used in the CSI literature for this task. Moreover, many audio descriptors such as pitch salience [27], chord profiles [16], self-similarity MFCC [39], cognition-inspired descriptors [2] were also utilized; however, they suffer from lower performance scores in isolation compared to PCPs.

After the feature extraction, several feature post-processing steps can be applied to achieve invariances in several musical facets such as key, tempo and structure. Key invariance can be obtained using OTI [29] or the 2D Fourier Transform Magnitude (2DFTM) coefficients of the tonal features [14]. Beat-synchronous features are used to achieve tempo invariances [12, 38]. Similarity Matrix Profile (SiMPle) [35], which is a "representation of the similarity join between subsequences", can be useful to control for structural invariance or to obtain audio thumbnails of songs which can be later used to estimate the similarity of two songs [34].

The final step of this general CSI system framework is the similarity estimation. For certain representations, e.g. 2DFTM, this step may consist of only a simple distance function such as Euclidean or Cosine distances. However, for more accurate smaller scale algorithms, a quadratic alignment algorithm is often used to obtain tempo or structural invariance. Since global alignments between versions don't often exist in practice, CSI researchers put more emphasis on the Local Alignment methods such as Smith-Waterman algorithm [36] that is designed to detect alignments among all possible subsequences by incorporating local constraints. Depending on these constraints, many versions of this algorithm were proposed, e.g. *Qmax* [31] and *Dmax* [8]. The longest alignment is taken as the cover similarity measure/distance generally after normalizing it to the length of the reference track.

A number of previous works also explored combining different features and similarity measures to improve their systems. Salamon et al. [27] combine the distance values obtained with *Qmax* for melody, accompaniment and bass line. Chen et al. [8] use a technique called Similarity Network Fusion (SNF) [41] to integrate the similarity matrices obtained with *Qmax* and *Dmax* for the final similarity estimation. Tralie [38] uses SNF to combine cross-similarity matrices obtained with using HPCP and MFCC features to get a final similarity score.

Over the years, new methods were proposed by the MIR community to solve specific problems of the CSI task; however, due to the O(N^2) complexity of local alignment algorithms, some have focused on alternative algorithms that scale better. With the introduction of the SHS dataset [4], techniques such as audio fingerprinting [3], database pruning strategies [24] and multi-modal approaches [10] were also explored in the literature. But the performance scores of these scalable approaches obtained for SHS were not satisfactory. Thus, a trade-off between efficiency and robustness exists in the CSI task as well as in many other MIR tasks, and the "Holy Grail" algorithm for CSI that is both scalable and robust is still missing.

Although a large amount of previous works exist for CSI task, there are only a few public datasets available for benchmarking. Covers80, released by Ellis [13], contains 80 cliques, or cover groups, with 2 songs per clique. Although small in terms of size, this dataset includes audio files of the songs, which provides an opportunity for developing new features or fine-tuning the existing feature extraction algorithms. The YoutubeCovers dataset [33] contains 50 cliques with 7 songs per clique, and instead of audio files, pre-computed Chroma, CENS and Chroma DCT-Reduced log Pitch (CRP) features are included in this dataset. SHS is a subset of Million Song Dataset, and it contains pre-computed features extracted with EchoNest API [1] for 12960 songs in 4128 cliques for the training subset and 5236 songs in 726 cliques for the test subset [4]. Although comparatively larger in size, SHS comes with features pre-computed with proprietary algorithms which makes it impossible to reproduce or even use other audio descriptors for the CSI task.

Based on the limitations of current CSI systems and dif-

[1] http://the.echonest.com/

ficulty in comparing them, we propose a new dataset, and public frameworks for feature extraction and benchmarking to give CSI research a uniform direction. Our contributions can be summarized as follows:

- The largest benchmark set with 15,000 songs including state-of-the-art audio features for CSI
- The Cover Analysis subset with 10,000 songs for musicological studies
- First large-scale quantitative analysis on modified musical characteristics
- Open Source frameworks for feature extraction and benchmarking specifically created for the CSI task
- Open Source implementations of seven state-of-the-art systems and their initial benchmarking results

3. DA-TACOS: DATASET FOR COVER SONG IDENTIFICATION

For facilitating benchmarking and providing a set of analyses regarding links among cover songs, here, we propose a new dataset for CSI research. Da-TACOS, a DaTAset for COver Song Identification and Understanding, contains commercial or live recordings of 25,000 songs that are distributed into 2 subsets: the *Cover Analysis subset* and the *Benchmark subset* with 10,000 and 15,000 songs, respectively. The song annotations are collected from SecondHandSongs.com [2], and are licenced under Creative Commons BY-NC 3.0 [3]. Metadata for each song includes song title, name of the performer, original song title, name of the original writer and release year.

We have also matched the original metadata with MusicBrainz [4] [37] to obtain the MusicBrainz ID (MBID), length and genre tags. Most songs belong to rock, pop, metal and jazz genres. The average length of songs in the dataset is 3.59 minutes.

Along with the metadata, we share low- and mid-level features pre-computed with open source feature extraction libraries from MIR community; a comprehensive list can be found in Table 1. To increase the reliability of our results and assist future works, we share a common feature extraction framework, with which we obtained the feature values, in our GitHub repository.

In particular, Da-TACOS addresses two needs of the current state of CSI research. First, in Section 2, we mentioned the difficulty of benchmarking CSI systems, and with this dataset, we take a step toward tackling this challenge. We provide a large set of pre-computed features that have been constantly used in CSI research, and we provide initial benchmarking results of a selected number of state-of-the-art CSI systems. Second, to our knowledge, our benchmark subset is the largest dataset to date for comparing the performances of CSI systems. We see this as an opportunity to scale up CSI research to discover methods that are more likely to be used in real world scenarios. We believe that having a large dataset for benchmarking will have a positive effect on the direction of future research for this task.

[2] http://secondhandsongs.com
[3] https://creativecommons.org/licenses/by-nc/3.0/
[4] https://musicbrainz.org

	Benchmark	Cover Analysis	
HPCP	✓	✓	Essentia [7]
MFCC	✓	✓	""
Key	✓	✓	""
CENS	✓	✓	Librosa [21]
Tempogram	✓	✓	""
Beat Onsets	✓	✓	Madmom [5]
Tempo	✓	✓	""
CREMA	✓	✓	CREMA [18]
Auto-tagger		✓	Choi et. al [9]

Table 1. List of features provided in each subsets of Da-TACOS and the related feature extraction libraries used.

3.1 The Cover Analysis subset

Our first subset is dedicated to a series of analyses to understand the changes in musical characteristics when a new version of a song is created. This subset includes 10,000 songs in 5,000 cliques, a pair of cover songs for each clique. Out of all songs, we were able to match 6,821 songs with a MBID. The information regarding feature extraction and results of our analyses can be found in Section 4.

3.2 The Benchmark subset

The second subset of Da-TACOS is designed for benchmarking purposes. This subset includes total of 15,000 songs: 13,000 songs in 1000 cliques with 13 songs each, and 2,000 songs that do not belong to any clique, acting as noise in the data. For this subset 10,027 songs have MBIDs, and an initial benchmark of a selected set of CSI systems can be found in Section 5.

4. WHAT IS A COVER?

In this section, we exploit the features to explore the frequency and intensity of a subset of Serrà's [28] posited changes between cover versions. The analyses below are performed on the Cover Analysis subset of Da-TACOS. While key and tempo are straightforward to compare, we devise custom distance measures to compare timing, structure, and semantic aspects, e.g. instrumentation, genre. In these latter cases, we compare distributions of the corresponding distances between true cover pairs in this subset to all other non-cover pairs in the subset. To quantify the extent to which the true cover and non-cover distributions differ in these cases, we report the 2-sample Kolmogorov-Smirinov (KS) score, with its associated p-value, which indicates the statistical significance of the difference between two distributions.

4.1 Results

4.1.1 Key

Using a key estimation algorithm [15], we considered all pairs with both songs exceeding a confidence of 0.75,

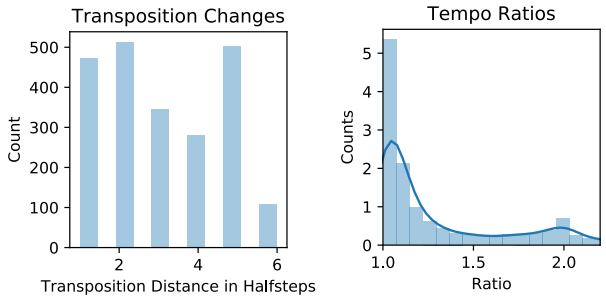


Figure 1. (Left) Distribution of halfsteps between key estimates for cover pairs with a reported key change. (Right) Distribution of tempo ratios between cover pairs.

which was 4288/5000 pairs. Among these, 69.3%, were reportedly in a different key. The distribution of said halfstep shifts is shown in Figure 1. Thus, the use of OTI in many CSI algorithms is justified. One caveat is that the key estimation algorithms report a single estimate which is either major or minor. Under this scheme, the key estimation algorithm reported that 17.5% of the pairs shifted from major to minor. However, upon spot checking, it was clear that many of these examples either switched keys at different times, or they were in modes beyond major and minor. In the absence of more sophisticated algorithms, expert ear trained individuals would be needed to determine how often changes beyond simple transpositions occur.

4.1.2 Tempo

We now examine tempo ratios between cover pairs by picking out the tempo with the maximum confidence from a state-of-the-art tempo estimator [6]. The right of Figure 1 shows the results. There is a slight peak around 2 which is likely due to "octave errors" from pieces which can be subdivided into 4/4. Beyond that, at the first quartile is a 1.03x change in tempo, in the second quartile is a 1.11x change in tempo, and in the third quartile is a 1.53x change in tempo. Thus, half of the songs are quite stable, but in the 50-75% quartile, we have a fair number of songs with a significant tempo change which can't easily be explained by a direct tempo doubling mistake, and which are likely "real." Hence, tempo is often a factor which needs to be controlled for when analyzing cover versions.

4.1.3 Structure

One particularly successful approach to multiscale music structure analysis uses eigenvectors of the Graph Laplacian, or "spectral clustering" [19]. While one can compare agreement of this technique to that of human annotators on the same piece of music [18], this representation does not immediately extend across versions of songs. We instead use the eigen*values* of the Graph Laplacian, which we stack up into a Euclidean vector which can be compared across songs. This has been referred to as "Shape DNA" in the context of 3D shape analysis of triangle meshes [25]. In our case, we use feature fused SSMs [40] downsampled to a common dimension of 256×256, followed by 30 eigenvalues of a random walk Laplacian.

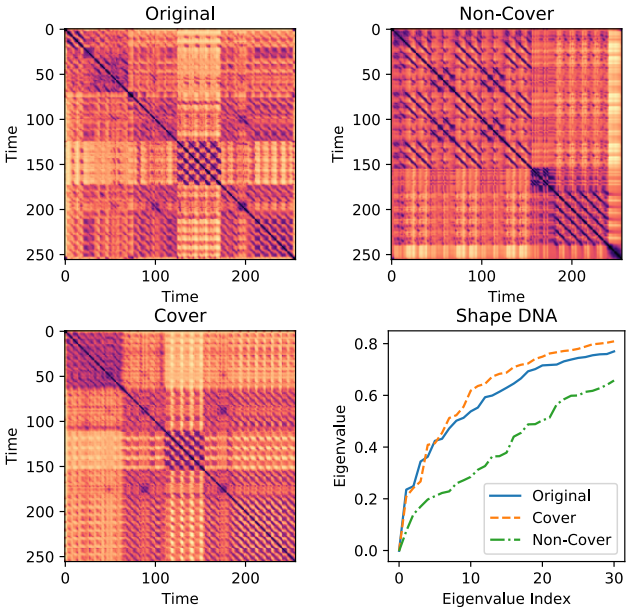


Figure 2. An example of fused similarity matrices of "The Wizard" by Uriah Heep (upper left), a cover by Blind Guardian (lower left), and "Million Pieces" by The Piano Tribute Players (upper right), which is unrelated. The corresponding Shape DNAs are shown in the lower right.

Figure 2 shows an example of Shape DNA between a pair of covers and a third, unrelated song. Even though the cover pairs' similarity matrices do not align perfectly and contain other variations, their shape DNAs are close, while they are both different from an unrelated song with a different structure. Figure 3 shows the distributions of shape DNA differences between true cover and non-cover pairs. The KS score between the two distributions is 0.22 ($p \ll 0.001$), indicating that while large structural changes do occur between cover versions (e.g. added/deleted sections), it is overall more likely for cover songs to share structure than random pairs of songs.

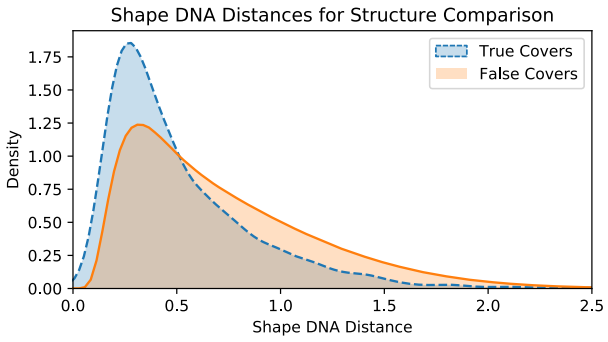


Figure 3. Distributions of shape DNA differences between pairs of songs as a means of assessing structural changes.

4.1.4 Timing

We now turn to timing, which we define as local changes in tempo over time. We first extract N beat onset estima-

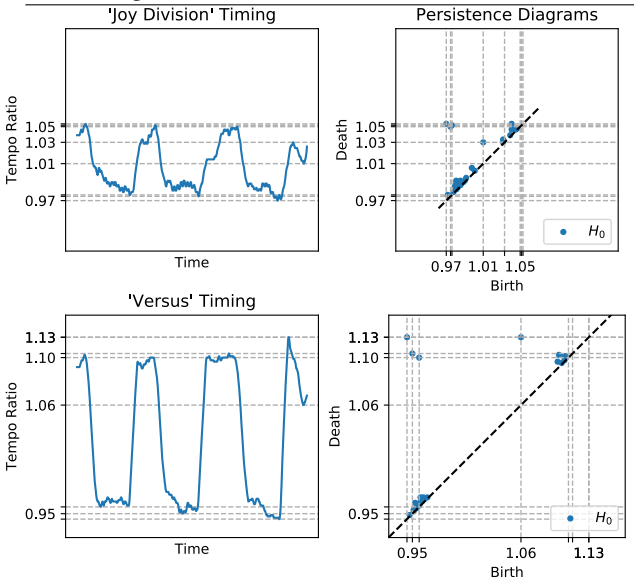


Figure 4. An example of $r[t]$ functions and their associated persistence diagrams for the song "24 Hours" by Joy Division and Versus. Both songs speed up in the chorus and slow down in the verse, so they each contain several local mins with high persistence which are born during the verses. They each also contain some low amplitude wobbling which shows up as dots near the diagonal.

tion times $b[t], t = 1, 2, \ldots N$ using the technique of Krebs [17], down to a resolution of 10 milliseconds. We then extract unit-less local tempo estimates by convolving $b[t]$ with a Gaussian derivative $b'[t] = b[t] * (-te^{-t^2/2})$, followed by a sliding window average of width 20 to smooth out noise. Finally, we divide $b'[t]$ by its median to obtain a relative, tempo-normalized local tempo deviation $r[t]$; $r[t] > 1$ if a song has sped up locally, and $r[t] < 1$ if it has slowed down locally.

The left column of Figure 4 shows $r[t]$ for two different versions of the same song. Note the multi-scale features of $r[t]$, from small wobbles in tempo to large changes that persist over a section. To capture all scales in one distance measure which can tolerate missing beats and added/deleted sections, we turn to the "lower star filtration," a watershed method from topological data analysis [11]. This summarizes a time series in a "persistence diagram"[5] (PD). This has been used, for instance, on speed time series of drivers to quantify driving behavior [26].

The right column of Figure 4 shows PDs for the $r[t]$ functions for an example cover pair, with the birth and death values of the points with 4 largest "persistence" (death-birth) marked. To compare PDs between two different songs, we use persistence images [1], which transform a diagram into birth-persistence space and place a Gaussian over each point, whose magnitude is proportional to the persistence. Figure 5 shows the distributions of Euclidean distances between peristence images for true cover

[5] A multiset of points whose x-coordinates correspond to local mins where pools of water form as water rises from bottom to top ("birth events"), and and whose y-coordinates correspond to local maxes paired to these mins where two pools merge together ("death events")

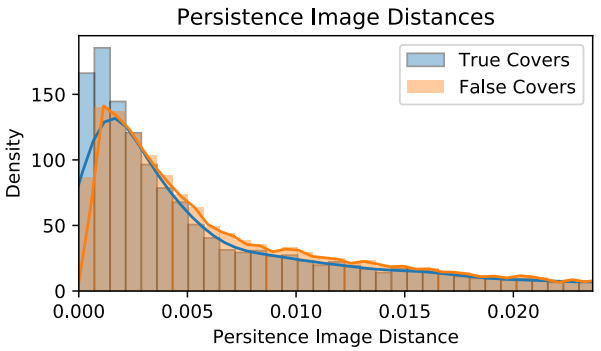


Figure 5. Distribution of persistence image distances of lower star filtrations relative tempo functions between pairs of songs.

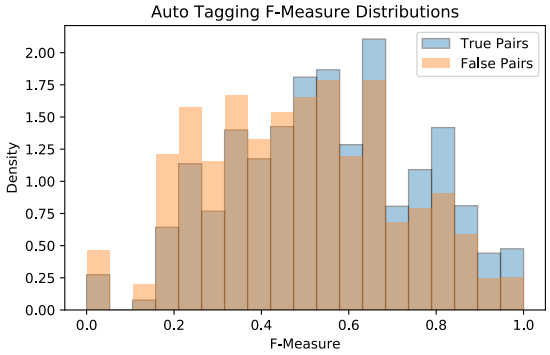


Figure 6. Distributions of f-measures for auto tagging for true cover pairs and non-cover pairs.

pairs and non-cover pairs. Though the distributions are quite similar, the KS score is 0.095 ($p \ll 0.001$), indicating that though relative timing can be different (as evidenced by Figure 4 where one song speeds up more in the chorus relative to the other), the difference is less for covers than for random pairs.

4.1.5 Semantic Aspects

To analyze semantic aspects of the songs such as mood, instrumentation and "genre" without explicitly defining them, we turn to auto tagging techniques of Choi et al. [9] which use log-mel spectrograms as input to return a set of tags which qualitatively describe the songs. Since the auto tagger returns many tags with low confidence, we only take tags which are in the 90$^{\text{th}}$ percentile over all confidences, which is a confidence value of 0.062. If p is the fraction of tags in song A contained in the set of tags for song B, and r is the fraction of tags in song B contained in the tags for song A, then the *f-measure* between two songs is defined as $2pr/(p + r)$, which is 1 if they are in complete agreement and 0 if they have nothing in common. Figure 6 shows the distribution of f-measures between true cover and non-cover pairs. While the distributions are overall quite similar, the f-measures are skewed slightly lower for non covers. The KS score between the two distributions is 0.118 ($p \ll 0.001$), indicating these two distributions are

different by more than chance; thus, we can conclude that although less frequent than between two random songs, stylistic changes occur between cover pairs.

5. BENCHMARKING

As mentioned in Section 3, Da-TACOS contains a benchmark subset of 15,000 songs for comparative algorithm evaluation. In this section, we present the results on seven different state-of-the-art algorithms on this data. To the best of our knowledge, this is the first work comparing these algorithms for CSI on a publicly available, large-scale dataset with features obtained with open source algorithms.

5.1 Methodology

One of the main limitations of current CSI research is the lack of a public framework to compare the performance of different systems. We acknowledge that the audio cover song identification task in the Music Information Retrieval Evaluation eXchange (MIREX) [6] addresses this. However, MIREX data is not publicly available, and it restricts the evaluation to a limited time window per year. According to the results from previous MIREX, [32] is still an algorithm which may be considered in the state-of-the-art CSI system. We chose to benchmark six other unsupervised algorithms which more recently presented good results on CSI for comparison in this competition [8, 14, 31, 35, 38, 39]. In their original implementations, these algorithms differ with the features used, a large scale or small scale design goal, their ability to combine distance measures or fusing more than one feature set [8, 38], exploitation of network structure of songs [8, 38], the application of beat-synchronous features [14, 38, 39], or a combination of these properties. In our work, we have the opportunity to control for implementation details that can greatly impact performance [20] both by sharing features across all algorithms, and by using common implementations of some sub-algorithms, including OTI, Similarity Network Fusion (SNF) [41] (for [38] and [8]), and QMax alignment [31].

5.2 Results

The empirical evaluation of CSI algorithms is another point in which published papers differ greatly. Commonly, different authors use different subsets of evaluation measures. For this reason, we used a large number of evaluation measures assessing the results, namely Mean Rank (MR), Mean Reciprocal Rank (MRR), Median Rank (MDR), Mean Average Precision (MAP), and the counting of correctly identified versions in top 1 and top 10.

In addition to using HPCPs as Chroma features for all the algorithms, we also use CREMA chord model features [18], sampled at the same rate, as a drop-in replacement for Chroma on all algorithms.

Table 2 presents the results obtained by all the algorithms considered in our evaluation, with a simplified version of Tralie's early fusion [38] which uses a weighted average in the early fusion stage instead of SNF for speed.

		MR	MRR	MDR	MAP	Top 1	Top 10
FTM2D [14]	H	207	0.314	15	0.126	3954	6131
	C	155	0.523	1	0.275	7185	9072
Simple [35]	H	358	0.362	13	0.165	4916	6361
	C	142	0.555	1	0.332	7739	9391
Dmax [8]	H	155	0.562	1	0.292	7939	9320
	C	134	0.571	1	0.322	7981	9611
LateFusion [8]	H	210	0.604	1	0.410	8761	9880
	C	177	0.621	1	**0.454**	8897	10223
Qmax [31]	H	119	0.606	1	0.333	8630	9931
	C	113	0.611	1	0.365	8625	10212
Qmax* [32]	C	**104**	0.619	1	0.373	8766	10246
SSM [39]	M	434	0.273	39	0.096	3540	5139
EarlyFusion [38]	H	116	**0.680**	1	0.426	**9843**	**10861**
	C	120	0.672	1	0.416	9667	10829

Table 2. Performance statistics of all algorithms. H stands for HPCP, C for CREMA and M for MFCC.

Overall, we find the CREMA improves results over HPCP, which suggests an adoption of CREMA for future research. We are particularly surprised at how well the large-scale FTM2D algorithm performs with CREMA.

6. CONCLUSION

In this work, we have presented a new public dataset, Da-TACOS, for analyzing how a number of musical facets vary among cover songs and benchmarking CSI systems. Our "what is a cover" analysis takes Serrà's [28] categories of modifiable musical characteristics as a basis, and the results demonstrate large variations between cover pairs across all of the aspects we examined, which supports Serrà's claims on the subject. However, the same analyses among non-cover pairs show a larger variation than cover pairs, and this can be interpreted as there are some links remaining among cover songs.

Moreover, we created a framework that includes open source implementations of seven state-of-the-art unsupervised CSI algorithms to facilitate the future work in this line of research. Using this framework, researchers can easily compare existing algorithms on different datasets, and we encourage all CSI researchers to incorporate their algorithms into this framework in order to support Open Science principles. Our feature extraction and benchmarking frameworks as well as instructions to can be found in our GitHub repository [7].

Our future work includes constructing several other subsets based on various characteristics of songs (e.g. subsets based on genre and release year), as well as training sets for supervised algorithms, to identify the further needs of CSI research. We believe that a better understanding of the relationships among cover songs is valuable both for musicological aspect of this line of research and for advancing the state of the art in CSI research.

[6] https://www.music-ir.org/mirex/wiki

[7] https://github.com/furkanyesiler/acoss

7. ACKNOWLEDGMENTS

This work is partially supported by the MIP-Frontiers project, the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 765068, and by TROMPA, the Horizon 2020 project 770376-2.

8. REFERENCES

[1] Henry Adams, Sofya Chepushtanova, Tegan Emerson, Eric Hanson, Michael Kirby, Francis Motta, Rachel Neville, Chris Peterson, Patrick Shipman, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *The Journal of Machine Learning Research*, 18(1):218–252, 2017.

[2] Jan Van Balen, Dimitrios Bountouridis, Frans Wiering, and Remco Veltkamp. Cognition-inspired descriptors for scalable cover song retrieval. In *15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, Taipei, Taiwan, 2014.

[3] Thierry Bertin-Mahieux and Daniel P.W. Ellis. Large-scale cover song recognition using hashed chroma landmarks. In *2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, New York, USA, 2011.

[4] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In *12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, Miami, Florida, USA, 2011.

[5] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. Madmom: A new Python audio and music signal processing library. In *Proc. of the 24th ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, 2016.

[6] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Accurate tempo estimation based on recurrent neural networks and resonating comb filters. In *16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, pages 625–631, Malaga, Spain, 2015.

[7] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, José Zapata, and Xavier Serra. Essentia: An audio analysis library for music information retrieval. In *14th Conference of the International Society for Music Information Retrieval (ISMIR 2013)*, Curitiba, Brazil, 2013.

[8] Ning Chen, Wei Li, and Haidong Xiao. Fusing similarity functions for cover song identification. *Multimedia Tools and Applications*, 77(2):2629–2652, 2018.

[9] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP))*, pages 2392–2396, New Orleans, Louisiana, USA, 2017.

[10] Albin Correya, Romain Hennequin, and Mickaël Arcos. Large-scale cover song detection in digital music libraries using metadata, lyrics and audio features. *arXiv preprint arXiv:1808.10351*, 2018.

[11] Herbert Edelsbrunner and John Harer. *Computational topology: An introduction*. American Mathematical Soc., 2010.

[12] Daniel P.W. Ellis. Identifying "cover songs" with beat-synchronous chroma features. *MIREX 2006*, pages 1–4, 2006.

[13] Daniel P.W. Ellis. The "covers80" cover song data set. *URL: http://labrosa. ee.columbia. edu/projects/coversongs/covers80*, 2007.

[14] Daniel P.W. Ellis and Thierry Bertin-Mahieux. Large-scale cover song recognition using the 2D Fourier Transform magnitude. In *13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, Porto, Portugal, 2012.

[15] Emilia Gómez. Tonal description of polyphonic audio for music content processing. *INFORMS Journal on Computing*, 18(3):294–304, 2006.

[16] Maksim Khadkevich and Maurizio Omologo. Large-scale cover song identification using chord profiles. In *14th International Society for Music Information Retrieval Conference (ISMIR 2013)*, pages 233–238, Curitiba, Brazil, 2013.

[17] Florian Krebs, Sebastian Böck, and Gerhard Widmer. An efficient state-space model for joint tempo and meter tracking. In *16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, pages 72–78, Malaga, Spain, 2015.

[18] Brian McFee and Juan P. Bello. Structured training for large-vocabulary chord recognition. In *18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, pages 188–194, Suzhou, China, 2017.

[19] Brian McFee and Daniel P.W. Ellis. Analyzing song structure with spectral clustering. In *15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, Taipei, Taiwan, 2014.

[20] Brian McFee, Jong Wook Kim, Mark Cartwright, Justin Salamon, Rachel M. Bittner, and Juan P. Bello. Open-source practices for music signal processing research: Recommendations for transparent, sustainable, and reproducible audio research. *IEEE Signal Processing Magazine*, 36(1):128–137, 2019.

[21] Brian McFee, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in Python. In *Proc. of the 14th Python in Science Conference*, pages 18–25, Austin, Texas, USA, 2015.

[22] Emily Miao and Nicole E. Grimm. The blurred lines of what constitutes copyright infringement of music: Robin Thicke v. Marvin Gaye's estate. *Westlaw Journal Intellectual Property*, 20:1, 2013.

[23] Julien Osmalskyj. *A Combining Approach to Cover Song Identification*. PhD thesis, University of Liege, Belgium, 2017.

[24] Julien Osmalskyj, Sébastien Piérard, Marc Van Droogenbroeck, and Jean-Jacques Embrechts. Efficient database pruning for large-scale cover song recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 714–718, Vancouver, Canada, 2013.

[25] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. Laplace–Beltrami spectra as 'Shape-DNA' of surfaces and solids. *Computer-Aided Design*, 38(4):342–366, 2006.

[26] David Rouse, Adam Watkins, David Porter, John Harer, Paul Bendich, Nate Strawn, Elizabeth Munch, Jonathan DeSena, Jesse Clarke, Jeff Gilbert, Sang Chin, and Andrew Newman. Feature-aided multiple hypothesis tracking using topological and statistical behavior classifiers. In *Signal Processing, Sensor/Information Fusion, and Target Recognition XXIV*, page 94740L, 2015.

[27] Justin Salamon, Joan Serrà, and Emilia Gómez. Melody, bass line, and harmony representations for music version identification. In *Proc. of the 21st Int. World Wide Web Conf. (WWW 2012): 4th Int. Workshop on Advances in Music Information Research (AdMIRe 2012)*, pages 887–894, Lyon, France, 2012.

[28] Joan Serrà. *Identification of Versions of the Same Musical Composition by Processing Audio Descriptions*. PhD thesis, Universitat Pompeu Fabra, Spain, 2011.

[29] Joan Serrà, Emilia Gómez, and Perfecto Herrera. Transposing chroma representations to a common key. In *IEEE CS Conference on The Use of Symbols to Represent Music and Multimedia Objects*, pages 45–48, 2008.

[30] Joan Serrà, Emilia Gómez, and Perfecto Herrera. Audio cover song identification and similarity: background, approaches, evaluation, and beyond. In *Advances in Music Information Retrieval*, pages 307–332. Springer, 2010.

[31] Joan Serrà, Xavier Serra, and Ralph G. Andrzejak. Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11(9):093017, 2009.

[32] Joan Serrà, Massimiliano Zanin, Cyril Laurier, and Mohamed Sordo. Unsupervised detection of cover song sets: Accuracy improvement and original identification. In *10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, pages 225–230, Kobe, Japan, 2009.

[33] Diego Furtado Silva, Vinícius M. A. de Souza, and Gustavo E. A. P. A. Batista. Music shapelets for fast cover song recognition. In *16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, pages 441–447, Malaga, Spain, 2015.

[34] Diego Furtado Silva, Felipe Vieira Falcão, and Nazareno Andrade. Summarizing and comparing music data and its application on cover song identification. In *19th International Society for Music Information Retrieval Conference (ISMIR 2018)*, pages 732–739, Paris, France, 2018.

[35] Diego Furtado Silva, Chin-Chia Michael Yeh, Gustavo E.A.P.A. Batista, and Eamonn J. Keogh. SiMPle: Assessing music similarity using subsequences joins. In *17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, pages 23–29, New York City, New York, USA, 2016.

[36] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

[37] Aaron Swartz. Musicbrainz: A semantic web service. *IEEE Intelligent Systems*, 17(1):76–77, 2002.

[38] Christopher J. Tralie. Early MFCC and HPCP fusion for robust cover song identification. In *18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China, 2017.

[39] Christopher J. Tralie and Paul Bendich. Cover song identification with timbral shape sequences. In *16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, Malaga, Spain, 2015.

[40] Christopher J. Tralie and Brian McFee. Enhanced hierarchical music structure annotations via feature level similarity fusion. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, 2019.

[41] Bo Wang, Aziz M. Mezlini, Feyyaz Demir, Marc Fiume, Zhuowen Tu, Michael Brudno, Benjamin Haibe-Kains, and Anna Goldenberg. Similarity Network Fusion for aggregating data types on a genomic scale. *Nature methods*, 11(3):333, 2014.

[42] Deena Weinstein. The history of rock's pasts through rock covers. In T. Swiss, J. Sloop, and A. Herman, editors, *Mapping the Beat: Popular Music and Contemporary Theory*, pages 137–151. Blackwell Publishing, Malden, MA, USA, 1998.

HARMONIC SYNTAX IN TIME RHYTHM IMPROVES GRAMMATICAL MODELS OF HARMONY

Daniel Harasim[1] **Timothy J. O'Donnell**[2] **Martin Rohrmeier**[1]
[1] Digital and Cognitive Musicology Lab, École Polytechnique Fédérale de Lausanne, Switzerland
[2] Department of Linguistics, McGill University, Canada
daniel.harasim@epfl.ch

ABSTRACT

Music is hierarchically structured, both in how it is perceived by listeners and how it is composed. Such structure can be elegantly captured using probabilistic grammatical models similar to those used to study natural language. They address the complexity of the structure using abstract categories in a recursive formalism. Most existing grammatical models of musical structure focus on one single dimension of music–such as melody, harmony, or rhythm. While these grammar models often work well on short musical excerpts, accurate analysis of longer pieces requires taking into account the constraints from multiple domains of structure. The present paper proposes abstract product grammars–a formalism which integrates multiple dimensions of musical structure into a single grammatical model–along with efficient parsing and inference algorithms for this formalism. We use this model to study the combination of hierarchically-structured harmonic syntax and hierarchically-structured rhythmic information. The latter is modeled by a novel grammar of rhythm that is capable of expressing temporal regularities in musical phrases. It integrates grouping structure and meter. The combined model of harmony and rhythm outperforms both single-dimension models in computational experiments. All models are trained and evaluated on a treebank of hand-annotated Jazz standards.

1. INTRODUCTION

Music is hierarchically organized, which is probably most evident in the structure of harmonic sequences. Grammatical models of music describe both local and non-local relations between musical objects such as notes or chords by assuming a latent hierarchical structure. Originally inspired by Schenkerian analysis and generative linguistics [9], grammatical models have been used in music theory [14, 19, 24, 25], computational musicology [1, 5, 6, 13, 16, 27], music information retrieval [3, 4, 12, 18, 26], and increasingly also music psychology [7, 20]. Consider for example the Jazz chord sequence C^6 D^7 Dm^7 G^7 C^6 of the A-part of the Jazz standard *Take the A-Train*. A hierarchical analysis of this sequence is shown in Figure 1a. The progression D^7 Dm^7 G^7 forms a *dominant phrase* inside the *tonic phrase* C^6 D^7 Dm^7 G^7 C^6, exhibiting a non-local harmonic relationship between the chords D^7 and G^7. The nesting of the phrases moreover illustrates the idea of how pieces can be decomposed into hierarchically-structured constituents (subtrees) which stand in part-whole relationship with one another [6]. Figure 2 displays a typical case of a non-local harmonic relation in Jazz harmony.

To analyze hierarchical harmonic structures, music theorists make use of many additional structural features such as melody, rhythm, voice-leading, and form, for disambiguation. From this perspective, the latent harmonic structure of a piece cannot be fully inferred from sequences of chord symbols alone. Most existing grammatical models of harmony, however, do not take these other domains of musical structure in account. In this paper, we propose a novel formalism that combines models of different musical features. The mathematical idea is similar to Coupled-context-free Grammars [17]. We extend that approach by a probabilistic model and apply the general construction to improve models of harmonic syntax by incorporating harmonic rhythm.

1.1 Problem Setting

Existing grammatical models of harmony typically do not capture how harmonic structure is laid out in time [21], as shown in Figure 1a. This analysis captures information such as the dependencies between different kinds of musical phrase (tonic, dominant, subdominant), ordering, and hierarchical constituency, but contains no information on the duration of chords. This paper extends models of harmonic syntax to include rhythmic structure illustrated in Figure 1b. This figure shows how the musical phrases in Figure 1a are laid out in time by progressively assigning constituents to a metrical grid consisting of eight measures. The inclusion of the metrical domain reveals previously hidden structure. In the first step, the root of the harmonic tree is assigned to the entire eight bars. In the second step, the tonic phrase is split into equal halves which are assigned to bars 1-4 and bars 5-8 of the metrical grid. In the third step, the second half of the piece is split into equal halves, introducing a V in the first part of the split

and limiting the tonic scale degree to the second part. The fourth step, in contrast, splits the first half (measures 1–4) into two and assigns the second half of this split to the second half of the progression (measures 5–8). Measures 3 and 4 are said to be a *harmonic upbeat* to measures 5 and 6. In the following, we present an integrated model of harmony and phrase rhythm [22] that accounts for the structural differences of the steps three and four. Note that we therefore assume the existence of hypermeter, the extension of metrical structures within a single measure to relations between measures [11].

We propose an approach that models the upbeat and the downbeat of harmonic constituents separately. Figure 1c shows a hierarchical analysis integrating harmonic syntax and harmonic rhythm. In this notation, the durations of upbeats are separated from the durations of downbeats by the symbol $\oplus$. The symbol $\ominus$ is used to indicate the "time stealing" from generation step 3 in Figure 1b.

2. GRAMMATICAL MODELS

2.1 Abstract Context-Free Grammars

The following two definitions are adopted from [6], where further explanation and examples can be found.

A *(non-probabilistic) Abstract Context-free Grammar* $G = (T, C, C_0, \Gamma)$ consists of a set T of *terminal symbols*, a set C of *constituent categories*, a set of *start categories* $C_0 \subseteq C$, and a set of partial functions

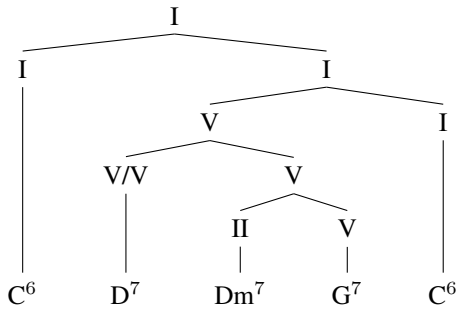
$$\Gamma := \{\, r \mid r : C \nrightarrow (T \cup C)^* \,\}, \tag{1}$$

called *rewrite rules* or *rewrite functions*. The arrow $\nrightarrow$ is used throughout the paper to denote partial functions and $\mathrm{dom}(r)$ denotes the set of arguments for which a partial function r is defined. A sequence $\beta \in (T \cup C)^*$ can be generated from a sequence $\alpha \in (T \cup C)^*$ by one *rule application* of a rewrite function $r \in \Gamma$, denoted by $\alpha \longrightarrow_r \beta$, if there exist $\alpha_1, \alpha_2 \in (T \cup C)^*$ and $A \in C$ such that $\alpha = \alpha_1 A \alpha_2$ and $\beta = \alpha_1 r(A) \alpha_2$. A sequence of rewrite rules $r_1 \dots r_n$ is called a *derivation* of a sequence of terminals $\alpha \in T^*$ if there exists a start category $\alpha_1 \in C_0$, and $\alpha_2, \dots, \alpha_n \in (C \cup T)^*$ such that

$$\alpha_1 \longrightarrow_{r_1} \alpha_2 \longrightarrow_{r_2} \cdots \longrightarrow_{r_n} \alpha, \tag{2}$$

where r_i is always applied to the leftmost category of α_i for $i \in \{1, \dots, n-1\}$. The set of derivations of α is denoted by $D(\alpha)$. The language of the grammar G is the set of terminal sequences that have a derivation in G.

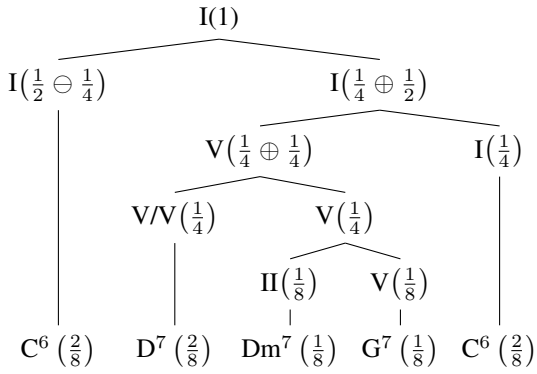
A *Probabilistic Abstract Context-free Grammar* is an Abstract Context-free Grammar where each category $A \in C$ is associated with a random variable X_A over rewrite functions r such that the probability $\mathbb{P}(X_A = r)$ is positive if and only if $r(A)$ is defined, that is $A \in \mathrm{dom}(r)$. In the following, we also use the notation $p(r \mid A) = \mathbb{P}(X_A = r)$ and $p(A \longrightarrow_r \alpha) = \mathbb{P}(X_A = r)\ \mathbb{1}(r(A) = \alpha)$. The probability $p(d)$ of a derivation $d = r_1 \dots r_n$ of a sequence of terminal symbols $\alpha \in T^*$ is defined as the product $\prod_{i=1}^{n} \mathbb{P}(r_i \mid A_i)$ where in each step r_i is applied to a category $A_i \in C$. The probability of α is then defined as $p(\alpha) = \sum_{d \in D(\alpha)} p(d)$.



(a) Generative syntax tree of the harmonic structure. The leafs of the tree are the chord symbols of the A-part. The internal nodes show scale degrees with respect to C major as latent categories. Subtrees form harmonic constituents. The nested structure of the subtrees shows how complex constituents are build from simpler constituents [6].

1:	I							
2:	I				I			
3:	I				V		I	
4:	I		V/V		V		I	
5:	I		V/V		II	V	I	
6:	C^6		D^7		Dm^7	G^7	C^6	

(b) Schematic generation of the chord sequence including their metrical positions. Each row consists of 8 measures and shows one step in the generation process. Chords are tied over following "empty" measures. The third and the fourth step show the two basic kinds of harmonic preparation with respect to their metrical placement. In step three, the preparation of the I by the V pushed the I back by two measures while in step four, the preparation of V by V/V protrudes into the time domain of the preceding I.



(c) Generative syntax tree of the harmonic structure with integrated rhythmic information. The numbers in parentheses denote the duration of the constituents relative to the whole progression. The branch I(1) $\longrightarrow$ I($\frac{1}{2} \ominus \frac{1}{4}$) I($\frac{1}{4} \oplus \frac{1}{2}$) is an instance of a split that anticipates the upbeat preparation of G^7 by D^7. Because of a 2 measures long upbeat, the left child is 2 measures shorter and the right child is 2 measures longer than in a preparation without an upbeat.

Figure 1: Hierarchical analysis of the A-part of the Jazz standard *Take the A-Train* in C major, considering the structural domains of harmony and rhythm.

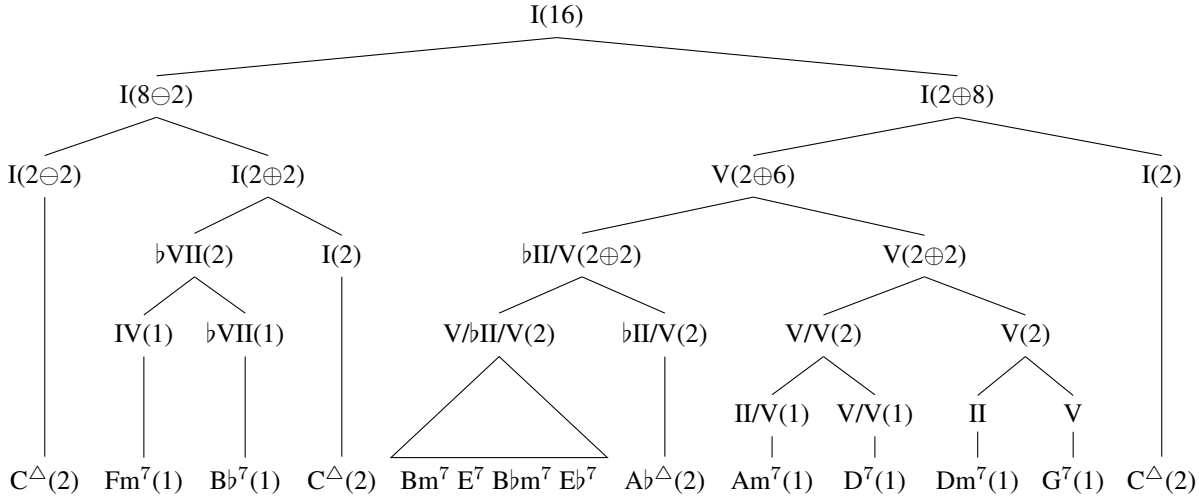


Figure 2: Hierarchical analysis of the Jazz standard *Half Nelson*, integrating harmonic and rhythmic structure. In this tree, a duration of 1 corresponds to one measure for the sake of readability (the whole tune spans 16 measures). The non-local dependency between the chords $A\flat^{\triangle}$ and G^7 constitutes a characteristic harmonic relation of the tune.

2.2 Product Grammars

This paper proposes to improve generative grammar models of harmony by forming a product of a harmony grammar and a rhythm grammar.

Let $G = (T, C, C_0, \Gamma)$ and $G' = (T', C', C'_0, \Gamma')$ be two PACFGs and let $\mathrm{ar}(r)$ denote the *arity* of a rule r, which is defined as the length of its right-hand side. The *product grammar*

$$G \bowtie G' = (T \times T', C \times C', C_0 \times C'_0, \Gamma \bowtie \Gamma') \quad (3)$$

is constructed from the Cartesian products of the sets of terminals, categories, and start categories. The rewrite functions of $G \bowtie G'$ are all pairs of functions of equal arity,

$$\Gamma \bowtie \Gamma' = \{\, (r, r') \in \Gamma \times \Gamma' \mid \mathrm{ar}(r) = \mathrm{ar}(r') \,\}. \quad (4)$$

For a product category $(A, A') \in C \times C'$ and rewrite functions $r \in \Gamma$ and $r' \in \Gamma'$ of equal arity, the application of (r, r') to (A, A') is defined component-wise,

$$(r, r')(A, A') = (r(A), r'(A')). \quad (5)$$

By abuse of notation, the right-hand side of this equation does not stand for a pair of sequences, but a sequence of pairs. The probability of a product rule application is defined as the product of the probabilities of the rule application components,

$$p((r, r') \mid (A, A')) = p(r \mid A)\, p(r' \mid A'). \quad (6)$$

That is, the choice of rule r is set to be independent of A' and r', and the choice of r' is independent of A and r in the generative process.

A helpful intuition of product grammars is that they compute the intersection of two sets of derivation trees for a sequence. The derivation trees of the grammar $G \bowtie G'$ are exactly those which are derivations in both G and G' if the labels of the trees (terminals and categories) are ignored. The probability of a derivation in $G \bowtie G'$ is then also equal to the product of its corresponding derivations in G and G'.

2.3 Rhythm Grammar

2.3.1 Full Rhythm Grammar

A *rhythmic category* $a \oplus b$ consists of two rational numbers $a \in \mathbb{Q}$ and $b \in \mathbb{Q}$ such that $0 \leq a$, $0 < b$, and $a + b \leq 1$. The first number a is called the *upbeat* and the second number b is called the *downbeat* of the category. The intuition behind the symbol $\oplus$ is that the total length of a rhythmic category equals the sum of its two components, $\lambda(a \oplus b) := a + b$, where λ is the function that denotes the length of the rhythmic constituent as a proportion of the overall piece, which is fixed to be the unit $1 \in \mathbb{Q}$. The condition $0 \leq a$ forbids negative upbeat parts, $0 < b$ ensures positive category lengths, and $a + b \leq 1$ ensures that no category is longer than the whole piece. For convenience, we use two additional short-hand notations: a category with no upbeat is denoted by the length of its downbeat, $b = 0 \oplus b$. The category of a rhythmic constituent that loses a portion c of its downbeat (formerly with length b) to the upbeat of the following rhythmic constituent is denoted by $b \ominus c := 0 \oplus (b - c)$. In this case $\lambda(b \ominus c) = b - c$, too.

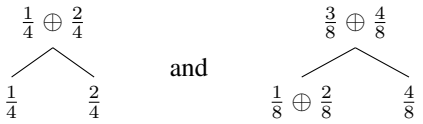
The start category of the rhythmic grammar is 1, the length of the piece, and any category with zero upbeat is allowed to be a terminal (leaf node). The essential grammar rules are given by two families of rewrite functions, one family of partial functions for splitting the upbeat components of categories $\texttt{usplit}_v : C \nrightarrow C^*$ and one family of total functions for splitting the downbeats $\texttt{dsplit}^u_v : C \to C^*$,

$$\begin{aligned} \texttt{usplit}_u(a \oplus b) &:= ((1-u)a \oplus ua) && (0 \oplus b) \\ \texttt{dsplit}^v_w(a \oplus b) &:= (a \oplus (1 - v - vw)b) && (vwb \oplus wb), \end{aligned} \quad (7)$$

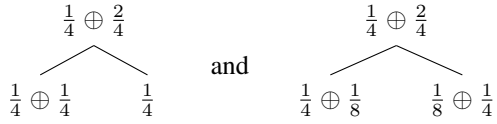
where $u, v, w \in \mathbb{Q}$ such that $\frac{1}{2} < u \leq 1$ and $a > 0$ in the first equation, and $0 \leq v < 1$ and $0 < w < 1$ in the second equation. The parameter u represents the downbeat

proportion of the upbeat, v is the upbeat proportion of the second category of a downbeat split, and w is the downbeat proportion of the second category of a downbeat split.

In other words: The upbeat split rule $\texttt{usplit}_u$ separates the upbeat from the downbeat and optionally splits the upbeat again into a new upbeat and downbeat. For example for $u = 1$ and $u = \frac{2}{3}$:



In contrast, the downbeat split $\texttt{dsplit}_w^v$ ignores the upbeat and splits the downbeat. It optionally introduces a new upbeat preparation. For example for $v, w = 0, \frac{1}{2}$ and $v, w = \frac{1}{2}, \frac{1}{2}$:



One rule $\texttt{unary}(a \oplus b) := a \oplus b$ is added to the grammar to ensure compatability with grammars that use rewrite rules of arity one.

The probability of a rhythmic rewrite functions does not depend on the particular rhythmic category that it rewrites, but only on whether or not the category has an upbeat of length zero. This enables a maximal sharing of probability mass by preserving consistency with the constraints of the rewrite rules. More precisely,

$$\begin{aligned} 1 = \; & p(\texttt{unary} \mid a \oplus b) \\ & + \sum_{\frac{1}{2} < u \leq 1} p(\texttt{usplit}_u \mid a \oplus b) \\ & + \sum_{0 \leq v < 1} \sum_{0 < w < 1} p(\texttt{dsplit}_w^v \mid a \oplus b) \end{aligned} \tag{8}$$

for $a > 0$ and

$$\begin{aligned} 1 = \; & p(\texttt{unary} \mid 0 \oplus b) \\ & + \sum_{0 \leq v < 1} \sum_{0 < w < 1} p(\texttt{dsplit}_w^v \mid 0 \oplus b). \end{aligned} \tag{9}$$

For practical applications, the parameters u, v, and w are limited to a finite set of rational numbers to put a proper normalized prior on the rule distributions.

2.3.2 Simplified Rhythm Grammar

For comparison, we additionally consider a simplified version of the rhythm grammar presented above which does not explicitly model upbeats. The rhythmic categories and the terminals of this grammar are rational numbers $0 < a \leq 1$ representing constituent durations relative to the full piece. Apart from the technical unary rule, the rules of the grammar form a family of total rewrite functions

$$\texttt{split}_s(a) := (sa) \qquad (a - sa). \tag{10}$$

The parameter $0 < s < 1$ is called the *temporal split ratio* of the rule. The probabilities of the rewrite rules are set to be independent from the category they rewrite. Therefore,

$$1 = p(\texttt{unary}) + \sum_{a \in \mathbb{Q}} p(\texttt{split}_a). \tag{11}$$

2.4 Harmony Grammar

The harmony grammar used in this paper is a standard probabilistic context-free grammar (Σ, N, S, R) in Chomsky normal form. It consists of a set Σ of chord symbols as terminal symbols, a set of copies of chord symbols N as non-terminal symbols, a distinguished start symbol $S \in N$, and a set of standard rewrite rules

$$R \subseteq \{\, A \longrightarrow B_1 \; B_2 \mid B_k \in N, A = B_1 \text{ or } A = B_2 \,\}.$$

In particular, rules of the form $A \longrightarrow A \; A$ are included by this definition. Each non-terminal symbol A is also associated with a random variable X_A over rewrite rules that have A as their left-hand side. The symbols, rules, and parameters of the grammar are read from dataset of tree annotations described in the next section.

Note that since every rewrite rule of a standard context-free grammar can be interpreted as a partial function with a singleton domain,

$$\text{dom}(A \longrightarrow \alpha) = \{\, A \,\} \quad \text{for all } \alpha \in (\Sigma \cup N)^*, \tag{12}$$

every standard context-free grammar is also an Abstract Context-free Grammar and can be used in the product grammar construction.

3. DATASET

This study uses a dataset of 75 hand-annotated tree analyses of Jazz chord sequences from the iRealPro dataset [23]. The tree annotations were performed by the authors and a student assistant. Each chord sequence is annotated with a single binary tree that spans the whole piece. In contrast to the introductory examples of this paper, the internal nodes of each tree in the data are not labeled by scale degrees but chord symbols. depth one subtrees corresponds to a rule of the grammar described in the previous section. Figure 3 shows the absolute frequencies of the 20 most frequent harmonic rewrite rules from the dataset, after each sequence was transposed to the root of C. Rules of the form $A \longrightarrow A \;\; A$, called *prolongation rules*, and rules of the form $A \longrightarrow B \;\; A$ for $A \neq B$, called *preparation rules*, are the most used rule schemes.

The dataset additionally includes the length of each chord in quarter notes. The chord durations of each piece are divided by the total duration of the piece. From the chord durations and the harmonic tree annotations, the duration of each constituent (subtree) can be calculated automatically as shown in Figure 4. The temporal split ratios of the rule applications–as introduced in Equation 10–are then in turn calculated from the durations of the constituents. Consider for example the rule application $\text{G}^7(\frac{5}{32}) \longrightarrow \text{F}^{\triangle}(\frac{2}{32}) \;\; \text{G}^7(\frac{3}{32})$ from Figure 4. Its temporal split ratio is $\frac{2}{5}$.

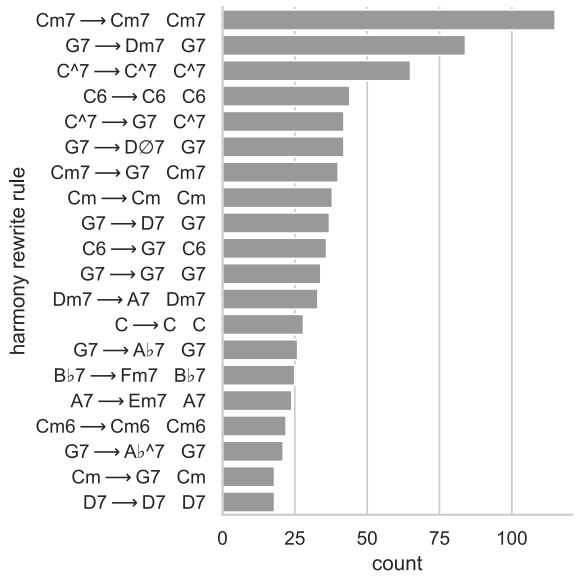


Figure 3: Absolute frequencies of the 20 most frequent harmonic rewrite rules of the tree annotations. All sequences are transposed to the common root C. Major-seventh chords are denotes as `C^7` and `A♭^7`.

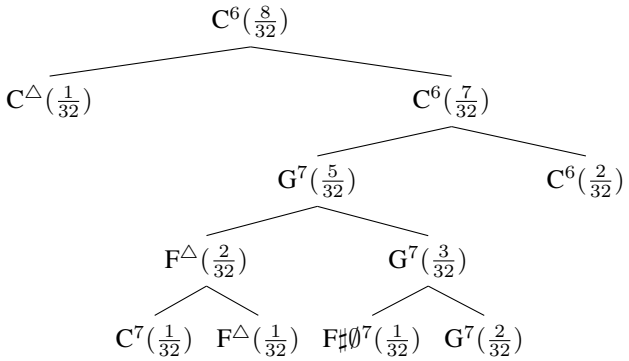


Figure 4: Tree annotation of the last chords of *St. Thomas*. Chord durations are shown relative to the total duration of the tune, $\frac{2}{32}$ corresponds to one measure. The durations of the inner nodes are calculated automatically.

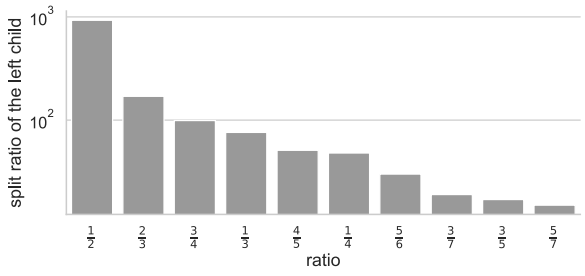


Figure 5: Absolute frequencies of the 10 most frequent split ratios of annotated tree constituents. The split ratio of a binary rewrite rule is defined as the time proportion of the left child. The y-axis is plotted using a logarithmic scale.

The 10 most frequent temporal split ratios are shown in Figure 5. The split ratio $\frac{1}{2}$ is by far the most frequent one. Most of the remaining ratios can be expressed either as $\frac{n-1}{n}$ or as $\frac{1}{n}$ for some $n \in \mathbb{N}$. The former arise for example from chains of descending fifths or applied dominants that accumulate time step by step in the temporal order of the piece. The latter arise from upbeat preparations that can be understood using the rhythmic categories described in Section 2.3.1. Two rhythmic rewrite rules that explain a split ratio of $\frac{1}{n}$ are $\left(\frac{n}{m}\right) \longrightarrow \left(\frac{\frac{n}{2}}{m} \ominus \frac{\frac{n}{2}-1}{m}\right) \quad \left(\frac{\frac{n}{2}-1}{m} \oplus \frac{\frac{n}{2}}{m}\right)$ and $\left(\frac{1}{m} \oplus \frac{n-1}{m}\right) \longrightarrow \left(\frac{1}{m}\right) \quad \left(\frac{n-1}{m}\right)$, where $m \in \mathbb{N}$. The former results from a downbeat split with $w = \frac{1}{2}$ and the latter results from an upbeat split with $u = 1$.

4. PARSING WITH PRODUCT GRAMMARS

A naive approach to parsing against a product grammar would enumerate all product categories and memoize the inverted rewrite rules on these categories. In this section, we show how the inefficient blow-up of the number of categories can be avoided using the independence assumption of Equation 6.

Consider an Abstract Context-Free Grammar in Chomsky normal form. The standard CYK algorithm–here used to calculate the probability of a sequence of terminals $w \in T^*$ of length n, indexed from 0 to $n-1$–can be formulated recursively by the equations

$$p(A, i, i) = \sum_{r \in \Gamma} p(A \longrightarrow_r w_i) \tag{13}$$

and

$$p(A, i, j) \tag{14}$$
$$= \sum_{k=i}^{j-1} \sum_{r \in \Gamma} p(A \longrightarrow_r B_1\ B_2) p(B_1, i, k)\ p(B_2, k+1, j)$$

where $A, B_1, B_2 \in C$ and $i, j \in \mathbb{N}$ such that $0 \leq i < j \leq n-1$. The probability of the sequence is then given by $p(w) = \sum_{A \in C_0} p(A, 0, n-1)$.

Given a product grammar $G \bowtie G'$, a sequence of product terminals can be parsed utilizing Equation 6,

$$p((A, A'), i, i) = \sum_{(r, r') \in \Gamma \bowtie \Gamma'} p(A \longrightarrow_r w_i)\ p(A' \longrightarrow_{r'} w'_i) \tag{15}$$

and

$$p((A, A'), i, j) = \sum_{k=i}^{j-1} \sum_{(r, r') \in \Gamma \bowtie \Gamma'} p(A \longrightarrow_r B_1\ B_2) \tag{16}$$
$$p(A' \longrightarrow_{r'} B'_1\ B'_2) p((B_1, B'_1), i, k)\ p((B_2, B'_2), k+1, j)$$

It is therefore sufficient to parse the component grammars individually at each step. In other words, the combined grammar is computed on-the-fly to achieve efficiency.

5. EXPERIMENTS

We compare four product grammars that integrate harmonic and rhythmic structure. Additionally, we report the performances of their single-domain components and of a random baseline. As first component, we consider the harmony grammar presented in Section 2.4, trained either on the annotations in the original keys of the tunes or on the annotations after each tune was transposed to C major. As second component, we consider the full rhythm grammar presented in Section 2.3.1 that distinguishes upbeats and downbeats of constituents, and its simplification that uses the total length of the constituents, presented in Section 2.3.2. All models are trained and evaluated on the dataset described in Section 3. Apart from the full rhythm grammar, all models are trained by counting the harmonic rewrite rules or the temporal split ratios present in the dataset. The full rhythm grammar is trained using variational Bayesian inference [8]. Every model predicts the latent tree structure of a given sequence using the maximum a posteriori tree. One-fold cross validation was applied to avoid overfitting to the data: 75 times the model was trained on 74 sequences and evaluated on the remaining sequence.

5.1 Evaluation Metric and Baseline

The similarity of two trees is calculated as the unlabeled tree accuracy, defined as follows. Let α be a sequence of n terminals, left-to-right indexed from 0 to $n-1$, let t be a tree with α as leafs, and let s be a subtree of t. The *span* of s is defined as the pair of the index of its left-most child and the index of its right-most child. The set of spans of t consists of the spans of all subtrees of t that are not leafs. The unlabeled tree accuracy of a tree prediction t to the respective Goldstandard tree t^* is then defined as the cardinality of the correctly predicted spans, divided by the total amount of spans of t^*.

Given a chord sequence of length n, the random baseline uniformly samples one tree from the set of all binary trees with n leafs.

5.2 Results and Discussion

The results of the computational experiments are shown in Figure 6. All combined models of harmony and rhythm perform significantly better than the single-domain harmony grammars and all models perform significantly better than the random baseline ($p < 0.01$ using 2-sample bootstrap tests). There is no statistical difference observable between the not transposed and the transposed harmony models. Surprisingly, the single-domain rhythm grammars perform much better than the single-domain harmony grammars. This is, however, only possible because we consider the unlabeled tree accuracy. Other measures such as perplexity would reveal the obvious incapability of the rhythm grammars to predict chord sequences.

Both rhythm grammars improve the harmony models similarly. As discussed in Section 3, the simplified version of the proposed rhythm grammar is also able to capture some complex rhythmical structures. The music-theoretically more sophisticated formalism, however, facilitates the interpretation and explanation of the observed split ratios.

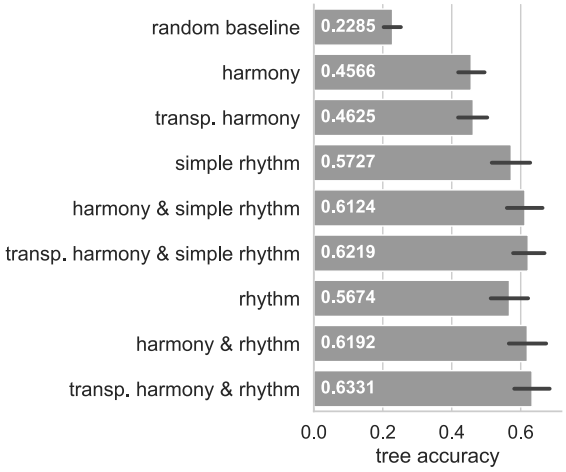


Figure 6: One-fold cross-validated tree accuracies of the tested models and the random baseline. The error bars show 95% bootstrap confidence intervals. The combined models of harmony and rhythm perform significantly better than the plain harmony grammars.

6. CONCLUSION

The usage of rhythmical information is shown to significantly improve the performance of harmonic syntax models. The empirical comparison between a music-theoretical motivated model and its simplified version shows that both models improve the harmony grammar equally well. The simplified model can therefore be used as an algorithmic proxy of the more expressive model. This might, however, only be true for rhythmically regular structures such as the harmonic rhythm of chord sequences from Jazz standards. It is, moreover, surprising how much information is already contained in the rhythm of the sequences, which underpins the importance of the rhythmic dimension of music [10]. In these sequences, both the harmonic syntax and the phrase rhythm work together to strengthen the intentionality of the music.

The here proposed model of interaction between harmony and rhythm is also capable to describe the interaction of pitch and rhythm in melodies. A rewrite function for syncopation could be added for future applications, since syncopation is an essential part of melodic rhythm.

The general product grammar construction presented in this paper integrates multiple domains of structure using strong independence assumptions. Future research can extent the formalism, explicitly modeling inter-domain dependencies. We hope that the presented approach will prove to be useful for applications such as rhythm quantization [2], the definition of similarity metrics [5], and computational composition assistance [15].

7. ACKNOWLEDGEMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 760081 – PMSB and from the Natural Sciences and Engineering Research Council of Canada (NSERC). We thank Claude Latour for supporting this research through the Latour Chair in Digital Musicology. The authors additionally thank the anonymous referees for their valuable comments.

8. REFERENCES

[1] Samer A. Abdallah and Nicolas E. Gold. Comparing models of symbolic music using probabilistic grammars and probabilistic programming. *Proceedings of ICMC/SMC*, pages 1524–1531, 2014.

[2] Francesco Foscarin, Florent Jacquemard, Philippe Rigaux, and Sakai Masahiko. A parse-based framework for coupled rhythm quantization and score structuring. In *Mathematics and Computation in Music*, pages 248–260, Cham, 2019. Springer International Publishing.

[3] Mark Granroth-Wilding and Mark Steedman. A Robust Parser-Interpreter for Jazz Chord Sequences. *Journal of New Music Research*, 43(4):355–374, October 2014.

[4] W Bas De Haas, José Pedro Magalhães, and Frans Wiering. Improving Audio Chord Transcription by Exploiting Harmonic and Metric Knowledge. *International Society for Music Information Retrieval Conference (ISMIR)*, pages 295–300, 2012.

[5] W Bas De Haas, Martin Rohrmeier, and Frans Wiering. Modeling Harmonic Similarity using a Generative Grammar of Tonal Harmony. *Proceedings of the Tenth International Conference on Music Information Retrieval (ISMIR)*, 2009.

[6] Daniel Harasim, Martin Rohrmeier, and Timothy J O'Donnell. A Generalized Parsing Framework for Generative Models of Harmonic Syntax. *19th International Society for Music Information Retrieval Conference*, 2018.

[7] Stefan Koelsch, Martin Rohrmeier, Renzo Torrecuso, and Sebastian Jentschke. Processing of hierarchical syntactic structure in music. *Proceedings of the National Academy of Sciences*, 110(38):15443–15448, 2013.

[8] Kenichi Kurihara and Taisuke Sato. An Application of the Variational Bayesian Approach to Probabilistic Context-Free Grammars. In *International Joint Conference on Natural Language Processing {(IJCNLP-04)} Workshop Beyond Shallow Analyses*, 2004.

[9] Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music*. Cambridge, MA, 1983.

[10] Florence Levé, Richard Groult, Guillaume Arnaud, Cyril Séguin, Rémi Gaymay, and Mathieu Giraud. Rhythm extraction from polyphonic symbolic music. In *12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, pages 375–380, 2011.

[11] Ryan McClelland. Extended upbeats in the classical minuet: Interactions with hypermeter and phrase structure. *Music Theory Spectrum*, 28(1):23–55, Spring 2006.

[12] Andrew McLeod and Mark Steedman. Meter Detection in Symbolic Music Using a Lexicalized PCFG. *Proceedings of the 14th Sound and Music Computing Conference*, 2017.

[13] Eita Nakamura, Masatoshi Hamanaka, Keiji Hirata, and Kazuyoshi Yoshii. Tree-structured probabilistic model of monophonic written music based on the generative theory of tonal music. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 276–280, 2016.

[14] Markus Neuwirth and Martin Rohrmeier. Towards a syntax of the Classical cadence. In *What is a Cadence*, pages 287–338. Leuven University Press, 2015.

[15] Alexandre Papadopoulos, Pierre Roy, and François Pachet. Assisted lead sheet composition using flowcomposer. In Michel Rueher, editor, *Principles and Practice of Constraint Programming*, pages 769–785, Cham, 2016. Springer International Publishing.

[16] Marcus Pearce and Martin Rohrmeier. Musical syntax ii: empirical perspectives. In *Springer handbook of systematic musicology*, pages 487–505. Springer, 2018.

[17] Gisela Pitsch. LR(k)-Parsing of Coupled-Context-Free Grammars. In *COLING 1994 Volume 1: The 15th International Conference on Computational Linguistics*, 1994.

[18] Donya Quick. Learning production probabilities for musical grammars. *Journal of New Music Research*, 45(4):295–313, 2016.

[19] Martin Rohrmeier. Towards a generative syntax of tonal harmony. *Journal of Mathematics and Music*, 5(1):35–53, 2011.

[20] Martin Rohrmeier and Ian Cross. Tacit tonality: Implicit learning of context-free harmonic structure. In *Proceedings of the 7th Triennial Conference of European Society for the Cognitive Sciences of Music (ESCOM 2009) Jyväskylä, Finland*, pages 443–452, 2009.

[21] Martin Rohrmeier and Marcus Pearce. Musical syntax i: Theoretical perspectives. In *Springer handbook of systematic musicology*, pages 473–486. Springer, 2018.

[22] Keith Salley and Daniel T. Shanahan. Phrase Rhythm in Standard Jazz Repertoire: A Taxonomy and Corpus Study. *Journal of Jazz Studies*, 11(1):1, 2016.

[23] Daniel Shanahan, Yuri Broze, and Richard Rodgers. A Diachronic Analysis of Harmonic Schemata in Jazz. In *Proceedings of the 12th International Conference on Music Perception and Cognition and the 8th Triennial Conference of the European Society for the Cognitive Sciences of Music*, pages 909–917, 2012.

[24] Mark J. Steedman. A Generative Grammar for Jazz Chord Sequences. *Music Perception: An Interdisciplinary Journal*, 2(1):52–77, 1984.

[25] Mark J Steedman. The blues and the abstract truth: Music and mental models. *Mental models in cognitive science: essays in honour of Phil Johnson-Laird*, pages 305–318, 1996.

[26] Hiroaki Tsushima, Eita Nakamura, Katsutoshi Itoyama, and Kazuyoshi Yoshii. Function- and Rhythm-Aware Melody Harmonization Based on Tree-Structured Parsing and Split-Merge Sampling of Chord Sequences. *Proceedings of the Tenth International Conference on Music Information Retrieval (ISMIR)*, pages 205–208, 2017.

[27] Hiroaki Tsushima, Eita Nakamura, Katsutoshi Itoyama, and Kazuyoshi Yoshii. Generative statistical models with self-emergent grammar of chord sequences. *Journal of New Music Research*, 47(3):226–248, May 2018.

LEARNING TO TRAVERSE LATENT SPACES FOR MUSICAL SCORE INPAINTING

Ashis Pati[1] **Alexander Lerch**[1] **Gaëtan Hadjeres**[2]
[1] Center for Music Technology, Georgia Institute of Technology, Atlanta, USA
[2] Sony CSL, Paris, France
ashis.pati@gatech.edu, alexander.lerch@gatech.edu, gaetan.hadjeres@sony.com

ABSTRACT

Music Inpainting is the task of filling in missing or lost information in a piece of music. We investigate this task from an interactive music creation perspective. To this end, a novel deep learning-based approach for musical score inpainting is proposed. The designed model takes both past and future musical context into account and is capable of suggesting ways to connect them in a musically meaningful manner. To achieve this, we leverage the representational power of the latent space of a Variational Auto-Encoder and train a Recurrent Neural Network which learns to traverse this latent space conditioned on the past and future musical contexts. Consequently, the designed model is capable of generating several measures of music to connect two musical excerpts. The capabilities and performance of the model are showcased by comparison with competitive baselines using several objective and subjective evaluation methods. The results show that the model generates meaningful inpaintings and can be used in interactive music creation applications. Overall, the method demonstrates the merit of learning complex trajectories in the latent spaces of deep generative models.

1. INTRODUCTION

Over the last decade, machine learning techniques have emerged as the tool of choice for the design of symbolic music generation models [1] with deep learning being the most widely used [2]. Deep generative models have been successfully applied to several different music generation tasks, e.g., monophonic music generation [3–5], polyphonic music generation [6, 7] and creating musical renditions with expressive timing and dynamics [8, 9]. However, most of these models assume sequential generation of music, i.e, the generated music depends only on the music that has preceded it. In other words, the models rely only on the past musical context. This approach does not align with typical human compositional practices which are often iterative and non-sequential in nature. In addition, the sequential generation paradigm places severe limitations on the degree of interactivity allowed by these models [10, 11]. Once generated, there is no way to tweak specific parts of the generation so as to conform to users' aesthetic sensibilities or compositional requirements.

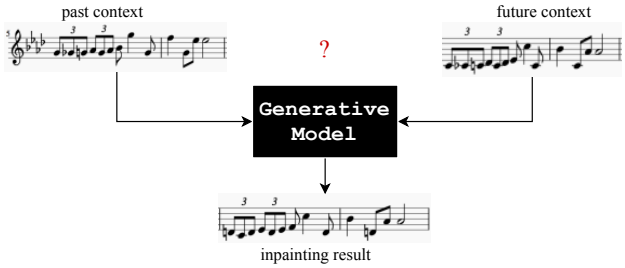


Figure 1: *Musical Score Inpainting* task schematic. A generative model needs to take past and future musical contexts into account to generate a sequence that can connect them in a musically meaningful manner.

In this paper, we seek to address these problems by incorporating future musical context into the generation process. Specifically, the task is to train models to fill in missing information in musical scores, duly taking into account the complete musical context — both past and future. In essence, this is similar to *inpainting* where the objective is to reconstruct missing or degraded parts of any kind of media [12]. For music, inpainting has been traditionally used for restoration purposes [13] or to remove unwanted artifacts such as clipping [14, 15] and packet loss [16]. However, we investigate models for *Musical Score Inpainting* (see Figure 1 and Section 3.1) as tools for music creation which can aid people in (i) getting new musical ideas based on specific styles, (ii) joining different musical sections together, and (iii) modifying or extending solos. In addition, such models can allow interactive music generation by enabling users to change the musical context and get new suggestions based on the updated context.

Our main technical contribution is a novel approach for musical score inpainting which relies on *latent* representation-based deep generative models. These models are trained to compress information from high-dimensional spaces, e.g., the space of all 1-bar melodies, to low-dimensional *latent* spaces. While these latent spaces have been shown to be able to encode hidden attributes of musical data (see Section 2.3), the primary form of interaction

with latent spaces has been using simple operations such as attribute vectors [17, 18] or linear interpolations [19, 20]. Using the proposed method (see Section 3), we demonstrate that Recurrent Neural Networks (RNNs) can be trained using latent embeddings to learn complex trajectories in the latent space. This, in turn, is used to predict how to fill in missing measures in a piece of symbolic music. Our secondary contributions are: (i) a stochastic training scheme which helps model training and generalization (see Section 3.4), and (ii) a novel data encoding scheme using uneven tick durations that allows encoding triplets without substantial increase in sequence length (see Section 3.5). The effectiveness of the proposed method is demonstrated using several objective and subjective evaluation methods in Section 4.

2. RELATED WORK

2.1 Audio & Music Inpainting

The first applications of audio inpainting methods were restoration-oriented [13, 14, 16, 21, 22] using different methods such as matrix factorization [14], non-local similarity measures [22] and audio similarity graphs [16]. While these techniques have been useful for audio-based tasks, they are not easily extendable to symbolic music.

For inpainting in the symbolic domain, the early attempts were based on Markov Chain Monte Carlo (MCMC) methods which allowed users to specify certain constraints, e.g., which notes to generate and which to retain [23, 24]. Another approach, proposed by Lattner et al., used iterative gradient descent to force the output of a deep generative model to conform to a specified structural plan [25]. However, methods based on MCMC (which rely on repeated sampling), and those using iterative gradient descent are slow during inference time and hence unsuitable for interactive applications. More recently, Hadjeres et al. proposed the AnticipationRNN framework [10] which used a pair of stacked RNNs to enforce user-defined constraints during inference. This allowed selective regeneration of specific parts of the music (generated or otherwise) using only two forward passes through the RNN-pair and enabled real-time generations.

2.2 Variational Auto-Encoders

The Variational Auto-Encoder (VAE) [26] is a type of generative model which uses an auto-encoding [27] framework; during training, the model is forced to reconstruct its input. The architecture comprises an encoder and a decoder. The encoder learns to map real data-points $\mathbf{x}$ from a high-dimensional data-space X to points in a low-dimensional space Z which is referred to as the *latent* space. The decoder learns to map the latent vectors back to the data-space. VAEs treat the latent vector as a random variable and model the generative process as a sequence of sampling operations: $\mathbf{z} \sim p(\mathbf{z})$, and $\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})$, where $p(\mathbf{z})$ is a prior distribution over the latent space, and $p(\mathbf{x}|\mathbf{z})$ is the conditional pdf. Variational inference [28] is used to approximate the posterior by minimizing the KL-divergence [29] between the approximate posterior $q(\mathbf{z}|\mathbf{x})$ and the true posterior $p(\mathbf{z}|\mathbf{x})$ by maximizing the evidence lower bound (ELBO) [26]. The training ensures that the reconstruction accuracy is maximized and realistic samples are generated when latent vectors are sampled using the prior $p(\mathbf{z})$.

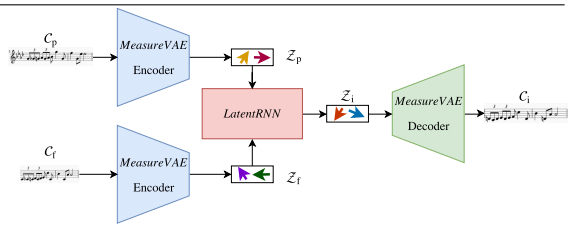


Figure 2: Schematic of the proposed approach. The pre-trained MeasureVAE encoder is used to convert the past and future context sequences ($\mathcal{C}_\mathrm{p}$ and $\mathcal{C}_\mathrm{f}$) into their respective latent vector sequences ($\mathcal{Z}_\mathrm{p}$ and $\mathcal{Z}_\mathrm{f}$). The LatentRNN learns to traverse the latent space of MeasureVAE to output a latent vector sequence $\mathcal{Z}_\mathrm{i}$ which is passed through the pre-trained decoder to output the inpainted musical sequence $\mathcal{C}_\mathrm{i}$.

2.3 Leveraging Latent Spaces for Music Generation

Latent representation-based models such as VAEs have been found to be quite useful for several music generation tasks. Bretan et al. used the latent representation of an auto-encoder-based model to generate musical phrases [30]. Lattner et al. forced the latent space of a gated auto-encoder to learn pitch interval-based representations which improved the performance of predictive models of music [31, 32]. Latent spaces of music generation models have also been used to explicitly encode and control musical attributes [3, 20, 33, 34], inter-track dependencies [35] and musical genre [36]. These studies show that trained latent spaces are able to encode hidden attributes of musical data which can be leveraged for different music generation tasks. However, latent space traversals have been relying on simpler methods such as attribute vectors [17, 18] or linear interpolations [19, 20].

3. METHOD

3.1 Problem Statement

We define the score inpainting problem as follows: given a past musical context $\mathcal{C}_\mathrm{p}$ and a future musical context $\mathcal{C}_\mathrm{f}$, the modeling task is to generate an inpainted sequence $\mathcal{C}_\mathrm{i}$ which can connect $\mathcal{C}_\mathrm{p}$ and $\mathcal{C}_\mathrm{f}$ in a musically meaningful manner. In other words, the model should be trained to maximize the likelihood $p(\mathcal{C}_\mathrm{i} \mid \mathcal{C}_\mathrm{p}, \mathcal{C}_\mathrm{f})$. Without much loss of generality, we assume that $\mathcal{C}_\mathrm{p}$, $\mathcal{C}_\mathrm{f}$, and $\mathcal{C}_\mathrm{i}$ comprise of n_p, n_f, and n_i measures of music, respectively.

3.2 Approach

The key motivation behind the proposed method is that the latent embeddings of deep generative models of music encode hidden attributes of music which can be leveraged to perform inpainting. Firstly, we train a VAE-model, referred to as *MeasureVAE*, to reconstruct single measures of music,

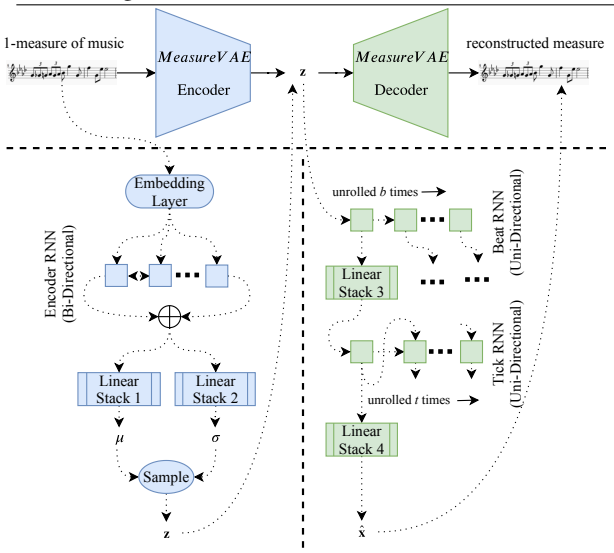


Figure 3: MeasureVAE schematic. Individual components of the encoder and decoder are shown below the main blocks (dotted arrows indicate data flow within the individual components). $\mathbf{z}$ denotes the latent vector and $\hat{\mathbf{x}}$ denotes the reconstructed measure.

i.e., the latent vectors of this model $\mathbf{z} \in Z$ map to individual measures of music. Once trained, the encoder of this model can be used to process sequences $\mathcal{C}_\mathrm{p}$ and $\mathcal{C}_\mathrm{f}$ and output corresponding latent vector sequences $\mathcal{Z}_\mathrm{p}$ and $\mathcal{Z}_\mathrm{f}$. Secondly, we train an RNN-based model, referred to as *LatentRNN*, to take as input the past and future latent vector sequences ($\mathcal{Z}_p$ and $\mathcal{Z}_\mathrm{f}$) and output a third latent vector sequence $\mathcal{Z}_\mathrm{i}$ which can be passed through the decoder of MeasureVAE to obtain $\mathcal{C}_\mathrm{i}$.

Effectively, the LatentRNN model learns to traverse the latent space of the MeasureVAE model so as to connect the provided contexts in a musically meaningful manner. The inference is fast since it only requires forward passes through the two models. This overall approach is shown in Figure 2. We call this joint architecture *InpaintNet*. While we restrict ourselves to $4/4$ monophonic melodic sequences in this paper, the approach can be extended to other time signatures and polyphonic sequences as well. The individual model architectures are discussed next.

3.3 Model Architectures

3.3.1 MeasureVAE

The MeasureVAE architecture (see Figure 3) is loosely based on the hierarchical recurrent MusicVAE architecture [3] which proved successful in modeling individual measures of music.

The encoder consists of a learnable embedding layer (operating on tick-level) followed by a bi-directional RNN [37]. The concatenated hidden state from both directions of the RNN is then passed through two identical parallel linear stacks to obtain the mean μ and variance σ which are used to sample the latent vector $\mathbf{z}$ via $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$.

The decoder follows a hierarchical structure where the

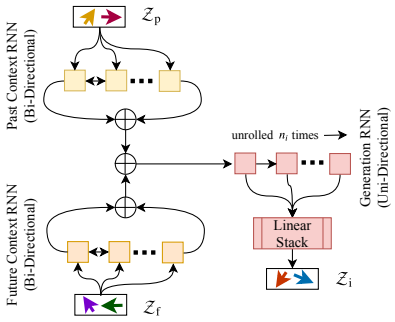


Figure 4: LatentRNN schematic. The Past-Context and Future-Context-RNNs encode $\mathcal{Z}_\mathrm{p}$ and $\mathcal{Z}_\mathrm{f}$, respectively. The Generation-RNN initialized using a concatenation of context-RNNs embeddings is unrolled n_i times to get $\mathcal{Z}_\mathrm{i}$.

sampled latent vector $\mathbf{z}$ is used to initialize the hidden state of a beat-RNN which is unrolled b times (where b is the number of beats in a measure). The output at each step of the beat-RNN is passed through a linear stack before being used to initialize the hidden state of a tick-RNN which is unrolled t times (where t is the number of events/ticks in a beat). The outputs of the tick-RNN are individually passed through a second linear stack which maps them back to the data-space. The hierarchical architecture mitigates the auto-regressive nature of the RNN and forces the decoder to use the latent vector more efficiently (as advocated in [3]).

3.3.2 LatentRNN

The LatentRNN model (see Figure 4) consists of 3 sub-components. There are 2 identical bi-directional RNNs, referred to as Past-Context-RNN and Future-Context-RNN, which process the latent vector sequences for the past and future contexts ($\mathcal{Z}_\mathrm{p}$ and $\mathcal{Z}_\mathrm{f}$), respectively. These are unrolled for n_p and n_f times in order to encode the context sequences, respectively. The final hidden states of the two context-RNNs are concatenated and then used to initialize the hidden state of a third RNN, referred to as the Generation-RNN, which is unrolled n_i times. The outputs of the Generation-RNN are passed through a linear stack to obtain n_i latent vectors corresponding to the inpainted measures.

The hyper-parameters for the model configurations are chosen based on initial experiments and are provided in Table 1. For the RNN layers in both models, Gated Recurrent Units (GRU) [38] are used.

3.4 Stochastic Training Scheme

We propose a novel stochastic training scheme for training the model. For each training batch, the number of measures to be inpainted n_i and the number of measures in the past context n_p are randomly sampled from a uniform distribution. Thus, the number of measures in the future context becomes $n_\mathrm{f} = N - n_\mathrm{i} - n_\mathrm{p}$, where N is the total number of measures in each sequence of the training batch. Using these, the input sequences are split into past, future and target sequences and the model is trained to predict

Measure VAE	
Embedding Layer	i=dict size, o=10
EncoderRNN	n=2, i=10, h=512, d=0.5
Linear Stack 1 Linear Stack 2	i=1024, o=256, n=2, non-linearity=SELU
BeatRNN	n=2, i=1, h=512, d=0.5
TickRNN	n=2, i=522, h=512, d=0.5
Linear Stack 3	i=512, o=1024, n=1, non-linearity=ReLU
Linear Stack 4	i=512, o=dict size, n=1, non-linearity=ReLU
Latent RNN	
Past-Context-RNN Future-Context-RNN	n=2, i=256, h=512, d=0.5
Generation RNN	n=2, i=1, h=1024, d=0.5
Linear Stack	i=2048, o=256, n=1, non-linearity=None

Table 1: Table showing configurations of both models. n: Number of Layers, i: Input Size, o: Output Size, h: Hidden Size, d: Dropout Probability, SELU: Scaled Exponential Linear Unit [39], ReLU: Rectifier Linear Unit

the target sequence given the past and future context sequences. This stochastic training scheme ensures that the model learns to deal with variable length contexts and can perform inpaintings at arbitrary locations.

3.5 Data Encoding Scheme

We use a variant of the encoding scheme proposed by Hadjeres et al. [24] for our data representation. The original encoding scheme quantizes time uniformly using the sixteenth note as the smallest sub-division. For each sub-division or tick, the note which starts on that tick is represented by a token corresponding to the note name. If no note starts on a tick, a special continuation symbol '__' is used to denote that the previous note is held. Rest is considered as a note and has a special token. The main advantages of this encoding scheme are (i) it uses only a single sequence of tokens, and (ii) uses real note names (e.g., separate tokens for A# and Bb) which allows generation of readable sheet music.

However, a limitation of using the sixteenth note as the smallest sub-division is that it cannot encode triplets. The naive approach of evenly subdividing the sixteenth note divisions to encode triplets increases the sequence length a factor of 3 which can make the sequence modeling task harder. To mitigate this limitation, we propose a novel uneven subdivision scheme. Each beat is divided into 6 uneven ticks (shown in Figure 5). This allows encoding triplets while only increasing the sequence length by a factor of 1.5. Consequently, each $4/4$ time signature measure is a sequence of 24 tokens.

4. EXPERIMENTS

The proposed method is compared with two baseline methods (see Section 4.1) using a dataset of monophonic folk melodies in the Scottish and Irish style taken from the Session website [5]. For the purposes of this work, only

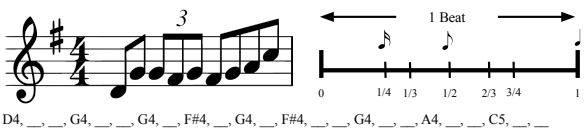


Figure 5: Figure showing the data representation. The token string on the bottom demonstrates the encoding scheme for the measure displayed on the top-left. Top-right shows the proposed uneven tick-duration scheme for each beat.

melodies with $4/4$ time signature in which the shortest note is greater than or equal to the sixteenth note are considered resulting in approx. 21000 melodies. Implementation details and source code are available online. [1]

4.1 Baseline

The performance of the proposed method is compared with the AnticipationRNN model proposed by Hadjeres et al. [10]. This model, referred to as *Base-ARNN*, uses a stack of 2 LSTM-based [40] RNN layers. Each of the 2 RNNs comprises of 2 layers with a hidden size of 256. In addition to the note-sequence tokens, this model also uses additional metadata information, i.e., tokens to indicate beat and downbeat locations as part of the user-defined constraints. For more details, the readers are directed to [10].

The original model operates on tick-level sequences and inpainting locations are specified in terms of individual tick locations. Hence, the inpainting locations may or may not be contiguous. In order to make a fair comparison, a second variant of the AnticipationRNN model is considered, referred to as *Reg-ARNN*, where the stochastic training scheme from Section 3.4 is used instead.

4.2 Training Configuration

The MeasureVAE model was pre-trained using single measures following the standard VAE optimization equation [26] with the β-weighting scheme [41, 42]. In order to prioritize high reconstruction accuracy, a low value of $\beta = 1\text{e}{-3}$ was used. Pre-training was done for 30 epochs resulting in a reconstruction accuracy of approx. 99%. While this seems to be better than results in [3], we attribute this to the shorter duration of generation (single measures) and the differences in datasets and data encoding. MeasureVAE parameters were frozen after pre-training and no gradient-based updates were performed on these parameters during the InpaintNet model training.

The Adam algorithm [43] was used for model training, with a learning rate of $1\text{e}{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1\text{e}{-8}$. To ensure consistency, all models were trained for 100 epochs (with early-stopping) with the same batch-size using a sub-sequence length of 16 measures (384 ticks). For the InpaintNet and Reg-ARNN models, the number of measures to be inpainted and the number of past measures were randomly selected: $n_\text{i} \in [2, 6]$, $n_\text{p} \in [1, 16 - 1 - n_\text{i}]$. This ensured that past and future contexts each contain at

[1] https://github.com/ashispati/InpaintNet

Model Variant	Test NLL
Base-ARNN	0.662
Reg-ARNN	0.402
InpaintNet (Our Method)	**0.300**
PastInpaintNet	0.643
FutureInpaintNet	0.481

Table 2: Table showing the average token-wise NLL (nats/token) on the held-out test set (lower is better). InpaintNet outperforms both baselines. The last two rows show the results for the ablation models described in Section 4.4.

least 1 measure. For the baseline models, teacher-forcing was used with a probability of 0.5.

4.3 Predictions on Test Data

Two experiments were conducted to evaluate the predictive power of the models.

The first experiment considered the average token-wise negative log-likelihood (NLL) on a held-out test set. The results (see first 3 rows of Table 2) indicate that our proposed model outperforms both baselines, showing an improvement of approx. 25% in the NLL over the Reg-ARNN model and approx. 55% over the Base-ARNN model.

The next experiment compared the models by varying the number of measures to be inpainted. Figure 6 shows the average token-wise NLL when n_{i} was increased from 2 to 8. Again, our proposed model outperforms both baselines. It should be noted that since the sub-sequence length is constant at 16 measures, increasing n_{i} means that the available context is reduced. Thus, there is an expected drop in the performance with increasing n_{i} as the models are forced to make longer predictions with less contextual information. However, the InpaintNet model performs better even when forced to predict beyond the training limit of 6 measures.

4.4 Ablations Studies

In order to further ascertain the efficiency of the proposed approach, ablation studies were conducted to evaluate the benefit of adding past and future context information. Specifically, we trained two variants of the InpaintNet model which relied on only one type of contextual information. The first model, referred to as PastInpaintNet only considered the past context $\mathcal{C}_{\mathrm{p}}$ as input whereas the second model, referred to as FutureInpaintNet considered only the future context $\mathcal{C}_{\mathrm{f}}$. The last two rows of Table 2 summarize the performance of these ablation models. It is clear that both past and future contexts are important for the modeling process. In addition, we also tried training a variant of the InpaintNet model with an untrained (randomly initialized) MeasureVAE model. This model failed to train properly achieving an NLL of approx. 1.33. This indicates that a structured latent space where latent vectors are trained to encode hidden data attributes is important for training the LatentRNN model.

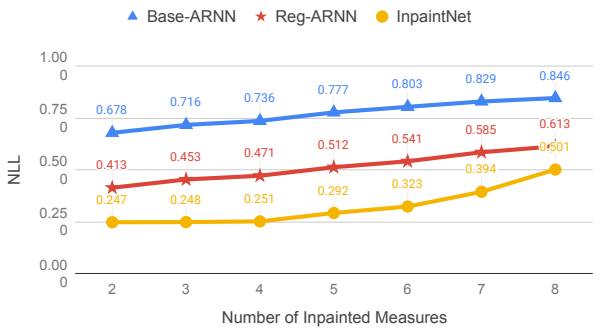


Figure 6: Figure showing token-wise NLL (nats/token) for different number of inpainted measures on the held-out test set (lower is better). InpaintNet outperforms both baselines. Models were trained to predict only 2 to 6 measures.

4.5 Qualitative Analysis

Considering that we are primarily interested in the aesthetic quality of the inpaintings, we encourage the readers to browse through the inpainting examples provided in the supplementary material. [2] We consider some of those examples in the analysis below.

Figure 7 shows sample inpaintings by the models for one of the melodies in the test set. While the Base-ARNN model collapses to produce long half notes which do not effectively reflect the surrounding context, the other two models do better. Both the Reg-ARNN and InpaintNet model generate rhythmically consistent inpaintings. The InpaintNet, in particular, mimics the rhythmic properties of the context better. For instance, measures 7 and 10 of the inpainted measures match the rhythm of measures 6, 14, and 15. Also, measure 8 matches measure 16. However, the use of G (subdominant scale degree in D-major) in the half-note to end measure 8 is unusual. We observed that in other examples also, the InpaintNet model occasionally produces pitches which are anomalous — either out-of-key or not fitting in the context. The Reg-ARNN model, on the other hand, tends to stay in key. Additional examples are provided in the supplementary material.

One advantage of working with the latent space is that the sampling operation, inherent in the VAE inference process, ensures that for the same context we can get different inpainting results. Figure 8 shows three such generations for the context of Figure 7. It is interesting to note that the base rhythm is retained across all three inpaintings. This feature is particularly interesting from an interactive music generation perspective, as this model can be used to quickly provide users with multiple ideas and will be investigated further in future work.

4.6 Subjective Listening Study

To evaluate the perceived quality of the inpainted measures, a listening test was conducted to compare our proposed model against the two baselines. A set of 30 melodies from the held-out test set were randomly selected and their first

[2] https://ashispati.github.io/inpaintnet/

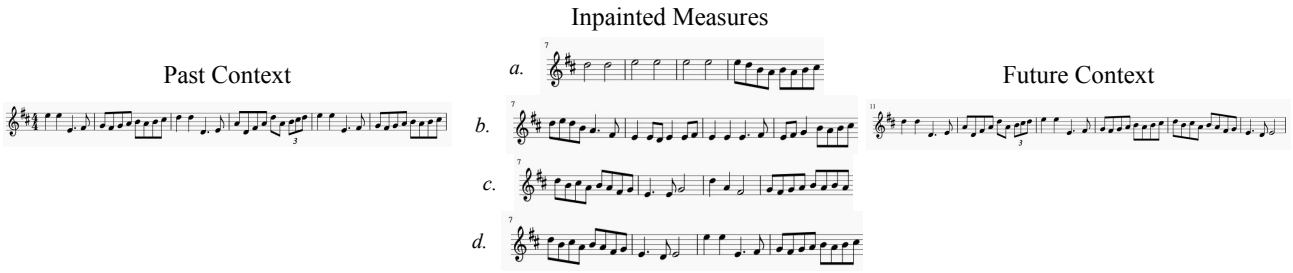


Figure 7: Figure showing the inpaintings generated by different models for the same context. From top to bottom — *a.*: Base-ARNN, *b.*: Reg-ARNN, *c.*: InpaintNet, *d.*: Original Melody.

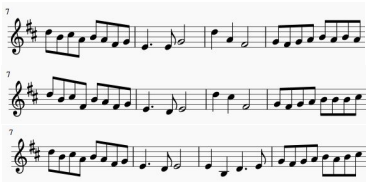


Figure 8: Figure showing different inpaintings (using the InpaintNet model) for the same context as Figure 7.

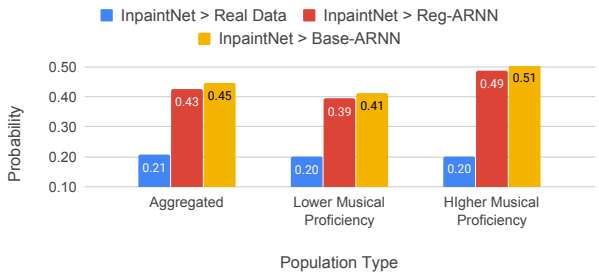


Figure 9: Figure based on the subjective listening study showing the probability that the InpaintNet model is rated higher. Analysis is based on the Bradley-Terry model [44, 45]. The proposed model loses against the real data but performs at par with the baseline models.

16 measures were extracted. The models were then used to inpaint 4 measures (measure number 7 to 10) in these melodic excerpts. Participants were presented with pairs of melodic excerpts and asked to select the one in which they thought the inpainted measures fit better within the surrounding context. In some of the pairs, one melodic excerpt was the real data (without any inpainting). Each participant was presented with 10 such pairs. A total of 72 individuals participated in the study (720 comparisons). The location of the inpainted measures was kept consistent across all examples so as to prevent confusion among participants and allow them to focus better on the inpainted measures.

The Bradley-Terry model [44,45] for paired comparisons was used to get an estimate of how the proposed model performs against the baselines and the real data (see Figure 9). While the proposed model expectedly has a very low probability of winning against the real data (wins approx. 1 out of 5 times), it performs only at par with the baseline models (with probability approx. 0.5). Significance tests using the Wilcoxon signed rank test were further conducted which validated that differences between the proposed model and the baselines were not statistically significant (p-value > 0.01). This was unexpected since the proposed model showed significant improvement over the baselines in the NLL metric. Further dividing the study population into two groups differing in musical proficiency (based on the Ollen index [46]) showed that, comparatively, the group with greater musical proficiency favored the generations from the InpaintNet model more than the group with less musical proficiency.

Additional analysis revealed that cases where the InpaintNet model performed the worst (maximum losses against the baselines), had anomalies in the predicted pitch similar to those discussed in Section 4.5. Specifically, they either had a single out-of-key note (e.g., F note in G-Major scale) or used a pitch or interval not used in the provided contexts. We conjecture that it is these anomalous pitch predictions which lead to poor perceptual ratings in spite of the model performing better in terms of modeling rhythmic features. This will be analyzed further in future studies.

5. CONCLUSION

This paper investigates the problem of musical score inpainting and proposes a novel approach to generate multiple measures of music to connect two musical excerpts by using a conditional RNN which learns to traverse the latent space of a VAE. We also improve upon the data encoding and introduce a stochastic training process which facilitate model training and improve generalization. The proposed model shows good performance across different objective and subjective evaluation experiments. The architecture also enables multiple generations with the same contexts, thereby, making it suitable for interactive applications [47]. We think the idea of learning to traverse latent spaces could be useful for other music generation tasks also. For instance, the architecture of the LatentRNN model can be changed to add contextual information from other voices/instruments to perform multi-instrument music generation. Future work will include a more thorough investigation of the anomalies in pitch prediction. A possible way to address that would be to add the context embedding as input at each step of unrolling the LatentRNN or use additional regularizers. Another promising avenue for future work is substituting RNNs with attention-based models [48] which have had success in sequential music generation tasks [9].

6. REFERENCES

[1] Rebecca Fiebrink, Baptiste Caramiaux, R Dean, and A McLean. *The machine learning algorithm as creative musical tool*. Oxford University Press, 2016.

[2] Jean-Pierre Briot and François Pachet. Deep learning for music generation: Challenges and directions. *Neural Computing and Applications*, Oct 2018.

[3] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical hatent vector model for learning long-term structure in music. In *Proc. of the 35th International Conference on Machine Learning (ICML)*, pages 4364–4373, Stockholmsmässan, Stockholm Sweden, 2018.

[4] Florian Colombo, Samuel P. Muscinelli, Alexander Seeholzer, Johanni Brea, and Wulfram Gerstner. Algorithmic composition of melodies with deep recurrent neural networks. In *Proc. of the 1st Conference on Computer Simulation of Musical Creativity (CSMC)*, 2016.

[5] Bob L Sturm, Joao Felipe Santos, Oded Ben-Tal, and Iryna Korshunova. Music transcription modelling and composition using deep learning. In *Proc. of the 1st Conference on Computer Simulation of Musical Creativity (CSMC)*, Huddersfield, UK, 2016.

[6] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. In *Proc. of International Society of Music Information Retrieval Conference (ISMIR)*, pages 324–331, Suzhou, China, 2017.

[7] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proc. of 29th International Conference on Machine Learning (ICML)*, Edinburgh, Scotland, 2012.

[8] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, pages 1–13, 2018.

[9] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. In *Proc. of International Conference of Learning Representations (ICLR)*, New Orleans, USA, 2019.

[10] Gaëtan Hadjeres and Frank Nielsen. Anticipation-RNN: Enforcing unary constraints in sequence generation, with application to interactive music generation. *Neural Computing and Applications*, Nov 2018.

[11] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. Deep learning techniques for music generation-A survey. *arXiv preprint arXiv:1709.01620*, 2017.

[12] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proc. of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000.

[13] Amir Adler, Valentin Emiya, Maria G Jafari, Michael Elad, Rémi Gribonval, and Mark D. Plumbley. Audio inpainting. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(3):922–932, 2012.

[14] Çağdaş Bilen, Alexey Ozerov, and Patrick Pérez. Audio declipping via nonnegative matrix factorization. In *Proc. of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 1–5. IEEE, 2015.

[15] Christopher Laguna and Alexander Lerch. An efficient algorithm for clipping detection and declipping audio. In *Proc. of the 141st AES Convention*, Los Angeles, USA, 2016.

[16] Nathanael Perraudin, Nicki Holighaus, Piotr Majdak, and Peter Balazs. Inpainting of long audio segments with similarity graphs. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(6):1083–1094, 2018.

[17] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3111–3119, 2013.

[18] Shan Carter and Michael Nielsen. Using artificial intelligence to augment human intelligence. *Distill*, 2017. https://distill.pub/2017/aia.

[19] Adam Roberts, Jesse Engel, Sageev Oore, and Douglas Eck. Learning latent representations of music to generate interactive musical palettes. In *Proc. of IUI Workshops*, 2018.

[20] Gaëtan Hadjeres, Frank Nielsen, and François Pachet. GLSR-VAE: Geodesic latent space regularization for variational autoencoder architectures. In *Proc. of IEEE Symp. Series on Computational Intelligence (SSCI)*, pages 1–7. IEEE, 2017.

[21] Çağdaş Bilen, Alexey Ozerov, and Patrick Pérez. Joint audio inpainting and source separation. In *Proc. of International Conference on Latent Variable Analysis and Signal Separation*, pages 251–258. Springer, 2015.

[22] Ichrak Toumi and Valentin Emiya. Sparse non-local similarity modeling for audio inpainting. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 576–580. IEEE, 2018.

[23] Jason Sakellariou, Francesca Tria, Vittorio Loreto, and François Pachet. Maximum entropy model for melodic patterns. In *Proc. of the ICML Workshop on Constructive Machine Learning*, Lille, France, 2015.

[24] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. DeepBach: A steerable model for Bach chorales generation. In *Proc. of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 1362–1371, Sydney, Australia, 2017.

[25] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints. *Journal of Creative Music Systems*, 2, March 2018.

[26] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proc. of International Conference of Learning Representations (ICLR)*, Banff, Canada, 2014.

[27] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proc. of the 25th International Conference on Machine learning (ICML*, pages 1096–1103, Helsinki, Finland, 2008.

[28] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational Inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[29] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[30] Mason Bretan, Gil Weinberg, and Larry Heck. A unit selection methodology for music generation using deep neural networks. In *Proc. of the 8th International Conference on Computational Creativity (ICCC)*, Atlanta, USA, 2016.

[31] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. A predictive model for music based on learned interval representations. In *Proc. of International Society of Music Information Retrieval Conference (ISMIR)*, pages 26–33, Paris, France, 2018.

[32] Andreas Arzt and Stefan Lattner. Audio-to-score alignment using transposition-invariant features. In *Proc. of International Society of Music Information Retrieval Conference (ISMIR)*, pages 592–599, Paris, France, 2018.

[33] Jesse Engel, Matthew Hoffman, and Adam Roberts. Latent constraints: Learning to generate conditionally from unconditional generative models. In *Proc. of International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[34] Ashis Pati and Alexander Lerch. Latent space regularization for explicit control of musical attributes. In *ICML Machine Learning for Music Discovery Workshop (ML4MD), Extended Abstract*, Long Beach, CA, USA, 2019.

[35] Ian Simon, Adam Roberts, Colin Raffel, Jesse Engel, Curtis Hawthorne, and Douglas Eck. Learning a latent space of multitrack measures. In *Proc. of the 2nd Workshop on Machine Learning for Creativity and Design*, Montréal, Québec, 2018.

[36] Gino Brunner, Andres Konrad, Yuyi Wang, and Roger Wattenhofer. MIDI-VAE: Modeling dynamics and instrumentation of music with applications to style transfer. In *Proc. of International Society of Music Information Retrieval Conference (ISMIR)*, pages 747–754, Paris, France, 2018.

[37] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[38] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proc. of 32nd International Conference on Machine Learning (ICML)*, pages 2342–2350, Lille, France, 2015.

[39] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 971–980, 2017.

[40] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[41] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. Beta-Vae: Learning basic visual concepts with a constrained variational framework. In *Proc. of International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[42] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proc. of the 20th Conference on Computational Language Processing*, 2015.

[43] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of International Conference on Learning Representations (ICLR)*, San Diego, USA, 2015.

[44] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[45] David R Hunter et al. MM algorithms for generalized Bradley-Terry models. *The annals of statistics*, 32(1):384–406, 2004.

[46] Joy E Ollen. *A criterion-related validity test of selected indicators of musical sophistication using expert ratings*. PhD thesis, The Ohio State University, 2006.

[47] Théis Bazin, Ashis Pati, and Gaëtan Hadjeres. A model-agnostic web interface for interactive music composition by inpainting. Neural Information Processing Systems (NeurIPS), 2018. Demonstration Track.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

DETECTING STABLE REGIONS IN FREQUENCY TRAJECTORIES FOR TONAL ANALYSIS OF TRADITIONAL GEORGIAN VOCAL MUSIC

Sebastian Rosenzweig[1] **Frank Scherbaum**[2] **Meinard Müller**[1]
[1] International Audio Laboratories Erlangen, Germany
[2] University of Potsdam, Potsdam, Germany
{sebastian.rosenzweig, meinard.mueller}@audiolabs-erlangen.de, frank.scherbaum@uni-potsdam.de

ABSTRACT

While Georgia has a long history of orally transmitted polyphonic singing, there is still an ongoing controversial discussion among ethnomusicologists on the tuning system underlying this type of music. First attempts have been made to analyze tonal properties (e. g., harmonic and melodic intervals) based on fundamental frequency (F0) trajectories. One major challenge in F0-based tonal analysis is introduced by unstable regions in the trajectories due to pitch slides and other frequency fluctuations. In this paper, we describe two approaches for detecting stable regions in frequency trajectories: the first algorithm uses morphological operations inspired by image processing, and the second one is based on suitably defined binary time–frequency masks. To avoid undesired distortions in subsequent analysis steps, both approaches keep the original F0-values unmodified, while only removing F0-values in unstable trajectory regions. We evaluate both approaches against manually annotated stable regions and discuss their potential in the context of interval analysis for traditional three-part Georgian singing.

1. INTRODUCTION

Polyphonic singing plays a vital role in many musical cultures. One of the oldest forms of polyphonic singing can be found in Georgia, a country located in the Caucasus region of Eurasia. The traditional three-part songs, which are typically passed down orally from one generation to the next, are acknowledged as Intangible Cultural Heritage by the UNESCO. Although being a long-studied subject, the non-tempered nature of traditional Georgian vocal music is still discussed controversially among musicologists [7,33]. So far, musicological studies on traditional Georgian music have mostly been conducted on the basis of manually transcribed field recordings. Such approaches are problematic, since important tonal cues (as well as many other performance aspects) are likely to get lost in the transcription process. The importance of field recordings in research on Georgian vocal music has raised the demand for computer-based methods to assist ethnomusicologists in analyzing the audio material.

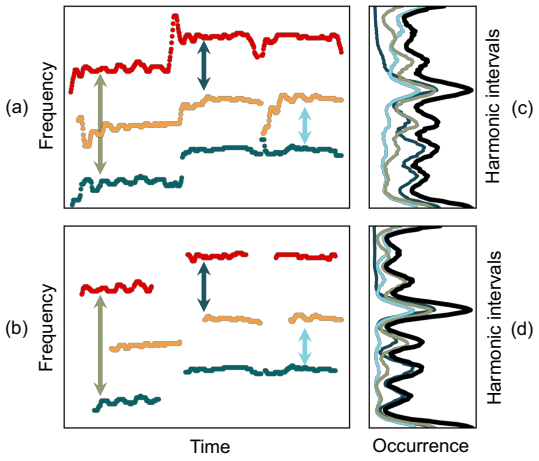


Figure 1. Detection of stable regions in F0-trajectories for a three-part singing recording. **(a)** Original F0-trajectories. **(b)** F0-trajectories restricted to stable regions. **(c)** Harmonic interval histogram based on (a). **(d)** Sharpened harmonic interval histogram based on (b). The histograms in (c) and (d) were computed considering the entire Erkomaishvili dataset.

One source of central importance for ethnomusicological research is a collection of audio recordings of the former master chanter Artem Erkomaishvili (1887–1967). The collection, which was created at the Tbilisi State Conservatory in 1966, comprises 101 three-part songs. Each chant was recorded in a three-stage "dubbing" process using tape recorders, where Erkomaishvili successively sung the individual voices with previously recorded voices being played back. In the study [20], a semi-automatic salience-based approach was applied to determine fundamental frequency (F0) trajectories of all three voices. The extracted F0-annotations are publicly available. [1] In a follow-up study [31], the authors determined from these trajectories harmonic (vertical) as well as melodic (horizontal) intervals, which give cues on the tonal organization [21, 22] of Georgian vocal music.

[1] https://www.audiolabs-erlangen.de/resources/MIR/2017-GeorgianMusic-Erkomaishvili

In general, studies on tonal analysis (see, e. g., [14, 15, 17, 31]) have shown that the usage of previously extracted F0-trajectories leads to various challenges. For example, as a stylistic element of traditional Georgian music, sung notes often start, end, or are continuously connected using pitch slides, see Figure 1a. Furthermore, automated F0-estimation procedures typically introduce inaccuracies such as extraction errors, outliers, or smoothing artifacts. Consequently, tonal analysis of Georgian vocal music based on highly fluctuating and error-prone F0-trajectories is problematic. For example, when computing harmonic interval statistics (as illustrated by Figure 1c), such artifacts may lead to an increased noise level and a less salient peak structure in the computed histograms. When analyzing melodic intervals, the presence of frequency variations (such as pitch slides) have a strong negative impact on subsequent analysis results. To alleviate such issues, contributions such as [17, 31] apply (semi-automatic) post-processing procedures to remove unstable regions in the trajectories and derive note-like events with a stable pitch. Note that for other scenarios (e. g. the tonal analysis of Hindustani Raga [29]), non-stable regions may contain musically important information.

Motivated by such tonal analysis applications, we present in this paper two automatic approaches that aim at identifying stable regions in frequency trajectories. Technically speaking, such regions correspond to horizontal structures (up to some tolerance) of trajectories. In acoustical and musical terms, such regions relate to pitched sounds where a singer has tuned into a harmonically stable pitch synchronized to other singers. In this context, our goal is to remove all frequency values in unstable regions, while keeping the original frequency values unmodified in the stable ones (see Figure 1b). For accomplishing this task, we introduce two conceptually different approaches—one based on morphological operations and the other one based on binary masking. Furthermore, we evaluate both approaches against manually annotated stable regions and indicate their potential for interval analysis using the Erkomaishvili recordings as example.

The remainder of this paper is organized as follows. We discuss related work in Section 2, then give a technical description of our approaches in Section 3, and summarize our experiments in Section 4.

2. RELATED WORK

In the following, we give an overview on work that is related to detecting stable regions in F0-trajectories. First, we want to note that stable region detection is not equivalent to F0-based transcription. In general, automated music transcription (AMT) aims at converting a music recording into some form of music notation [1, 2, 13]. In this process, many AMT systems apply temporal and spectral quantization of previously extracted F0-trajectories to derive pitches, onsets, and offsets of note events [3, 4, 6, 10, 15, 16, 18, 23, 30]. Rather than using quantized or modified F0-trajectories for our analysis, we aim at using trajectories restricted to stable regions (that may or may not correspond to note events) while leaving the original F0-values unmodified.

Detecting stable, transitional, and fluctuating patterns in F0-trajectories plays an important role for various tasks such as vibrato detection [5, 26, 37], singing style classification [24, 27], and motif detection [12, 25]. For example, in [35–37], the authors address the problem of detecting portamento (note transition) regions in Chinese string music. In [15], the authors identify stable regions as an important step towards transcribing recordings of Flamenco singing. In [19], the authors propose a vocal trajectory segmentation algorithm based on hysteresis defined on pitch–time curves. However, the underlying octave equivalence assumption may not be fulfilled in traditional Georgian vocal music. For a recent overview article of singing voice analysis, we refer to [11].

Furthermore, there are various studies on Indian Raga music, which are related to our work. In [9], a global pitch histogram ("pitch inventory") of the whole recording is computed. Then, informed by the histogram's peaks, stable regions are derived using empirically chosen thresholds for duration and fluctuation tolerance. In [14], the authors compute the local slope of the F0-trajectory and obtain stable regions by thresholding and quantization. However, due to the underlying scale assumptions, such approaches can not be directly applied to analyzing traditional Georgian singing, where pitch drifts may occur over the course of the song.

3. STABLE REGION DETECTION

In this section, we formalize the notion of a frequency trajectory as used in this work (Section 3.1). Then, to motivate the subsequent procedures, we introduce a simple median-based filtering approach (Section 3.2). As our main technical contributions, we introduce two conceptually different approaches for determining stable regions in frequency trajectories—one based on morphological operations (Section 3.3) and the other one based on binary masking (Section 3.4).

3.1 Frequency Trajectories

To account for the logarithmic nature of human pitch perception, we convert frequency values into the log-frequency domain. To this end, we fix a reference frequency ω_{ref} given in Hertz (Hz). In the following, we set $\omega_{\mathrm{ref}} = 55$ Hz. Then, an arbitrary frequency value ω is converted into the logarithmic domain by defining

$$F_{\mathrm{cents}}(\omega) := 1200 \cdot \log_2 \left(\frac{\omega}{\omega_{\mathrm{ref}}} \right), \tag{1}$$

which measures the distance between ω and ω_{ref} in cents. In this paper, we model a frequency trajectory as a function

$$\gamma : \mathbb{Z} \to \mathbb{R} \cup \{*\}, \tag{2}$$

which assigns to a given time index $n \in \mathbb{Z}$ either a real-valued frequency value $\gamma(n) \in \mathbb{R}$ (given in cents) or the symbol $\gamma(n) = *$ (when the frequency value is left to be

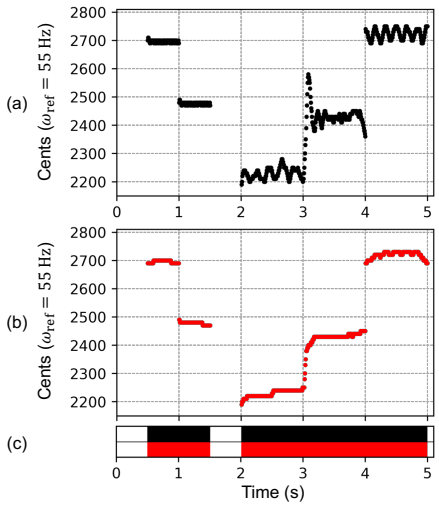


Figure 2. Effect of median filtering. **(a)** Original trajectory γ. **(b)** Median-filtered trajectory γ^{Median}. **(c)** Activation regions of γ (black) and γ^{Median} (red).

unspecified). In our implementation, we use a time resolution of 5.8 ms per time index and a frequency resolution of 10 cents. Figure 2a shows a frequency trajectory, which will serve as our running example in the remainder of this section. In the first two seconds, two notes are played on a piano without interruption. Subsequently, in the next two seconds, there are two sung notes smoothly connected by a pitch slide. Finally, the recording contains a note sung with vibrato.

3.2 Median Filtering

For tonal analysis based on frequency trajectories, one often applies some kind of filtering to remove outliers and other undesired pitch fluctuations [17, 32]. For example, by applying a median filter of odd length $L \in \mathbb{N}$, one obtains a smoothed trajectory γ^{Median} defined by

$$\gamma^{\text{Median}}(n) := \text{median}\big\{\gamma(n - \tfrac{L-1}{2} : n + \tfrac{L-1}{2})\big\} \quad (3)$$

for $n \in \mathbb{Z}$. In this definition, the symbol $*$ is handled as $-\infty$. Figure 2b shows γ^{Median} of our running example using $L = 69$ (corresponding to 0.4 sec). This example shows how median filtering introduces smoothing while removing outliers (such as the peak around the third second). However, the non-stable transition between the two sung notes remains after filtering. This is not what we aim at. First, we do not want to change frequency values in stable regions (with the goal not to introduce smoothing effects in subsequent tonal analysis steps). Second, we aim at explicitly detecting unstable regions, which can then be removed from the frequency trajectory. In the following, we present two conceptually different approaches that fulfill these requirements.

3.3 Morphological Approach

The first approach, which is inspired by work of Vávra et al. [34], uses morphological operations as known in image processing. Applying these operators to frequency trajectories, dilation corresponds to max filtering, and erosion to min filtering. Given a trajectory γ, this results in a dilated trajectory $\gamma^L_{\max}$ and an eroded trajectory $\gamma^L_{\min}$ defined by

$$\gamma^L_{\max}(n) := \max\big\{\gamma(n - \tfrac{L-1}{2} : n + \tfrac{L-1}{2})\big\}, \quad (4a)$$

$$\gamma^L_{\min}(n) := \min\big\{\gamma(n - \tfrac{L-1}{2} : n + \tfrac{L-1}{2})\big\}, \quad (4b)$$

for $n \in \mathbb{Z}$, where $L \in \mathbb{N}$ is assumed to be an odd integer. In max filtering, the symbol $*$ is handled as $-\infty$, whereas in min filtering it is handled as $+\infty$. Figure 3a shows the resulting trajectories $\gamma^L_{\max}$ and $\gamma^L_{\min}$ for our running example using $L = 43$ (corresponding to 0.25 sec). In a next step, we define the difference Δ^L between the dilated and eroded trajectories, also termed morphological gradient [28]:

$$\Delta^L(n) := \gamma^L_{\max}(n) - \gamma^L_{\min}(n) \quad (5)$$

for $n \in \mathbb{Z}$, where we set $\Delta^L(n) = *$ whenever $\gamma^L_{\max}(n)$ or $\gamma^L_{\min}(n)$ are not defined. As shown in Figure 3b, the difference Δ^L is large in non-stable parts (e. g., around the third second), whereas it is small in stable parts (e. g., within each of the piano notes). Fixing a suitable threshold $\tau > 0$ (given in cents), we define the trajectory γ^{Morph} by setting

$$\gamma^{\text{Morph}}(n) := \begin{cases} \gamma(n), & \text{for } |\Delta^L(n)| \leq \tau, \\ *, & \text{otherwise.} \end{cases} \quad (6)$$

The threshold τ can be seen as a tolerance parameter that specifies the maximally allowed fluctuation under which a

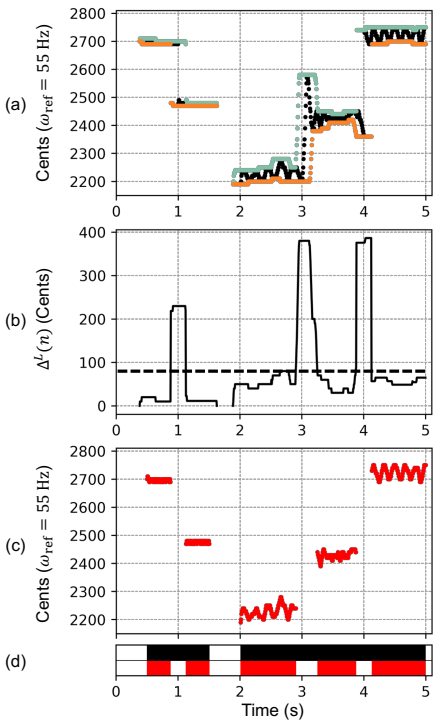


Figure 3. Morphological approach for detecting stable regions. **(a)** Frequency trajectories γ (black), $\gamma^L_{\max}$ (green), and $\gamma^L_{\min}$ (orange). **(b)** Morphological gradient Δ^L with threshold $\tau = 90$. **(c)** Trajectory γ^{Morph} restricted to stable regions. **(d)** Activation regions for γ (black) and γ^{Morph} (red).

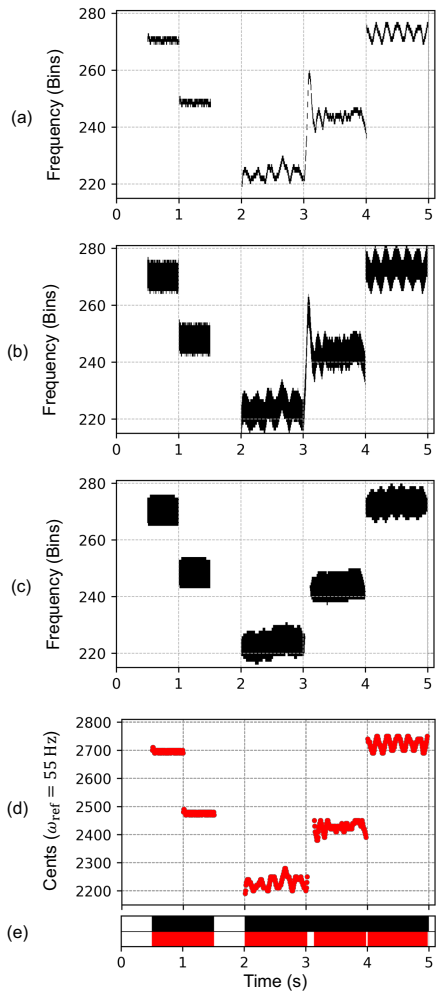


Figure 4. Masking approach for detecting stable regions. **(a)** Binary representation Γ_R. **(b)** Max-filtered representation Γ_R^β. **(c)** Median-filtered binary mask $\Gamma_R^{\beta,L}$. **(d)** Trajectory γ^{Mask} restricted to stable regions. **(e)** Activation regions for γ (black) and γ^{Mask} (red).

trajectory is still considered to be stable. The resulting trajectory γ^{Morph} for our running example is depicted in Figure 3c using a threshold of $\tau = 90$ cents. As shown in Figure 3d, the morphological approach succeeds in identifying stable regions. However, it also introduces a truncation at both sides of sudden jumps (e. g., around the first and fourth second) by half the filter length $(L-1)/2$. In the next section, we show how this truncation effect can be reduced by applying a 2D-masking approach involving some median filtering. Finally, we want to note that considering the morphological gradient is conceptionally similar to the approach based on Gaussian derivate filtering as described in [15]. In our approach, the threshold parameter τ can be adjusted dynamically to account for characteristics of individual trajectories, e. g. by considering the p-quantile of the morphological gradient Δ^L.

3.4 Masking Approach

We now introduce an alternative approach for detecting stable trajectory regions, which works in the 2D-domain. In a first step, we encode a trajectory γ as a binary 2D-representation $\Gamma_R : \mathbb{Z} \times \mathbb{Z} \to \{0, 1\}$. Given a frequency resolution of $R \in \mathbb{R}$ (given in cents), Γ_R is defined by

$$\Gamma_R(n, b) := \begin{cases} 1, & \text{for } \left\lfloor \frac{\gamma(n)}{R} + 0.5 \right\rfloor = b, \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

with time index $n \in \mathbb{Z}$ and frequency bin index $b \in \mathbb{Z}$ (corresponding to a logarithmic frequency axis). Figure 4a shows the binary representation Γ_R using $R = 10$ cents for our running example. In the second step, we introduce some tolerance in frequency direction by vertically applying a max-filtering using a filter length parameter $\beta \in \mathbb{N}_0$ (specified in bins). This results in the representation Γ_R^β defined by

$$\Gamma_R^\beta(n, b) := \max\{\Gamma_R(n, b - \beta : b + \beta)\}. \tag{8}$$

This operation is illustrated by Figure 4b using $\beta = 5$ (leading to a frequency width of $2\beta + 1 = 11$ bins corresponding to 110 cents). In a third step, inspired by an algorithm for Harmonic–Percussive Source Separation [8], a median filter of odd length $L \in \mathbb{N}$ is applied in horizontal direction yielding a representation $\Gamma_R^{\beta,L}$:

$$\Gamma_R^{\beta,L}(n, b) := \text{median}\big\{\Gamma_R^\beta(n - \tfrac{L-1}{2} : n + \tfrac{L-1}{2}, b)\big\}. \tag{9}$$

Applying horizontal median filtering suppresses vertical structures (e. g., pitch slides), while enhancing horizontal structures (corresponding to stable regions), see Figure 4c for an illustration when using $L = 43$ (corresponding to 0.25 sec). In the fourth step, the output trajectory γ^{Mask} is obtained by setting

$$\gamma^{\text{Mask}}(n) := \begin{cases} \gamma(n), & \text{if } \Gamma_R^{\beta,L}(n, b) = 1, \\ *, & \text{otherwise,} \end{cases} \tag{10}$$

with $b = \left\lfloor \frac{\gamma(n)}{R} + 0.5 \right\rfloor$. This last step can be thought of as "masking" the input trajectory γ using the binary mask $\Gamma_R^{\beta,L}$. Figure 4d shows the resulting trajectory γ^{Mask} for our running example. Note that, even though the masking procedure involves some quantization parameter R, the final trajectory γ^{Mask} coincides with the original trajectory γ in stable regions. Similar to the parameter τ for computing γ^{Morph}, the parameter β controls the frequency tolerance within stable regions for γ^{Mask}. As also indicated by our running example, the truncation effects at sudden jumps introduced by the morphological approach have been eliminated by our masking approach (compare γ^{Morph} and γ^{Mask} around the first and fourth second). While the 2D-masking approach is computationally more expensive than the 1D-morphological approach, it allows for processing multiple (non-overlapping) trajectories at the same time. Furthermore, one may account for weighted trajectories (e. g., trajectories with assigned amplitude or confidence values) by using real-valued instead of binary masks. Note that both algorithms do not enforce continuity of output trajectories. In particular, strict parameter settings (e. g. small τ and small β) may result in fluctuating sound events (e. g. a note sung with strong vibrato) being split up into several disconnected regions.

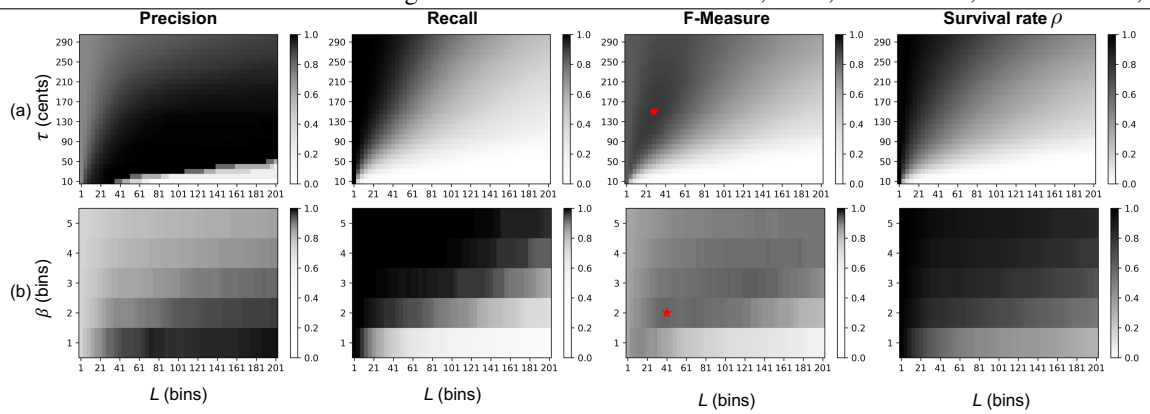


Figure 5. Precision, recall, F-Measure, and survival rate ρ of parameter sweeps averaged over five recordings (see Table 1). The parameter settings chosen for subsequent experiments are marked with red stars. **(a)** Morphological approach. **(b)** Masking approach.

ID	γ^{Anno}	γ^{Morph}				γ^{Mask}			
	ρ	P	R	F	ρ	P	R	F	ρ
001	61%	0.82	0.94	0.88	70%	0.82	0.94	0.88	71%
002	79%	0.94	0.85	0.89	72%	0.93	0.87	0.90	74%
010	68%	0.87	0.92	0.89	72%	0.84	0.95	0.89	77%
087	78%	0.88	0.98	0.93	87%	0.87	0.98	0.92	88%
110	74%	0.90	0.96	0.93	79%	0.88	0.97	0.92	80%

Table 1. Precision (P), recall (R), F-Measure (F), and survival rate (ρ) evaluated on the basis of manually annotated F0-trajectories for five Erkomaishvili recordings.

4. EVALUATION

In this section, we report on experiments that indicate the role of the parameters and the behavior of the morphological and the masking approach. In Section 4.1, we numerically compare both approaches using a set of manually annotated stable regions in F0-trajectories from the publicly available Erkomaishvili dataset [20]. Using suitable parameter settings, we then apply both algorithms to the trajectories of all 101 recordings in the dataset (see Section 4.2). It turns out that a consistent detection of stable regions using the two conceptually different approaches is a good indicator that the results are musically meaningful. Finally, in Section 4.3, we demonstrate the potential of our approaches for enhancing harmonic interval distributions.

4.1 Evaluation Measures and Parameters

In order to compare the algorithms' performance, we annotated stable regions of F0-trajectories extracted from five representative Erkomaishvili recordings. To this end, we used an interactive interface described in [20] to manually remove all unstable trajectory regions that correspond to note transitions and other artifacts. As evaluation metrics, we use standard precision (P), recall (R) and F-measure (F) computed frame-wise on the basis of the trajectories' activations. First, all frames with no specified frequency value in the original trajectory ($\gamma(n) = *$) are left unconsidered. Frames classified as stable by our approaches are counted as *true positives* (TP) if they agree with frames annotated as stable, otherwise they are counted as *false positives* (FP). Furthermore, frames annotated as unstable are counted as *false negatives* (FN), if they are classified as unstable. Then,

$$\mathrm{P} := \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \quad \mathrm{R} := \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \quad \mathrm{F} := \frac{2\cdot\mathrm{P}\cdot\mathrm{R}}{\mathrm{P}+\mathrm{R}}. \quad (11)$$

Note that $\mathrm{P} := 0$ for $\mathrm{TP}+\mathrm{FP} = 0$, $\mathrm{R} := 0$ for $\mathrm{TP}+\mathrm{FN} = 0$, and $\mathrm{F} := 0$ for $\mathrm{P}+\mathrm{R} = 0$. Furthermore, we introduce an evaluation measure referred to as *survival rate* and denoted as ρ. This measure, which indicates the percentage of remaining trajectory values after filtering, is defined as follows:

$$\rho := \frac{|\{n : \gamma^{\text{Stable}}(n) \neq *\}|}{|\{n : \gamma(n) \neq *\}|} \cdot 100, \quad (12)$$

with $\gamma^{\text{Stable}} = \gamma^{\text{Morph}}$ for the morphological approach, $\gamma^{\text{Stable}} = \gamma^{\text{Mask}}$ for the masking approach and $\gamma^{\text{Stable}} = \gamma^{\text{Anno}}$ for an annotated trajectory γ^{Anno}.

In order to analyze the algorithms' behavior for different parameter settings, we conduct parameter sweeps over L, τ, and β, using a fixed frequency resolution of $R = 10$ cents. For each evaluation metric, we construct a matrix with each entry corresponding to a metric's value for a specific parameter setting averaged over the five annotated recordings. The resulting matrices for precision, recall, F-measure, and survival rate are depicted in Figure 5a for the morphological approach and in Figure 5b for the masking approach. The visualizations show that τ and β play a similar role: high values of τ and β make the approaches more tolerant to local frequency fluctuations in the trajectories, thus increasing the survival rates. In contrast, when decreasing τ and β, less values remain in the filtered trajectories, leading to lower survival rates. Furthermore, note that increasing the filter length L leads to an increase in precision and a decrease in recall for both approaches. In the case of the morphological approach, very large filter lengths lead to a survival rate of $\rho = 0$ (nothing is remaining), which also leads to a precision of zero.

For our further experiments, we use fixed parameter settings for both approaches that correspond to maxima in the F-measure matrices (see red stars in Figure 5). The

	P	R	F	$\rho\,(\gamma^{\mathrm{Morph}})$	$\rho\,(\gamma^{\mathrm{Mask}})$
μ	0.89	0.94	0.92	73%	77%
σ	0.02	0.01	0.02	5%	5%

Table 2. Evaluation of the masking approach against the morphological approach considering the trajectories of all 101 recordings of the Erkomaishvili dataset (with fixed parameter settings from Section 4.1). The mean μ and standard deviation σ refer to statistics taken over the dataset.

morphological approach reaches a maximum F-measure of 0.90 for $\tau = 150$ cents and $L = 29$ bins, whereas the masking approach reaches a maximum F-measure of 0.90 for $\beta = 2$ bins and $L = 41$ bins. Using these parameter settings, the evaluation results for our five annotated examples (IDs correspond to songs on the publicly available website [2]) are given in Table 1. From the table, we can see that both approaches are able to detect stable regions in all five examples. We want to note that the optimal parameter settings vary from song to song, depending on the occurring note durations, characteristics of pitch slides, and other performance aspects. As an alternative to a fixed setting, one may chose the parameters in a song-dependent way, e. g., by fixing the survival rate. In summary, our experiments on the Erkomashvili dataset showed that the specific choice of parameters is not crucial within a certain range (see also the F-measure matrices of Figure 5).

4.2 Consistency

The two approaches for detecting stable regions in trajectories are conceptually different. Nevertheless, in the case of the five annotated recordings, both approaches worked successfully and performed in a similar fashion. Based on the hypothesis that a consistent performance of both approaches is a necessary condition for obtaining meaningful results, we applied both approaches independently to all 101 recordings of the Erkomaishvili dataset. We then compared the results by evaluating the trajectories obtained by the masking approach against the trajectories obtained by the morphological approach using the evaluation metrics defined in Section 4.1. The mean μ and standard deviation σ (taken over the dataset) of the evaluation results are shown in Table 2. The numbers indicate that both approaches deliver similar results on average with a small standard deviation. Furthermore, both approaches roughly exhibit the same average survival rate for the chosen parameter settings. Beyond these overall measures, we also looked at recordings where the two approaches delivered less consistent results. A manual inspection revealed that these recordings often contain speech-like passages (rather than singing) and extremely short notes such as in the songs with ID 022 and ID 074. Results for all 101 recordings are publicly available through audio–visual interfaces. [3]

[2] https://www.audiolabs-erlangen.de/resources/MIR/2017-GeorgianMusic-Erkomaishvili

[3] https://www.audiolabs-erlangen.de/resources/MIR/2019-ISMIR-StableF0

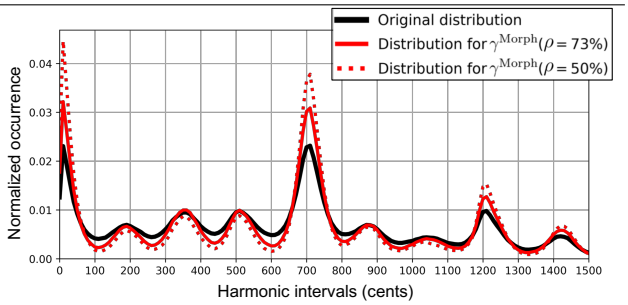


Figure 6. Harmonic interval distributions obtained from the entire Erkomaishvili dataset.

4.3 Harmonic Interval Analysis

In the following, we want to demonstrate the potential of the presented approaches for interval analysis of Georgian vocal music by computing harmonic interval size distributions from the filtered trajectories. To this end, similar to [20,31], we superimpose the filtered trajectories of lead, middle and bass voice and determine the frame-wise intervals for each voice pair (as indicated in Figure 1). Then, by accumulating the occurrences of the different intervals over time, we obtain interval histograms. These histograms are normalized (using the ℓ^1-norm) to obtain distributions. Figure 6 shows three such distributions obtained by considering all 101 recordings of the Erkomaishvili dataset. The first distribution (black solid line) is based on the original F0-trajectories. The second distribution (solid red line) is obtained by considering only stable regions after morphological filtering. (Here, we use the parameter setting discussed in Section 4.1. Filtering with the masking approach leads to similar distributions.) Note that the filtering leads to a sharper interval distribution emphasizing the peaks at the harmonically relevant intervals while not changing the respective peak locations. Using stricter parameter settings leads to a further sharpening (see red doted line in Figure 6). However, overdoing the filtering may drastically reduce the survival rate. This, in turn, may lead to a distortion or even a loss of peak structures corresponding to relevant harmonic intervals.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we presented two conceptually different approaches for detecting stable regions in frequency trajectories, which perform equally well with respect to a set of manually annotated trajectories. Rather than advocating a specific parameter setting, our goal was to introduce these concepts in a mathematical rigorous way, while highlighting their potential using the Erkomaishvili dataset as example scenario. Going beyond harmonic interval analysis, future work will be concerned with applying these filtering techniques for the analysis of melodic intervals, singer interaction, and intonation drifts—aspects of foremost importance in ethnomusicological research on traditional Georgian vocal music.

Acknowledgements: This work was supported by the German Research Foundation (DFG MU 2686/13-1, SCHE 280/20-1). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

6. REFERENCES

[1] Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36(1):20–30, 2019.

[2] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[3] Paul Brossier, Juan Pablo Bello, and Mark D. Plumbley. Fast labelling of notes in music signals. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Barcelona, Spain, 2004.

[4] Paul Brossier, Juan Pablo Bello, and Mark D. Plumbley. Real-time temporal segmentation of note objects in music signals. In *Proceedings of the International Computer Music Conference (ICMC)*, Miami, Florida, USA, 2004.

[5] Georgios Chrysochoidis, Georgios Kouroupetroglou, and Sergios Theodoridis. Vibrato detection in byzantine chant music. In *International Symposium on Communications, Control and Signal Processing (ISCCSP)*, pages 636–639, Athens, Greece, 2014.

[6] José Miguel Díaz-Báñez and Juan-Carlos Rizo. An efficient DTW-based approach for melodic similarity in flamenco singing. In *International Conference on Similarity Search and Applications*, pages 289–300, 2014.

[7] Malkhaz Erkvanidze. The Georgian musical system. In *Proceedings of the International Workshop on Folk Music Analysis*, pages 74–79, Dublin, Ireland, 2016.

[8] Derry FitzGerald. Harmonic/percussive separation using median filtering. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 246–253, Graz, Austria, September 2010.

[9] Kaustuv Kanti Ganguli and Preeti Rao. On the distributional representation of ragas: experiments with allied raga pairs. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 1(1):79–95, 2018.

[10] Emilia Gómez and Jordi Bonada. Towards computer-assisted flamenco transcription: An experimental comparison of automatic transcription algorithms as applied to A cappella singing. *Computer Music Journal*, 37(2):73–90, 2013.

[11] Eric J. Humphrey, Sravana Reddy, Prem Seetharaman, Aparna Kumar, Rachel M. Bittner, Andrew Demetriou, Sankalp Gulati, Andreas Jansson, Tristan Jehan, Bernhard Lehner, Anna Krupse, and Luwei Yang. An introduction to signal processing for singing-voice analysis: High notes in the effort to automate the understanding of vocals in music. *IEEE Signal Processing Magazine*, 36(1):82–94, 2019.

[12] Vignesh Ishwar, Shrey Dutta, Ashwin Bellur, and Hema A. Murthy. Motif spotting in an alapana in carnatic music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 499–504, Curitiba, Brazil, 2013.

[13] Anssi P. Klapuri and Manuel Davy, editors. *Signal Processing Methods for Music Transcription*. Springer, New York, 2006.

[14] Gopala Krishna Koduri, Sankalp Gulati, Preeti Rao, and Xavier Serra. Rāga recognition based on pitch distribution methods. *Journal of New Music Research*, 41(4):337–350, 2012.

[15] Nadine Kroher and Emilia Gómez. Automatic transcription of Flamenco singing from polyphonic music recordings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):901–913, 2016.

[16] Matthias Mauch, Chris Cannam, Rachel Bittner, George Fazekas, Justing Salamon, Jiajie Dai, Juan Bello, and Simon Dixon. Computer-aided melody note transcription using the Tony software: Accuracy and efficiency. In *Proceedings of the International Conference on Technologies for Music Notation and Representation*, May 2015.

[17] Matthias Mauch, Klaus Frieler, and Simon Dixon. Intonation in unaccompanied singing: Accuracy, drift, and a model of reference pitch memory. *Journal of the Acoustical Society of America (JASA)*, 136(1):401–411, 2014.

[18] Andrew McLeod, Rodrigo Schramm, Mark Steedman, and Emmanouil Benetos. Automatic transcription of polyphonic vocal music. *Applied Sciences*, 7(12), 2017.

[19] Emilio Molina, Lorenzo J. Tardón, Ana M. Barbancho, and Isabel Barbancho. Sipth: Singing transcription based on hysteresis defined on the pitch-time curve. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 23(2):252–263, 2015.

[20] Meinard Müller, Sebastian Rosenzweig, Jonathan Driedger, and Frank Scherbaum. Interactive fundamental frequency estimation with applications to ethnomusicological research. In *Proceedings of the AES International Conference on Semantic Audio*, pages 186–193, Erlangen, Germany, 2017.

[21] Aleksey Nikolsky. Evolution of tonal organization in music mirrors symbolic representation of perceptual reality. part-1: Prehistoric. *Frontiers in Psychology*, 6:1405, 2015.

[22] Aleksey Nikolsky. Evolution of tonal organization in music optimizes neural mechanisms in symbolic encoding of perceptual reality. part-2: Ancient to seventeenth century. *Frontiers in Psychology*, 7:211, 2016.

[23] Ryo Nishikimi, Eita Nakamura, Masataka Goto, Katsutoshi Itoyama, and Kazuyoshi Yoshii. Scale- and rhythm-aware musical note estimation for vocal F0 trajectories based on a semi-tatum-synchronous hierarchical hidden semi-markov model. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 376–382, Suzhou, China, 2017.

[24] Maria Panteli, Rachel M. Bittner, Juan Pablo Bello, and Simon Dixon. Towards the characterization of singing styles in world music. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 636–640, New Orleans, LA, USA, 2017.

[25] Preeti Rao, Joe Cheri Ross, Kaustuv Kanti Ganguli, Vedhas Pandit, Vignesh Ishwar, Ashwin Bellur, and Hema A. Murthy. Classification of melodic motifs in raga music with time-series matching. *Journal of New Music Research*, 43(1):115–131, 2014.

[26] Lise Regnier and Geoffroy Peeters. Singing voice detection in music tracks using direct voice vibrato detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1685–1688, Taipei, Taiwan, 2009.

[27] Rafael Caro Repetto, Rong Gong, Nadine Kroher, and Xavier Serra. Comparison of the singing style of two jingju schools. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 507–513, 2015.

[28] Jean-François Rivest, Pierre Soille, and Serge Beucher. Morphological gradients. *Journal of Electronic Imaging*, 2(4):326–336, 1993.

[29] Joe Cheri Ross, Vinutha T. P., and Preeti Rao. Detecting melodic motifs from audio for hindustani classical music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 193–198, Porto, Portugal, 2012.

[30] Matti Ryynänen and Anssi P. Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32(3):72–86, 2008.

[31] Frank Scherbaum, Meinard Müller, and Sebastian Rosenzweig. Analysis of the Tbilisi State Conservatory recordings of Artem Erkomaishvili in 1966. In *Proceedings of the International Workshop on Folk Music Analysis*, pages 29–36, Málaga, Spain, 2017.

[32] Johan Sundberg. Perceptual aspects of singing. *Journal of voice*, 8(2):106–122, 1994.

[33] Zaal Tsereteli and Levan Veshapidze. On the Georgian traditional scale. pages 288–295, Tbilisi, Georgia, 2014.

[34] František Vávra, Pavel Nový, Hana Mašková, Michala Kotlíková, and Arnoštka Netrvalová. Morphological filtration for time series. In *Conference on Applied Mathematics (APLIMAT)*, pages 983–990, Bratislava, Slovakia, 2004.

[35] Luwei Yang, Elaine Chew, and Khalid Z. Rajab. Logistic modeling of note transitions. In *International Conference on Mathematics and Computation in Music (MCM)*, pages 161–172, London, UK, 2015.

[36] Luwei Yang, Khalid Z. Rajab, and Elaine Chew. AVA: an interactive system for visual and quantitative analyses of vibrato and portamento performance styles. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 108–114, New York City, USA, 2016.

[37] Luwei Yang, Khalid Z. Rajab, and Elaine Chew. The filter diagonalisation method for music signal analysis: frame-wise vibrato detection and estimation. *Journal of Mathematics and Music*, 11(1):42–60, 2017.

THE ACOUSTICBRAINZ GENRE DATASET: MULTI-SOURCE, MULTI-LEVEL, MULTI-LABEL, AND LARGE-SCALE

Dmitry Bogdanov[1] **Alastair Porter**[1] **Hendrik Schreiber**[2] **Julián Urbano**[3] **Sergio Oramas**[4]
[1] Music Technology Group, Universitat Pompeu Fabra, Spain
[2] tagtraum industries incorporated, USA
[3] Multimedia Computing Group, Delft University of Technology, Netherlands
[4] Pandora, USA
dmitry.bogdanov@upf.edu, alastair.porter@upf.edu, hs@tagtraum.com,
urbano.julian@gmail.com, soramas@pandora.com

ABSTRACT

This paper introduces the AcousticBrainz Genre Dataset, a large-scale collection of hierarchical multi-label genre annotations from different metadata sources. It allows researchers to explore how the same music pieces are annotated differently by different communities following their own genre taxonomies, and how this could be addressed by genre recognition systems. Genre labels for the dataset are sourced from both expert annotations and crowds, permitting comparisons between strict hierarchies and folksonomies. Music features are available via the AcousticBrainz database. To guide research, we suggest a concrete research task and provide a baseline as well as an evaluation method. This task may serve as an example of the development and validation of automatic annotation algorithms on complementary datasets with different taxonomies and coverage. With this dataset, we hope to contribute to developments in content-based music genre recognition as well as cross-disciplinary studies on genre metadata analysis.

1. INTRODUCTION

Content-based music genre recognition (MGR) is a popular task in Music Information Retrieval (MIR) research [27]. The goal is to build systems that can predict the genre or subgenre of unknown music recordings (tracks, songs) using music features automatically computed from audio of those recordings. Such research can be supported by recent developments in the context of the AcousticBrainz [1] project, which facilitates access to a large dataset of music features [21] and metadata [22]. AcousticBrainz is a community database containing music features extracted from over four million distinct audio files [2] uniquely identified by public MusicBrainz Identifiers (MBID) [3] and thus tied to rich textual metadata. Users who contribute to the project run software on their computers to process their personal music collections and submit features to the AcousticBrainz database. Based on these features, additional metadata not already included in MusicBrainz, like mood, tempo, key, and genres can be estimated from content-based features in the database.

To facilitate new research in MGR, we have curated four supplemental genre datasets mapped to recordings in AcousticBrainz and containing fine-grained, hierarchical genre annotations, derived from both crowdsourced labels and expert annotations. Each of the four datasets contains multiple labels featuring hundreds of subgenres covering in total over 2,086,000 recordings, which are connected to AcousticBrainz via MBIDs. We refer to the combination of the four datasets and the music features from AcousticBrainz as the AcousticBrainz Genre Dataset. The four main characteristics of this new dataset are:

- **Multi-source.** It allows us to explore how the same music can be annotated differently by communities who follow their own genre taxonomies, and how this can be addressed when developing and evaluating MGR systems. This is especially valuable, because it has been previously noted that the evaluation of MGR systems is difficult due to subjectivity in genre annotations, with little inter-annotator agreement [8]. We are not aware of any other dataset offering such a unique and comprehensive view on genres.
- **Multi-level.** We provide information about the hierarchy of genres and subgenres within each annotation source. Previous research typically used a small number of broad genre categories. According to Sturm's 2012 survey [26], the most popular public datasets for automatic genre recognition were GTZAN and ISMIR04 [7, 13, 28], with 10 and 6 genres, respectively. Only 3.7% of the surveyed systems used 25 or more la-

[1] https://acousticbrainz.org

[2] As of April 2019.

[3] https://musicbrainz.org/doc/MusicBrainz_Identifier

Dataset	GTZAN [28]	Rosamerica [14]	FMA [9]	USPop [1]	KPop [17]	MSD [2]	RWC [12]	Ballroom [13]	ISMIR04 [7]	Acousticbrainz
Recordings	1,000	400	106,574	7,000	1,894	1,000,000	100	698	729	692,217–1,935,991
Genres	10	8	16	10	7	No[1]	10	9	6	15–31
Subgenres	—	—	161	—	—	—	33	—	—	265–745
Hierarchical	No	No	Yes	No	No	No	Yes	No	No	Yes
Multi-Label	No	No	Yes	No	No	No	No	No	No	Yes
Audio	Yes	Yes[2]	Yes	No	No	No[3]	Yes	Yes	Yes	No
Public ID	No	No	Yes	No	No	Yes	No	No	No	Yes

[1] While the original dataset only contains free-form tags and no explicit genre labels, there have been several attempts to map MSD-tracks to genres [10, 23, 24]. [2] Available upon request. [3] 7-Digital previews have been available.

Table 1: Popular genre recognition datasets, compared to the proposed AcousticBrainz Genre Dataset.

bels. In contrast, our dataset contains dozens of genres and hundreds of subgenres.

- **Multi-label.** Genre recognition is often treated as a single category classification problem, likely because existing datasets are often single-label (e.g., GTZAN [28] or Ballroom [13]; see Table 1). Yet, previous studies suggest that if there is a diversity of responses in terms of genre labels to any particular recording, the standard evaluation methodology that uses single genre category as ground truth is not adequate [8,20]. Our data is intrinsically multi-label, which allows treating genre recognition as a multi-label classification problem.
- **Large-scale.** MIR research is often performed on small music collections. We provide a very large dataset with audio features for over two million recordings annotated with genres and subgenres. However, we only provide precomputed features, not audio.

Compared to popular MGR datasets (see Table 1), the AcousticBrainz Genre Dataset is unique in that it is the only one that has all of these characteristics, which opens up interesting research opportunities. The remainder of the paper is structured as follows. We describe the dataset in detail in Section 2. In Section 3, we report on how the data has already been used for a task held within MediaEval 2017–18 [3, 4]. Section 4 describes a baseline implementation, and finally Section 5 presents our conclusions.

2. DATASET

The AcousticBrainz Genre Dataset dataset consists of genre annotations (Section 2.1) and precomputed music features (Section 2.2), distributed in predefined splits (Section 2.3). All related information about the dataset including downloads, data format, and baselines is available online. [4]

2.1 Genre Annotations

We provide four datasets with genre and subgenre annotations extracted from different online metadata sources. Two sources feature expert annotations using a strict taxonomy, two others use free-form tags from users: [5]

- **AllMusic** [6] and **Discogs** [7] are based on editorial metadata databases maintained by music experts and enthusiasts. These sources contain explicit genre/subgenre annotations of music albums following predefined genre taxonomies. To build the datasets we assumed that the annotations for an album also correspond to all of the recordings it contains. AllMusic data has been previously used [23] to provide genre annotations for the Million Song Dataset [2], while Discogs has been recently proposed as an alternative source of genre metadata for MIR [5]. To retrieve annotations from these sources we used the artist, album name and year metadata associated with each recording in AcousticBrainz. AllMusic has no publicly available API, and therefore we used a scraper to parse HTML data directly from the website. For Discogs, its public API was used. Annotations in AllMusic contain up to three levels of hierarchy, which we simplified to two levels by taking the most generic and the most specific annotations.
- **Lastfm** [8] is based on a collaborative music tagging platform with large amounts of genre labels provided as folksonomy tags by its users for music recordings. **Tagtraum** [9] is similarly based on genre labels collected from users of the music tagging application beaTunes. [10] To retrieve labels from the Lastfm API and genre annotations from the Tagtraum database we queried them using used artist names and recording titles. We then automatically inferred a genre/subgenre taxonomy and annotations from these labels following the algorithm proposed in [24]. This procedure exploits the fact that co-occurrences for genres are usually asymmetrical. For example, while Alternative Rock almost always co-occurs with Rock, Rock does not necessarily co-occur with Alternative Rock. This lets us derive a hierarchy. We performed manual post-processing to consolidate spelling variations and to remove location and era names (e.g., "50s", "Canadian") or labels that were clearly not a genre (e.g., "awesomelyrics").

Each source's genre taxonomy varies in class space,

[4] https://mtg.github.io/acousticbrainz-genre-dataset

[5] The resulting genre metadata is licensed under CC BY-NC-SA4.0 license, except for data extracted from the AllMusic database, which is released for non-commercial scientific research purposes only.

[6] https://allmusic.com

[7] https://discogs.com

[8] https://last.fm

[9] http://www.tagtraum.com

[10] https://www.beatunes.com

Dataset	AllMusic	Discogs	Lastfm	Tagtraum
Type	Explicit	Explicit	Tags	Tags
Annotation level	Album	Album	Track	Track
Recordings	1,935,991	1,290,489	806,627	692,217
Release groups[11]	233,789	169,109	164,290	98,333
Genres	21	15	30	31
Subgenres	745	300	297	265
Genres/track	1.33	1.37	1.14	1.13
Subgenres/track	3.14	1.70	1.28	1.72

Table 2: Overview of the AcousticBrainz Genre Dataset. Data is split in 70/15/15% for training, validation and test.

specificity, and breadth, and has its own definitions for the classes (i.e., the same label may have different meanings in difference sources). Most importantly, annotations in each source are multi-label: there may be multiple genre and subgenre annotations for the same music recording. It is guaranteed that each recording has at least one genre label, but subgenres are not always present.

Table 2 provides an overview of the entire AcousticBrainz Genre Dataset. The bottom rows show the size of the genre taxonomies in each source. Compared to the others, the AllMusic taxonomy comprises few genres, but is much richer in terms of subgenres. Conversely, the Tagtraum taxonomy has the most genres, but the least number of subgenres. Figure 1 shows the distributions of genres in all four sets, where we can appreciate clear biases towards pop, rock and electronic. [12] This bias seems less acute in the Discogs and Lastfm sets. Figure 2 shows how label counts are distributed in all four datasets. In terms of genres, most recordings are annotated with only one genre, with some having as many as 8 genres in AllMusic and Discogs. In terms of subgenres, most recordings in the simpler Tagtraum and Lastfm sets are annotated with 1 or 2 subgeners, but in the more complex AllMusic and Discogs sets we find 10 or more subgenre annotations for some recordings. We can see that the distribution in AllMusic is quite smooth, while in the other sets we see clear biases towards 1 genre and 1 or 2 subgenres. We did not aim to create a representative or unbiased dataset, instead collecting as much data as possible for recordings in AcousticBrainz. We understand that biases likely exist due to the coverage of MusicBrainz, AcousticBrainz, and the sources of genre information.

A more detailed picture of the complexity and similarity among datastets can be made in terms of entropy of the label distributions. In particular, we may compute the conditional entropy of a dataset X given another dataset Y:

$$H(X|Y) = -\sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x,y)\log\frac{p(x,y)}{p(y)}\,, \qquad (1)$$

where $\mathcal{X}$ and $\mathcal{Y}$ are the taxonomies of X and Y, respectively. Eqn (1) computes the amount of information needed to describe a recording in X given its labels in Y. For simplicity, we ignore the multi-label nature of the data and set $p(x)$ equal to the probability that a recording contains the label x, ignoring the other labels in the same recording. As a byproduct, this allows us to compute $H(X|X) \neq 0$, understood as the amount of information needed to fully describe a recording in X when some label in X is already known. To make this distinction explicit, let us refer to this as *conditional pseudo-entropy* $\tilde{H}$.

[11] https://musicbrainz.org/doc/Release_Group

[12] Details on the genre/subgenre taxonomies and their distributions are reported on the dataset website.

	Allmusic	Discogs	Lastfm	Tagtraum
Allmusic	59.6	39.6	28.9	33.3
Discogs	35.4	21.2	15.1	17.8
Lastfm	32.1	19.2	11.2	16.0
Tagtraum	29	17.7	11.6	10.6

(a) Genre and subgenre labels.

	Allmusic	Discogs	Lastfm	Tagtraum
Allmusic	1.94	2.40	1.62	1.49
Discogs	2.37	2.15	1.57	1.50
Lastfm	2.87	2.88	1.18	1.8
Tagtraum	2.09	2.00	1.17	0.67

(b) Only genre labels.

Table 3: Conditional pseudo-entropy $\tilde{H}(X|Y)$ between pairs of datasets, where X is the dataset in the row and Y the one in the column.

Table 3a shows the conditional pseudo-entropies when considering both genre and subgenre labels. As the diagonal shows, the AllMusic dataset is much more complex than the others, as anticipated by the high number of subgenres in the taxonomy and the smooth distribution shown in Figure 2. Interestingly, the Lastfm column shows that knowing Lastfm labels provides the most information when predicting labels in the other taxonomies, only surpassed by known labels in the target taxonomies (diagonals). Lastfm and Tagtraum are the most similar sets, with AllMusic and Discogs being the most dissimilar. This suggests that labels produced by different non-expert user communities and following no common guidelines, are more similar than those produced by different set of experts following different guidelines.

Table 3b shows similar results, but considering only the genre labels. The pseudo-entropies are orders of magnitude smaller because genres encode less information, and as a result relative differences among datasets are also smaller. Discogs is the most complex dataset because of its higher variability in the number of genres per recording (see rows in Figure 2), followed by AllMusic. This time, we see that Tagtraum provides the most information when predicting labels in another taxonomy. As before, the most similar sets are Lastfm and Tagtraum, and the most dissimilar are AllMusic and Discogs.

2.2 Music Features

We provide music features precomputed from audio for all music recordings. All features are taken from the

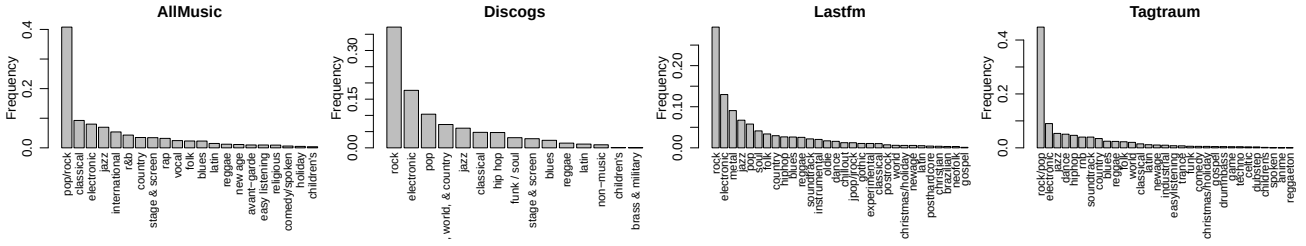


Figure 1: Distributions of genre labels.

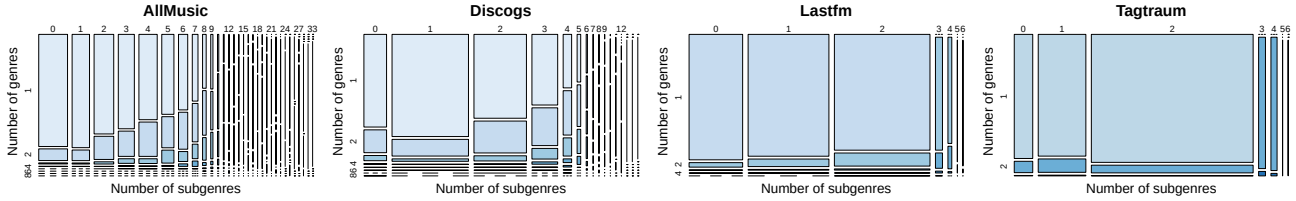


Figure 2: Distributions of label counts. Box heights represent the amount of recordings with the number of genre labels indicated in the row, and widths represent the amount of recordings with the number of subgenre labels in the column.

AcousticBrainz database and were extracted from audio using Essentia, an open-source library for music audio analysis [6]. They include features characterizing overall loudness, dynamics, and spectral shape of the signal, rhythm descriptors (including beat positions and BPM value), and tonal information (including chroma features, keys and scales). [13] Only a statistical characterization of time frames is provided (bag of features), that is, no frame-level data is available. The features for each recording are provided in a JSON file. [14]

2.3 Training, Validation and Test Sets

We provide four training sets and four validation sets with all data publicly available, and four test sets with a hidden ground truth. The training and validation sets can be used for the evaluation of MGR systems (Section 3.3). The test sets do not include a publicly available ground truth and have anonymized MBIDs; they are reserved for future MGR challenges. Nevertheless, it is possible to run an evaluation on the test sets upon request. [15]

The datasets were created by a random split of the full data ensuring that:

- No recording appears in more than one set;
- No recordings in any set are from the same release groups present in other sets (e.g., albums, singles, EPs);
- The same genre and subgenre labels are present in all three sets for the same source;
- Genre and subgenre labels are represented by at least 40 and 20 recordings from 6 and 3 release groups in training and validation/test sets, respectively.

The approximate split ratios of the datasets are 70% for training, 15% for validation, and 15% for testing. Partitioning scripts are provided to create training-validation splits ensuring these characteristics in the data. The four ground truths partially overlap. The full intersection of all training sets contains 247,716 recording, while the intersection of the two largest sets, AllMusic and Discogs, contains 831,744 recordings.

All data are published in JSON and TSV formats; details about the formats are available online. Each recording in the training and validation sets is identified by an MBID, which can be used by researchers to gather related data. Importantly, our split avoids the "album effect" [11], which leads to a potential overestimation of the performance of a system when a test set contains recordings from the same albums as the training set. We don't filter for the artist effect, in order to preserve some low-count tags and to address the fact that artists can release albums with different broad genres. MusicBrainz artist IDs allow researchers to perform this filtering if desired. The training sets additionally include information about release groups of each recording, which may be useful for researchers in order to avoid this effect when developing their systems.

3. RESEARCH TASK

MGR systems typically attempt to predict a single label per recording. Given that the AcousticBrainz Genre Dataset features multiple hierarchical labels from different sources, we suggest the following two subtasks designed for the datasets introduced in Section 2.

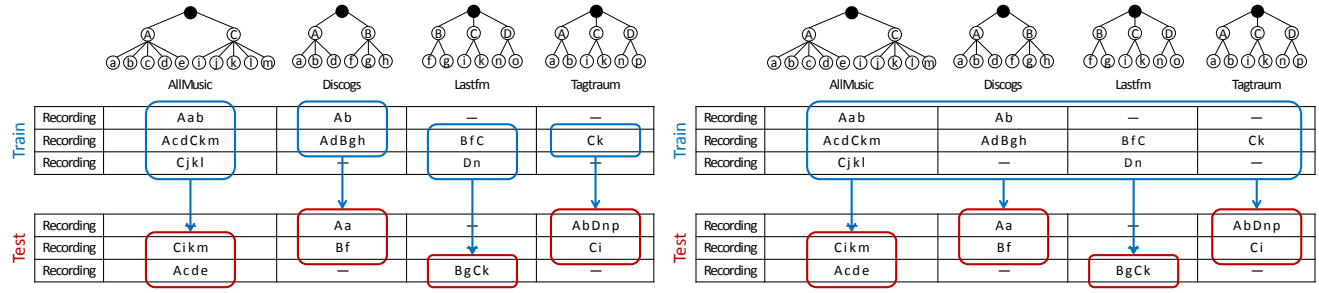
3.1 Subtask 1: Single-source Classification

This task, depicted in Figure 3a, explores conventional systems, each one trained on a single dataset. Researchers make predictions for the test set of each dataset separately, using their respective class spaces (genres and subgenres). These predictions will be produced by a separate system for each dataset, trained without any information from the

[13] More details are available online: `http://essentia.upf.edu/documentation/streaming_extractor_music.html`

[14] An example JSON file: `http://acousticbrainz.org/api/v1/6bb7e980-791c-44b5-9024-cc7c90bc8230/low-level?n=0`

[15] Please, contact the authors.



(a) Subtask 1: Single-source classification

(b) Subtask 2: Multi-source classification

Figure 3: Suggested tasks for the AcousticBrainz Genre Dataset.

other sources. This subtask can serve as a baseline for the multi-source classification task described below.

3.2 Subtask 2: Multi-source Classification

This task (Figure 3b) explores the combination of several ground-truth sources to train, but still make predictions for each test set separately, again following the corresponding genre class spaces. These predictions may be produced by a single system for all datasets or by one system for each dataset. Researchers are free to make their own decisions about how to combine the training data from all sources.

3.3 Evaluation

The development of an appropriate methodology that models each subtask as a single experiment with a "source" factor and replicated observations, is an interesting point that we leave for future research. For simplicity, we follow traditional evaluation on each test dataset separately, as if they were four independent experiments. As for metrics, we propose ROC AUC, precision, recall and F-score at the label level for a system-oriented view, and also at the recording level for a user-oriented view. We do not use hierarchical metrics because the hierarchies in the Lastfm and Tagtraum datasets are not explicit. Instead, we compute metrics at different levels:

- Per recording: using all labels, only genre labels, or only subgenre labels
- Per label: using all recordings
- Per genre label: using all recordings
- Per subgenre label: using all recordings

The ground truth does not necessarily contain subgenre annotations for some recordings, so we only considered recordings containing subgenres for the evaluation at the subgenre level. We provide evaluation scripts for development purposes and two simple baselines:

- Random baseline reproduces the joint distribution of labels as found in the training sets.
- Popularity baseline always predicts the most popular genre in the training set.

In the context of the MediaEval 2017–18 task, [16] researchers were expected to create predictions for both subtasks, reporting whether they used the entire data available for development or only its parts for every submission. Overall, we received over 100 submissions from 7 research teams covering both subtasks.

[16] Task details and evaluation results are available online: `https://multimediaeval.github.io/2018-AcousticBrainz-Genre-Task`

4. BASELINE

In this section we present our baseline approach for the proposed MGR tasks. This baseline employs an oversimplistic deep learning architecture for the single-source task and a fusion approach that demonstrates the possibilities of merging different genre ground truth sources in the multi-source task. To this end, we explore how stacking deep feature embeddings obtained on different datasets can benefit MGR systems. We propose an early fusion approach, similar to the one proposed in [19] for multi-modal genre classification. The approach incorporates knowledge across datasets by stacking deep feature embeddings learned on each dataset individually and using those as an input to predict genres for each dataset.

4.1 Input Features

We use all available features provided for the challenge. As a pre-processing step, we apply one-hot encoding for a few categorical features related to tonality (`key_key`, `key_scale`, `chords_key`, and `chords_scale`) and standardize all features (zero mean, unit variance). In total, this amounts to 2669 input features.

4.2 Neural Network Architecture

A simple feedforward network (extractor network) is used to predict the probabilities of each genre given a track. The network consists of an input layer of 2669 units (the size of the feature vector for an input recording), followed by a hidden dense layer of 256 units with ReLu activation, and the output layer where the number of units coincides with the number of genres to be predicted in each dataset. Dropout of 0.5 is applied after the input and the hidden layer. As the targeted genre classification task is multi-label, the output layer uses sigmoid activations and is evaluated with a binary cross-entropy loss.

Mini-batches of 32 items are randomly sampled from the training data to compute the gradient. The Adam [15] optimizer is used to train the models, with the suggested

Subtask	AllMusic	Discogs	Lastfm	Tagtraum
Single-source	0.648	0.759	0.828	0.802
Multi-source	0.812	0.886	0.906	0.887

Table 4: ROC AUC on validation datasets.

default parameters. The networks are trained for a maximum of 100 epochs with early stopping on validation loss. Once trained, we extract the 256-dimensional vectors from the hidden layer for the training, validation, and test sets.

The model architecture is used to train a multi-label genre classifier on each of the four datasets. The models are trained on 80% of the training set and validated after each epoch using the other 20% using the provided split script with release group filtering. Predictions are computed for the validation and test sets.

4.3 Embedding Fusion Approach

One model per dataset is trained. These models serve for predictions in Subtask 1. For Subtask 2, the given models are used as feature extractors. All four models share the same input format, so input feature vectors from one dataset can be used as input to a model trained on other datasets. For each model we feed all tracks from the training, validation and test sets of each dataset, and obtain the activations of the hidden layer as a 256-dimensional feature embedding. Therefore, for each track in each dataset we obtain four different feature embeddings, coming from each of the four previously trained models.

Given the four feature embeddings of each track, we apply the ℓ_2-norm to each of them and then stack them together into a single 1024-dimensional feature vector. We obtain new feature vectors for every track in the training, validation and test sets of each dataset. We use these feature vectors as input to a fusion network where the input layer is directly connected to the output layer. Dropout of 0.5 is applied after the input layer. The output layer is exactly the same as in the extractor network, where sigmoid activation and binary cross-entropy loss are applied. The fusion network is trained following the same methodology and partitions described for the extractor network. We train a fusion network per dataset, and obtain the genre probability predictions of the validation and test sets for Subtask 2.

4.4 Predictions Thresholding

The predictions made by each model are continuous, while the task requires binary prediction of genre labels. We apply a plug-in rule approach thresholding the prediction values to maximize the evaluation metrics. As an example, we decided to maximize the macro F-score, and applied thresholds individual for each genre label [18].

4.5 Results and Analysis

Full results and code for the baseline are available at the dataset website. Table 4 presents the ROC AUC metric on the validation sets. Table 5 presents the final results after applying thresholding. We can clearly see the benefit of models based on the embedding fusion approach compared to the models trained individually on each dataset. While the individual models (Subtask 1) are hardly usable, the combined models got a significant improvement in performance.

		Dataset			
		AllMusic	Discogs	Lastfm	Tagtraum
Single-source					
Per recording (all labels)	P	0.016	0.069	0.075	0.124
	R	0.579	0.538	0.446	0.507
	F	0.030	0.119	0.124	0.194
Per label (all labels)	P	0.023	0.076	0.074	0.097
	R	0.492	0.249	0.238	0.232
	F	0.032	0.095	0.095	0.115
Multi-source					
Per recording (all labels)	P	0.142	0.286	0.266	0.299
	R	0.475	0.545	0.476	0.513
	F	0.195	0.339	0.305	0.349
Per label (all labels)	P	0.065	0.108	0.115	0.127
	R	0.155	0.210	0.220	0.223
	F	0.074	0.122	0.133	0.140

Table 5: Precision, recall and F-scores on validation datasets produced by our baseline approach.

In our baseline, we focused on optimizing macro F-score, however choosing this metric for threshold optimization can have a negative effect on micro-averaged metrics. In the case of infrequent subgenre labels and an uninformative classifier, an optimal, but undesirable strategy may involve always predicting those labels [18]. Indeed, this was the case for the individual models, but the fusion models did not have this issue.

Overall, we may expect further improvements in performance by means of a more sophisticated network architecture (e.g., [16, 25]). The baseline is available online at the dataset webpage.

5. CONCLUSIONS

We have presented the AcousticBrainz Genre Dataset, a large-scale dataset of music features and hierarchical multi-label genre annotations from different sources. This is unique data for MIR research, as it allows researchers to explore how the same music pieces are annotated differently by different communities following their own genre taxonomies, and how this could be addressed by genre recognition systems. To this end, we have proposed a research task for building MGR systems based on music features available in the AcousticBrainz database and to explore how multiple sources of genre annotations can be combined by MGR systems. This task was already held within the MediaEval 2017–18 evaluation campaigns, and it may serve as an example of the development and validation of automatic annotation algorithms on complementary datasets with different taxonomies and coverage.

Acknowledgments
We thank all contributors to AcousticBrainz. This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No 688382 (AudioCommons) and 770376-2 (TROMPA), as well as the Ministry of Economy and Competitiveness of the Spanish Government (Reference: TIN2015-69935-P). We also thank tagtraum industries for providing the Tagtraum genre annotations.

6. REFERENCES

[1] Adam Berenzweig, Beth Logan, Daniel PW Ellis, and Brian Whitman. A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, 28(2):63–76, 2004.

[2] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *International Society for Music Information Retrieval Conference*, 2011.

[3] Dmitry Bogdanov, Alastair Porter, Julián Urbano, and Hendrik Schreiber. The MediaEval 2017 AcousticBrainz Genre Task: Content-based Music Genre Recognition from Multiple Sources. In *MediaEval Benchmark Workshop*, 2017.

[4] Dmitry Bogdanov, Alastair Porter, Julián Urbano, and Hendrik Schreiber. The MediaEval 2018 AcousticBrainz Genre Task: Content-based Music Genre Recognition from Multiple Sources. In *MediaEval Benchmark Workshop*, 2018.

[5] Dmitry Bogdanov and Xavier Serra. Quantifying music trends and facts using editorial metadata from the discogs database. In *18th International Society for Music Information Retrieval Conference*, 2017.

[6] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, Jose R. Zapata, and Xavier Serra. Essentia: An audio analysis library for music information retrieval. In *International Society for Music Information Retrieval Conference*, 2013.

[7] Pedro Cano, Emilia Gómez, Fabien Gouyon, Perfecto Herrera Boyer, Markus Koppenberger, Bee Suan Ong, Xavier Serra, Sebastian Streich, and Nicolas Wack. ISMIR 2004 audio description contest. 2006.

[8] Alastair JD Craft, Geraint A. Wiggins, and Tim Crawford. How Many Beans Make Five? The Consensus Problem in Music-Genre Classification and a New Evaluation Method for Single-Genre Categorisation Systems. In *International Society for Music Information Retrieval Conference*, 2007.

[9] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. FMA: A dataset for music analysis. In *International Society for Music Information Retrieval Conference*, 2017.

[10] Sander Dieleman, Philémon Brakel, and Benjamin Schrauwen. Audio-based music classification with a pretrained convolutional network. In *International Society for Music Information Retrieval Conference*, 2011.

[11] Arthur Flexer and Dominik Schnitzer. Album and artist effects for audio similarity at the scale of the Web. In *Sound and Music Computing Conference*, 2009.

[12] Masataka Goto. Development of the RWC music database. In *Proceedings of the International Congress on Acoustics (ICA)*, 2004.

[13] Fabien Gouyon, Anssi P. Klapuri, Simon Dixon, Miguel Alonso, George Tzanetakis, Christian Uhle, and Pedro Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1832–1844, 2006.

[14] Enric Guaus i Termens. *Audio content processing for automatic music genre classification: descriptors, databases, and classifiers*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2009.

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] Khaled Koutini, Alina Imenina, Matthias Dorfer, Alexander Rudolf Gruber, and Markus Schedl. MediaEval 2017 AcousticBrainz Genre Task: Multilayer Perceptron Approach. In *MediaEval 2017 Workshop*, Dublin, Ireland, 2017.

[17] Jin Ha Lee, Kahyun Choi, Xiao Hu, and JH Downie. K-pop genres: A cross-cultural exploration. In *International Society for Music Information Retrieval Conference*, 2013.

[18] Zachary C Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. Optimal thresholding of classifiers to maximize f1 measure. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 225–239. Springer, 2014.

[19] Sergio Oramas, Francesco Barbieri, Oriol Nieto, and Xavier Serra. Multimodal deep learning for music genre classification. *Transactions of the International Society for Music Information Retrieval*, 1(1), 2018.

[20] Sergio Oramas, Oriol Nieto, Francesco Barbieri, and Xavier Serra. Multi-label music genre classification from audio, text, and images using deep features. *International Society for Music Information Retrieval Conference*, 2017.

[21] Alastair Porter, Dmitry Bogdanov, Robert Kaye, Roman Tsukanov, and Xavier Serra. AcousticBrainz: a community platform for gathering music information obtained from audio. In *International Society for Music Information Retrieval Conference*, 2015.

[22] Alastair Porter, Dmitry Bogdanov, and Xavier Serra. Mining metadata from the web for AcousticBrainz. In *International workshop on Digital Libraries for Musicology*, 2016.

[23] Alexander Schindler, Rudolf Mayer, and Andreas Rauber. Facilitating comprehensive benchmarking experiments on the million song dataset. In *International Society for Music Information Retrieval Conference*, 2012.

[24] Hendrik Schreiber. Improving genre annotations for the Million Song Dataset. In *International Society for Music Information Retrieval Conference*, 2015.

[25] Hendrik Schreiber. MediaEval 2018 AcousticBrainz genre task: A CNN baseline relying on Mel-Features. In *Proceedings of the MediaEval 2018 Multimedia Benchmark Workshop*, Sophia Antipolis, France, 10 2018.

[26] Bob L. Sturm. A survey of evaluation in music genre recognition. In *International Workshop on Adaptive Multimedia Retrieval*, 2012.

[27] Bob L. Sturm. The State of the Art Ten Years After a State of the Art: Future Research in Music Information Retrieval. *Journal of New Music Research*, 43(2):147–172, 2014.

[28] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

DATA-DRIVEN SONG RECOGNITION ESTIMATION USING COLLECTIVE MEMORY DYNAMICS MODELS

Christos Koutlis[1] **Manos Schinas**[1] **Vasiliki Gkatziaki**[1]
Symeon Papadopoulos[1] **Yiannis Kompatsiaris**[1]
[1] CERTH-ITI, Thessaloniki, Greece
ckoutlis@iti.gr, manosetro@iti.gr, vasgat@iti.gr, papadop@iti.gr, ikom@iti.gr

ABSTRACT

Cultural products such as music tracks intend to be appreciated and recognized by a portion of the audience. However, no matter how highly recognized a song might be at the beginning of its life, its recognition will inevitably and progressively decay. The mechanism that governs this decreasing trajectory could be modelled as a forgetting curve or a collective memory decay process. Here, we propose a composite model, termed T-REC, that involves chart data, YouTube views, Spotify popularity of tracks and forgetting curve dynamics with the purpose of estimating song recognition levels. We also present a comparative study, involving state-of-the-art and baseline models based on ground truth data from a survey that we conducted regarding the recognition level of 100 songs in Sweden. Our method is found to perform best among this ensemble of models. A remarkable finding of our study pertains to the role of the number of weeks a song remains in the charts, which is found to be a major factor for the accurate estimation of the song recognition level.

1. INTRODUCTION

Music is a form of art that attracts the vast majority of global population and has a remarkable impact on people's emotions and behavior. In particular, in-store consumer purchase behavior has been related to background music in the research literature [9, 21, 22, 31]. Also, the role of music popularity, liking and recognition levels in shopping intentions [4, 35] and the perception of time [2] has been investigated. Background music providers supply companies with music playlists with the purpose of optimizing the in-store experience of their customers and their brand perception. Having effective means of estimating song recognition can provide such companies with a useful tool for generating better playlists. Motivated by the above, in this paper we propose an accurate song recognition model as basis for experimentally measuring the impact of song recognition on in-store purchase behavior. [1]

Although song *recognition* is the focus of this paper, song *popularity* is more frequently encountered in the research literature. The popularity of songs is a concept used to express how much attention a certain song *currently* receives. There have been attempts towards determining song popularity making use of the online available information from posts in microblog websites [12, 23, 28, 29] and in the blogosphere [1], search queries and number of shared files in peer-to-peer networks [17, 28], play counts in social media music sites such as Last.fm [3, 29], the amount of time of radio play, the music industry awards that it received [25] and popularity indices provided by streaming platforms such as Spotify [3]. Of course the traditional ways of determining music popularity such as the Billboard Magazine chart are also used for comparison with the modern web-based popularity indices [16, 17].

Music exhibits its own complex dynamics in terms of popularity growth and decay while the means used to promote it in the public constantly evolve and capitalize on the advances and trends of digital media and communication technologies. During the last decade, researchers have investigated special attributes of songs that may lead to a successful release [8, 15, 36], and have attempted to successfully predict hit songs [6, 16]. However, while song popularity is a research topic that has attracted intense academic interest the level of a music track's recognition is a notion that has been significantly less studied.

As song recognition we define the fraction of the audience that recognizes (comprehend that they have heard it before) a specific music track through audio exposure. This notion is different from the notion of song popularity as a song might no longer be trending (for instance an old song no longer placed in the charts) but at the same time a considerable portion of the music audience might recognize its tune. To further illustrate this differentiation, in Table 1 we present the most popular songs of 2018 [2] and most recognized songs of all time [3] . It is apparent that songs on the left column currently overwhelm the charts and online playlists, but songs on the right column are

[1] We consider atmospherics, such as high/low recognition music, as causal factors for consumer behavior in stores.

[2] According to Spotify's "Top Tracks of 2018" list.

[3] According to experimental results obtained by "Hooked On Music" (http://www.hookedonmusic.org.uk/) and published on BBC's webpage (https://www.bbc.com/news/science-environment-29847739).

Top 2018 songs	Top recognized songs
God's plan/Drake	Wannabe/Spice Girls
SAD!/XXXTENTACION	Mambo No 5/Lou Bega
rockstar/Post Malone	Eye Of The Tiger/Survivor
Psycho/Post Malone	Just Dance/Lady Gaga
In My Feelings/Drake	SOS/ABBA
Better Now/Post Malone	Pretty Woman/Roy Orbison
I like It/Cardi B	Beat It/Michael Jackson
One Kiss/Calvin Harris	I Will Always Love You/Whitney Houston
IDGAF/Dua Lipa	Don't You Want Me/The Human League
FRIENDS/Marshmello	I Don't Want To Miss A Thing/Aerosmith

Table 1: Left column: top 2018 songs in terms of popularity. Right column: top recognized songs of all time.

surely recognized by a very high percentage of the population even though they are not currently popular.

To estimate the portion of the audience that recognize a music track, one should take into account the cognitive aspects of the problem. That is to say, collective memory dynamics and more precisely the mechanisms that govern its initial increase and its decay after the initial period of popularity. For the more general concept of human memory decay many studies have been conducted [10,11,20,24] indicating forgetting curves with exponential decay dynamics. Music-specific research has adopted exponential forgetting curves [7, 14] for song "freshness" assessment, as well. A method with double exponential dynamics was proposed as a general memory decay model [5], while log-normal dynamics were employed to model the dynamics of scientific paper impact [33]. Equally important is to take into account the notions of learning curves [26] and over-learning [27] in order to determine the initial amount of learned information and the velocity of forgetting. In many studies researchers argue for the significant role of (i) repetition of a stimulus in learning [10, 13, 19, 26, 27] and (ii) the degree of original learning in the velocity of forgetting [10, 20, 30, 34]. Namely, the more exposed to a stimulus humans are the higher the initial amount of learned information is and the slower they forget it.

Here, we propose T-REC, a song recognition model that takes as input the chart positions a track has gained along with the respective dates, its current YouTube views and Spotify popularity. T-REC also considers sigmoid learning curve dynamics, exponential decay forgetting curve dynamics and a decay rate being a function of the number of weeks each track is maintained in the charts, which we consider here as a proxy of the original learning degree. In other words, the number-of-weeks feature is an indicator of how strongly the audience is exposed to a specific tune and as it increases, the forgetting procedure (i) starts from a higher point and (ii) decelerates further, as indicated by the related research literature on human memory. Eventually, T-REC results in the estimation of song recognition levels per market and globally. Other competitive models are also considered for comparison purposes. We have conducted a survey for the estimation of the current level of recognition of 100 songs in Sweden, which we then use as ground truth for the evaluation of our method's performance and for comparison with the other methods. We make the resulting data available for the community [18].

2. MATERIALS AND METHODS

2.1 Data

For estimating the recognition levels of music tracks, our starting point was a list of tracks provided by *Soundtrack Your Brand*, a collaborating background music provider. The list consists of 39,466 tracks from 21,450 artists and from 75 countries. We also make use of data from 211 charts, 198 track charts and 13 singles charts, that span long periods of time (in some cases from the 60s until today) from 62 countries around the globe including Sweden. We also used the Spotify API to annotate chart entries with the Spotify id and International Standard Recording Codes (ISRC) [4] of each of the songs. Since our user study was carried out on a Swedish population, we present the monitored charts for Sweden along with the corresponding monitored periods in Table 2.

chart name	since	until
Spotify Daily Chart	2017-01-01*	2018-03-06
Spotify Weekly Chart	2016-12-23	2018-03-01
Veckolista Svenskt Topp-20	2015-01-17	2018-06-15
Veckolista Singlar	1988-01-16	2018-06-15
Veckolista Heatseeker	2015-01-10	2018-06-15
Veckolista Svenska Singlar	2015-01-10	2015-01-16
SINGLES TOP 100	1975-11-08	2018-06-01
Sweden Top 20	2001-06-12	2018-07-07
Sweden Singles Top 100	2017-12-29	2018-07-05

Table 2: The list of Swedish charts we used in this study. The first column presents the chart name, the second and third columns present the start and end dates of monitoring respectively. *All dates are in YY-MM-DD format.

Most songs do not make it in the charts, thus we additionally employ YouTube views and Spotify popularity of tracks as current track popularity proxies. Knowing the Spotify id of the tracks and the id of an associated official video in YouTube [5], we retrieved these two signals by using the public APIs offered by Spotify and YouTube respectively. The intuition behind the use of these two metrics, is that they reflect the exposure of a song in two widely used platforms. Number of video views in YouTube is a direct measure of how many people heard a song. On the other hand, although Spotify popularity is a score generated internally by Spotify and the exact formula is not known, that score reflects the actual number of streams a song received recently. Therefore, we can safely assume that a song having a high popularity score is currently listened more than songs with a lower score.

2.2 Models

2.2.1 T-REC

The proposed song recognition model builds upon three main components, the *recognition growth* that represents the level of recognition a track reaches during its initial

[4] https://isrc.ifpi.org/

[5] To this end, we used the Soundiiz (www.soundiiz.com) service, which supports playlist conversion between platforms.

prosperity time (when it is placed in charts), the *recognition decay* that represents the collective memory decay process (i.e. the mechanism of collective forgetting of songs) and the *recognition proxy-based adjustment* that adjusts the recognition level of tracks, which is especially useful for tracks with no chart information.

Having annotated chart entries with the corresponding ISRC, we were able to retrieve the positions of tracks in the Swedish charts of Table 2. These are then used to estimate their *recognition growth* (in Sweden) according to Equation 1:

$$g(t) = \begin{cases} 100 \cdot \frac{c_K + 1 - r_K(t)}{c_K} \cdot \sigma_1, \\ \qquad\qquad \text{if track in chart } K \text{ at time } t \\ 0, \text{ otherwise} \end{cases} \tag{1}$$

where c_K is the number of tracks in chart K, $r_K(t)$ is the rank of the track in chart K at time $t \in [t_0, t_{today}]$ [6] and $\sigma_1 = \sigma_1(\theta_0, \theta_1, x) = (1 + e^{-\theta_1 \cdot x + \theta_0})^{-1}$ adjusts the rank's importance using an S-shaped learning curve with $x \in (0, +\infty)$ and $\theta_0, \theta_1 \in \mathbb{R}$. The logistic part of the model is incorporated to control the importance of a chart position given the number of weeks x the track has remained in the charts. If a track remained in the charts for only one week its rank's importance would be a lot lower (54.9%) compared to it remaining for 20 weeks (98.2%). Therefore, in the first case the decay process will begin from a much lower point. The value of $g(t)$ is assigned to all $g(t_i)$ with $t_i \in [t - n + 1, t]$ according to the chart's frequency e.g. if it is weekly $n = 7$. If a track gains multiple values at a single date, the maximum value is used.

Towards formally defining the *recognition decay*, we build on findings from research literature in the area of human memory, and more precisely on the concept of forgetting curves. A forgetting curve is the rule by which the memory regarding a specific learned item is reduced. In our case we consider as learned items the music tracks. Hence, we aim at estimating the function that describes the forgetting procedure that has been proposed to be exponentially decreasing in many studies [20, 24]. We also opt for the exponentially decreasing forgetting curve for song recognition but at a less steep rate. It is natural to consider that the level of *recognition decay* is impossible to be higher than the level of *recognition growth* at its peak for a particular track. Also, each time the track reemerges in the charts the forgetting procedure restarts from a new higher point of recognition. Additionally, we consider a variable decay rate as a reasonable consideration would be that not all songs' recognition decays with the same velocity. The *recognition decay* is defined in Equation 2:

$$d(t) = \begin{cases} g(t), \text{ if } d(t-1) \leq g(t) \\ \sigma_2 \cdot d(t-1) + (1 - \sigma_2) \cdot g(t), \text{ otherwise} \end{cases} \tag{2}$$

where $\sigma_2 = \sigma_2(\phi_0, \phi_1, x) = (1 + e^{-\phi_1 \cdot x + \phi_0})^{-1}$ is the recognition retention percentage with x being the number of weeks the track has remained in the charts and $g(t)$ is the previously defined in Equation 1 *recognition growth*. If a track has remained for a long time in the charts its retention percentage would be considerably high and its forgetting process would be rather slow, while if a track has remained in the charts for only few weeks its retention percentage would be low and its forgetting process fast.

The first logistic function σ_1 controls the initial recognition level from which the decreasing trajectory begins and the second logistic function σ_2 controls the velocity of recognition decay, both individually per track.

To model the *recognition proxy-based adjustment*, we consider a multiple linear regression model with input the track's current Spotify popularity index (P_S) and the log-transformed YouTube views (P_{YT}) as in Equation 3:

$$s(t_{today}) = \alpha_0 + \alpha_1 \cdot log(P_{YT}) + \alpha_2 \cdot P_S \tag{3}$$

The composite T-REC model is defined as a linear combination of *recognition decay* and *recognition proxy-based adjustment* at t_{today}:

$$\text{T-REC} = w_0 + w_1 \cdot d(t_{today}) + w_2 \cdot s(t_{today}) \tag{4}$$

2.2.2 Competitive Models

In order to perform a comparative study, four competitive models are employed for the task of song recognition estimation. Two of them are well-known regression models, one is related to collective memory decay while the last one is the plain Spotify popularity index (P_S).

The first model is based on Multiple Linear Regression (MLR) and the second on Random Forests (RF). The log-transformed YouTube views and the Spotify track popularity are considered as inputs to these models and actual recognition as their target. MLR actually corresponds to the proxy-based adjustment introduced in Equation 3. The third competitive model is the state-of-the-art log-normal decay model (LOGN) [33] that Wang et al. developed for modelling the decay process of scientific paper citations. We use the form of this model that is presented in the supplementary material of [5], namely Equation 5:

$$l(t) = e^{\left[ln\left(\frac{\lambda}{\sqrt{2\pi}\sigma}(c^t + m)\right) - \mu^2\right]} \cdot t^{\frac{\mu}{\sigma^2} - 1} \cdot e^{-\frac{ln^2(t)}{2\sigma^2}} \tag{5}$$

where t is time, $c^t = m\left[e^{\lambda\Phi\left(\frac{ln(t)-\mu}{\sigma}\right)} - 1\right]$, $\Phi(\cdot)$ is the cumulative distribution function of the normal distribution and λ, μ, σ, m are arbitrary parameters.

2.2.3 Optimization and Evaluation

We consider a holdout strategy (70% training, 30% test) for the models' evaluation as described in [32]. The optimization of all models' parameters is performed in the training set by the truncated Newton algorithm as implemented by the SciPy package. We use as objective function the mean absolute error between measured (by the user study) and computed (by each model) song recognition.

[6] As t_0 we set the song's release date and as t_{today} the current date.

	class	#	fraction	Sweden
gender	male	521	50.05%	50.24%
	female	520	49.95%	49.76%
age	18-24	54	5.19%	18.15%*
	25-34	422	40.54%	22.46%
	35-44	277	26.61%	20.01%
	45-54	244	23.44%	21.12%
	55-65	44	4.22%	18.26%

Table 3: Demographics of the test population and Sweden (normalized within the group of people between 15 and 65 years old). *This figure refers to 15-24 age group.

The models' performance is then evaluated in the test set by the mean absolute error (MAE) between the measured and the computed recognition as in Equation 6:

$$\text{MAE} = \frac{\sum_{i=1}^{k} \mid x_i - y_i \mid}{k} \quad (6)$$

where k is the number of tracks, x_i is the measured recognition for track i and y_i is the computed recognition for track i. A perfectly accurate model would lead to a MAE value of 0.

2.3 User study

To proceed with the user study, we employed an initial and much simpler version of the recognition score. This initial version had a constant decay rate across all tracks and for the tracks with no chart data the average recognition score of the closest, in terms of YouTube views and Spotify popularity, tracks was considered as their recognition score. [7]

After the assignment of the initial recognition score (corresponding to the time the survey was conducted) to each of the 39,466 tracks, we formed two lists. One list containing the 600 most recognized tracks in Sweden and a second containing the 600 least recognized tracks in Sweden. [8] Consequently, 50 tracks were randomly chosen out of each of these two lists as representative of high and low recognition tracks.

A study was then conducted in order to obtain the actual recognition percentages for each of these 100 songs among a test population of 1041 annotators in Sweden. [9] We divided the initial list of 100 songs in 10 groups of 10 songs (5 of low and 5 of high recognition level in a randomized order), then each participant listened to 30-second samples of all the songs of one group and for each song he/she indicated whether he/she recognized it or not. We had ∼100 respondents per song [10] so we got a score 0-100 based on the percentage of respondents who responded positively.

[7] The rationale behind the alterations on this model that led to T-REC is illustrated in the results section.

[8] Given that recognition estimation is the result of a sampling process, we expect measurements in the extremes (i.e. least and most recognised songs) to be less noisy than in intermediate recognition levels. This motivated our choice to perform the initial song selection out of two distinct sets (high, low).

[9] The study was performed through the Cint survey platform (`https://www.cint.com/`).

[10] Some variability was due to the fact that not all respondents completed the process successfully.

θ_0	θ_1	ϕ_0	ϕ_1	α_0
0.233	0.043	0.847	0.029	1.299
α_1	α_2	w_0	w_1	w_2
0.999	-0.093	22.586	0.452	0.928

Table 4: Parameter values for T-REC after fitting.

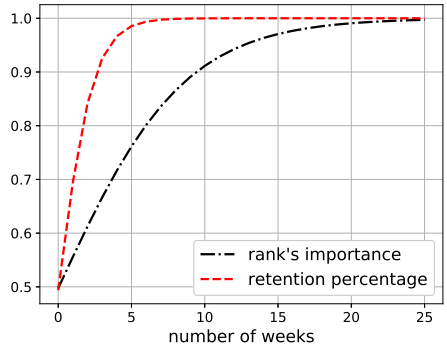


Figure 1: The logistic parts of *recognition growth* (rank's importance) and *recognition decay* (retention percentage) components as formed after the model fitting.

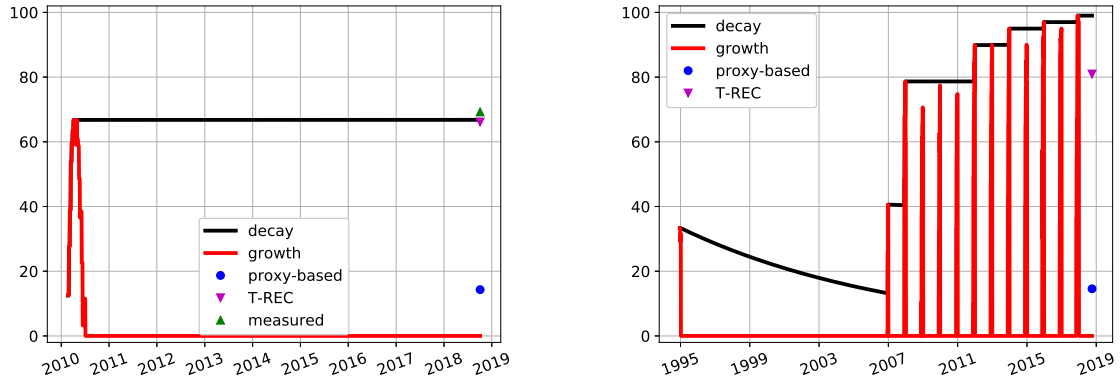
We consider the recorded responses as ground truth for our experiments and we evaluate our model as well as the competitive models on this basis. Demographics of the test and Swedish population [11] are illustrated in Table 3. We observe divergent age demographics, yet almost identical gender demographics between the test and actual population. As the selection of annotators was carried out by Cint, we could not better approximate the Swedish population distribution. Despite the over-representation of some age groups and under-representation of others, T-REC is still a sound methodology; given a different population sample to learn from, the model tuning step (section 2.2.3) would lead to a slightly different recognition estimation model.

3. RESULTS

The analysis of the survey data shows that the initial recognition score classified the tracks effectively with 50/50 (100%) correctly labeled as low and 37/50 (74%) correctly labeled as high recognition (measured recognition $<$50% is considered as low, while $>$50% as high). The 13 songs that were falsely classified as high recognition actually obtain a smaller recognition score than the rest (on average 6 units lower). Despite the promising classification performance the measured recognition was in many cases far from the computed score especially in cases of tracks with no chart data. Thus, we developed the updated version of the recognition score (T-REC) described in section 2.2.1.

Table 4 presents the parameter values of T-REC after optimization. The model gives significant weights on both *recognition decay* (w_1) and *recognition proxy-based adjustment* (w_2) components, but considers YouTube views

[11] Sources: statista.com/statistics/521717/sweden-population-by-age/, statista.com/statistics/521540/sweden-population-by-gender/



(a) Rude Boy - Rihanna

(b) All I Want for Christmas Is You - Mariah Carey

Figure 2: T-REC components (*recognition growth*, *decay* and *proxy-based adjustment*) for two highly recognized songs.

(α_1) as more important than Spotify popularity (α_2) for the under study problem. The impact of the rest of the parameters on the final model, namely the shape of the two logistic functions that control the *recognition growth* (θ_0, θ_1) and *recognition decay* (ϕ_0, ϕ_1) components is illustrated in Figure 1. The logistic part of *recognition growth* (rank's importance) is less steep than the logistic part of *recognition decay* (retention percentage), indicating that a music track will need almost 7 weeks in the charts to achieve a very slow rate towards oblivion, but at least 25 weeks to achieve its highest contemporary recognition.

Moreover, two examples of how T-REC models the mechanism of song recognition decay are illustrated in Figure 2. As illustrated in Figure 2a, the song "Rude Boy" by Rihanna stayed in Swedish charts for 19 weeks, and according to the *recognition decay* component it maintained 99.9% of its initial recognition. Consequently, the *recognition proxy-based adjustment* input adjusts T-REC very close to the measured recognition (error=3.11). A different example presented in Figure 2b shows that Mariah Carey's "All I Want for Christmas Is You" initially was not a big hit in Sweden, remaining for only three weeks in the charts back in 1995. Afterwards, its recognition exhibited a significant decrease during the next decade, but after 2007 when the song kept reemerging in the charts every year its recognition decay rate slowed down and both its *recognition growth* and *decay* components grew larger. The *recognition proxy-based adjustment* component adjusts T-REC a little lower. Although we lack ground truth for this song to compare it to T-REC's estimation (as it was not in the survey's lists), we believe that 80.99% recognition is closer to the real recognition rate [12] than the 98.99% computed by the *recognition decay* component, which is obviously too high even for a massive hit such as this.

Figure 3 illustrates the performance of T-REC on estimating the actual current recognition level of songs in Sweden. Most of the points are concentrated close to the identity line except for some tracks of intermediate recognition levels, which are overestimated. Table 5 compares the performance of T-REC with the one of all competitive models in terms of average MAE. All models (except for Spotify popularity index) are trained on 100 different training sets, each containing 70 randomly selected tracks out of the initial set of 100 tracks and then their MAE is measured on the 100 corresponding test sets, each containing the remaining 30 tracks. T-REC exhibits the best performance among all models with a very high statistical significance level as indicated by the p-value=10^{-17}, according to the Wilcoxon signed rank paired test. [13]

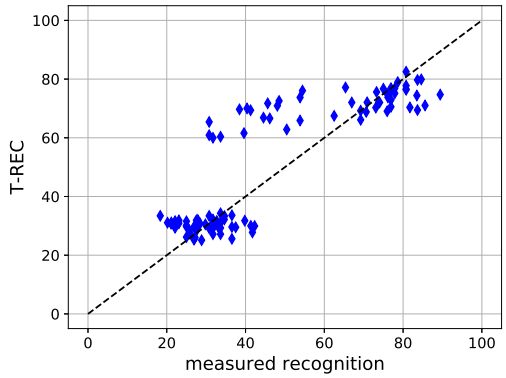


Figure 3: Scatter plot with the T-REC recognition score on the y axis and measured recognition on the x axis.

As additional information we provide long lists of Top-100 recognized songs and the corresponding Top-10 artists (according to T-REC) for Sweden and USA, in the supplementary material. Furthermore, we compare T-REC's Top-N lists with Billboard's "The Hot 100's All-Time Top 100 Songs" list.The results show that T-REC assigns top scores to most of these songs as well. This fact also holds (but to a lower degree) when the chart data from "The Hot 100" are omitted from the input list of charts.

The diverging behaviour of the songs with intermediate recognition level in Figure 3 is also apparent in YouTube

[12] Or only slightly underestimating it given that the top-3 measured recognition percentages of our survey are 89.42, 85.57 and 84.61.

[13] This is the maximum p-value among all four comparisons.

model	AMAE
P_S	20.63
MLR (Equation 3)	11.30
RF	10.27
LOGN (Wang et al. 2013 [33])	22.00
T-REC	**8.50**

Table 5: Average MAE over 100 randomly selected test sets for T-REC, Spotify popularity (P_S), Multiple Linear Regression (MLR), Random Forest (RF) and log-normal (LOGN) models. For Spotify popularity we computed once the mean absolute error over all 100 tracks.

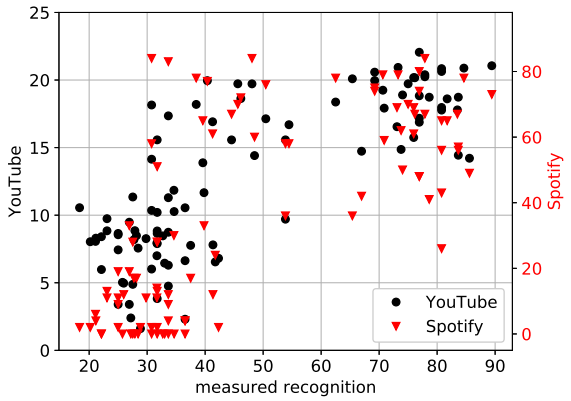


Figure 4: Scatter plot - y axes: YouTube views (log) and Spotify popularity, x axis: measured recognition.

and Spotify indices as shown in Figure 4. One possible explanation for this behavior is that 15 out of these 17 songs have been released during the last three years and they still are in their initial popularity phase. Thus, there has not passed a considerable amount of time in order for these tracks to experience significant recognition decay, which T-REC would likely capture. As exemplified in Figure 5 the more recent the track the bigger the error our model produces, which is a limitation of the proposed model, even though the average errors in all periods are small (the maximum is 9.5) and less than any other compared model.

Finally, we would like to elaborate on the rationale behind the refinements that we performed on our model in order to take its final form (Equation 4). In Figure 4 a linear interaction is observed between (i) the log-transformed YouTube views and Spotify popularity and (ii) the measured recognition, with Pearson correlation coefficients 0.79 and 0.71 respectively. Thus, we incorporated the multiple linear model with the corresponding input quantities as *recognition proxy-based adjustment* component in the final T-REC formula. The consideration of a constant decay rate in the formula of *recognition decay* is not plausible, since it would further lead to a zero rate as the best choice after model fitting, which is highly unrealistic, as the model would degenerate into the *recognition growth* component. As a result, the final form of T-REC includes a variable decay rate across music tracks that depends on

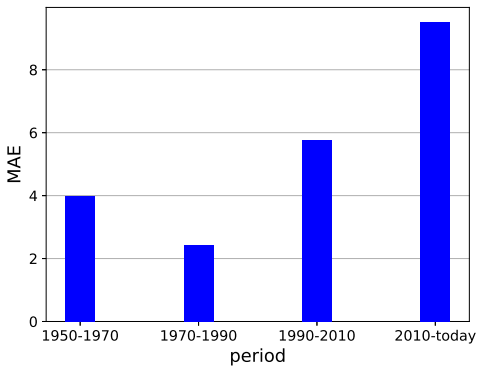


Figure 5: Mean absolute error of T-REC on tracks released in different periods of time.

the number of weeks the track has remained in the charts. This refinement resulted in significantly lower errors showcasing the major role of the number-of-weeks feature in song recognition estimation. More specifically, the initial recognition score achieved a MAE of 12.32, while T-REC a much lower MAE of 8.50 as presented in Table 5.

4. CONCLUSIONS

In this work, we studied collective memory dynamics with regards to song recognition. We proposed a model for the approximation of the corresponding decreasing trajectory and the estimation of the current song recognition level. Our recognition model comprises three main components: a) growth, b) decay, and c) proxy-based adjustment and it leverages chart data, YouTube views, Spotify popularity and forgetting curve dynamics. Also, our method considers different recognition decay rates and initial recognition levels per song, according to the number of weeks the song has remained in the charts.

We compared our model to other state-of-the art and baseline models on the task of accurately estimating the current recognition level of songs. To this end, we conducted a study in Sweden in order to measure the recognition level of 100 songs, which we then used as ground truth for the models' evaluation. The experimental results showed that our method exhibits great performance on this task, much better than the competitive models with a high statistical significance level.

Finally, we reached two remarkable conclusions:

1. according to our model's parameters, a music song needs almost 7 weeks in the charts to achieve a very slow velocity towards oblivion and at least 25 weeks to achieve its highest contemporary recognition;
2. the role of the number-of-weeks feature incorporated in our model through the logistic functions is found to be of utmost importance for the accurate estimation of a song's recognition level.

Future work will include extensions that alleviate the deviation of recent tracks' recognition estimation and also account for demographic-specific estimations.

5. ACKNOWLEDGMENTS

This work is funded by the European Commission under the contract number H2020-761634 FuturePulse. We would also like to thank our collaborators from Soundtrack Your Brand, Magnus Rydén, Jasmine Moradi and Daniel Johansson, who contributed to the problem formulation, provided the initial list of tracks, and carried out the crowd sourcing experiment to measure the recognition rates that were used as "ground truth" in our experiments.

6. REFERENCES

[1] F. Abel, E. Diaz-Aviles, N. Henze, D. Krause, and P. Siehndel. Analyzing the blogosphere for predicting the success of music and movie products. In *2010 International Conference on Advances in Social Networks Analysis and Mining*, pages 276–280, 2010.

[2] N. Bailey and C. S. Areni. When a few minutes sound like a lifetime: Does atmospheric music expand or contract perceived time? *Journal of Retailing*, 82(3):189–202, 2006.

[3] A. Bellogín, A. P. de Vries, and J. He. Artist popularity: Do web and social music services agree? In *Proc. of the 7th International AAAI Conference on Weblogs and Social Media*, pages 673–676, 2013.

[4] G. Broekemier, R. Marquardt, and J. Gentry. An exploration of happy/sad and liked/disliked music effects on shopping intentions in a women's clothing store service setting. *Journal of Services Marketing*, 22:59–67, 2008.

[5] C. Candia, C. Jara-Figueroa, C. Rodriguez-Sickert, A. L. Barabási, and C. A. Hidalgo. The universal decay of collective memory and attention. *Nature Human Behaviour*, 3(1):82–91, 2019.

[6] C. Chiru and O. G. Popescu. Automatically determining the popularity of a song. In *International Joint Conference on Rough Sets (IJCRS 2017)*, pages 392–406, 2017.

[7] A. Chmiel and E. Schubert. Using psychological principles of memory storage and preference to improve music recommendation systems. *Leonardo Music Journal*, 28:77–81, 2018.

[8] P. François and R. Pierre. Hit song science is not yet a science. In *Proc. of ISMIR 2008*, pages 355–360, 2008.

[9] S. O. Daunfeldt, N. Rudholm, and H. Sporre. Effects of brand-fit music on consumer behavior: A field experiment. *HUI Working Papers 121, HUI Research*, 2017.

[10] H. Ebbinghaus. *Memory: A Contribution to Experimental Psychology*. Teachers College, Columbia University, New York, 1913.

[11] G. Elliott, C. L. Isaac, and N. Muhlert. Measuring forgetting: a critical review of accelerated long-term forgetting studies. *Cortex*, 54(100):16–32, 2014.

[12] J. Grace, D. Gruhl, K. Haas, M. Nagarajan, C. Robson, and N. Sahoo. Artist ranking through analysis of on-line community comments. In *Proc. of the 17th International World Wide Web Conference*, 2008.

[13] A. Heathcote, S. Brown, and D. J. K. Mewhort. The power law repealed: The case for an exponential law of practice. *Psychonomic Bulletin & Review*, 7(2):185–207, 2000.

[14] Y. Hu and M. Ogihara. Nextone player: A music recommendation system based on user behavior. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011*, pages 103–108, 2011.

[15] M. Interiano, K. Kazemi, L. Wang, J. Yang, Z. Yu, and N. L. Komarova. Musical trends and predictability of success in contemporary songs in and out of the top charts. *Royal Society Open Science*, 5(5):171274, 2018.

[16] Y. Kim, B. Suh, and K. Lee. #nowplaying the future Billboard: Mining music listening behaviors of Twitter users for hit song prediction. In *SoMeRA@SIGIR*, pages 51–56, 2014.

[17] N. Koenigstein and Y. Shavitt. Song ranking based on piracy in peer-to-peer networks. In *Proc. of ISMIR 2009*, pages 633–638, 2009.

[18] Christos Koutlis, Manos Schinas, Vasiliki Gkatziaki, Symeon Papadopoulos, and Yiannis Kompatsiaris. T-rec song recognition dataset. `https://doi.org/10.5281/zenodo.3255311`, June 2019.

[19] N. Leibowitz, B. Baum, G. Enden, and A. Karniel. The exponential learning equation as a function of successful trials results in sigmoid performance. *Journal of Mathematical Psychology*, 54(3):338–340, 2010.

[20] G. R. Loftus. Evaluating forgetting curves. *Journal of Experimental Psychology*, 11(2):397–406, 1985.

[21] E. Manoppo, D. P. E. Saerang, and L. Lambey. A comparative study of males and females buyingbehavior toward music playing in store in manado. *Jurnal Berkala Ilmiah Efisiensi*, 16(3):249–257, 2016.

[22] I. L. Menajang. Comparative study of consumer buying behavior at music playing and non music playingshoe stores. *Jurnal EMBA*, 2(3):1321–1328, 2014.

[23] C. Mesnage, R. Santos-Rodriguez, M. McVicar, and T. De Bie. Trend extraction on Twitter time series for music discovery. In *Workshop on Machine Learning for Music Discovery, 32nd International Conference on Machine Learning*, 2015.

[24] J. M. J. Murre and J. Dros. Replication and analysis of Ebbinghaus' forgetting curve. *PLOS ONE*, 10(7):e0120644, 2015.

[25] J. Ren, J. Shen, and R. J. Kauffman. What makes a music track popular in online social networks? In *Proc. of the 25th International Conference Companion on World Wide Web*, pages 95–96, 2016.

[26] F. E. Ritter and L. J. Schooler. The learning curve. In *International encyclopedia of the social and behavioral sciences*, pages 8602–8605. Elsevier Science Ltd, Oxford, 2001.

[27] D. Rohrer, K. Taylor, H. Pashler, J. T. Wixted, and N. J. Cepeda. The effect of overlearning on long-term retention. *Applied Cognitive Psychology*, 19(3):361–374, 2005.

[28] M. Schedl. Analyzing the potential of microblogs for spatio-temporal popularity estimation of music artists. In *Proc. of IJCAI 2011*, 2011.

[29] M. Schedl, T. Pohle, N. Koenigstein, and P. Knees. What's hot? Estimating country-specific artist popularity. In *Proc. of ISMIR 2010*, pages 117–122, 2010.

[30] N. J. Slamecka and L. T. Katsaiti. Normal forgetting of verbal lists as a function of prior testing. *Journal of Experimental Psychology*, 14(4):716–727, 1988.

[31] R. K. Srivastava. Impact of musical fits and image of different malls on consumer purchase behaviour. *International Journal of Indian Culture and Business Management*, 12(3):318–341, 2016.

[32] I. Mozetic V. Cerqueira, L. Torgo. Evaluating time series forecasting models: An empirical study on performance estimation methods. *arXiv:1905.11744*, 2019.

[33] D. Wang, C. Song, and A. L. Barabási. Quantifying long-term scientific impact. *Science*, 342(6154):127–132, 2013.

[34] J. T. Wixted. Analyzing the empirical course of forgetting. *Journal of Experimental Psychology*, 16(5):927–935, 1990.

[35] R. Yalch and E. Spangenberg. Effects of store music on shopping behavior. *The Journal of Consumer Marketing*, 7(2):55–63, 1990.

[36] L. C. Yang, S. Y. Chou, J. Y. Liu, Y. H. Yang, and Y. A. Chen. Revisiting the problem of audio-based hit song prediction using convolutional neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 621–625, 2017.

TOWARDS INTERPRETABLE POLYPHONIC TRANSCRIPTION WITH INVERTIBLE NEURAL NETWORKS

Rainer Kelz[1], **Gerhard Widmer**[1,2]
Austrian Research Institute for Artificial Intelligence (OFAI), Austria
Institute of Computational Perception, Johannes Kepler University Linz, Austria
rainer.kelz@ofai.at

ABSTRACT

We explore a novel way of conceptualising the task of polyphonic music transcription, using so-called invertible neural networks. Invertible models unify both discriminative and generative aspects in one function, sharing one set of parameters. Introducing invertibility enables the practitioner to directly inspect what the discriminative model has learned, and exactly determine which inputs lead to which outputs. For the task of transcribing polyphonic audio into symbolic form, these models may be especially useful as they allow us to observe, for instance, to what extent the concept of single notes could be learned from a corpus of polyphonic music alone (which has been identified as a serious problem in recent research). This is an entirely new approach to audio transcription, which first of all necessitates some groundwork. In this paper, we begin by looking at the simplest possible invertible transcription model, and then thoroughly investigate its properties. Finally, we will take first steps towards a more sophisticated and capable version. We use the task of piano transcription, and specifically the MAPS dataset, as a basis for these investigations.

1. INTRODUCTION

For practitioners who apply deep neural network models to music information retrieval tasks, interpretability of predictions is of great interest. Knowing what the model was able to learn from the data, and examining the underlying causes for a prediction increases trust in the model. Being aware of the reasons for a classification result allows us to discover whether the model has learned rules that would pass a basic sanity check with a domain expert, or if it has picked up on seemingly irrelevant factors present in the data, which made it possible for the network to solve the task in a different, unexpected, possibly unwanted way [31]. There are quite a few ways to obtain an explanation from a neural network. Several methods use the gradient of an output with respect to the input as a starting point, such as [28, 32]. There are also methods that aim to provide model agnostic explanations, for instance [24, 27], and a specialization of one of the aforementioned methods to MIR systems [23] in particular.

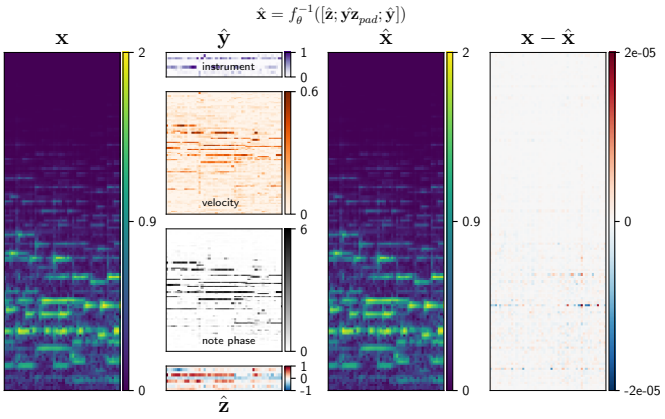


Figure 1: Computing the framewise transcription $\hat{\mathbf{y}}$ and nuisance variables $\hat{\mathbf{z}}$ from spectrogram input $\mathbf{x}$. The predictions $[\hat{\mathbf{y}}; \hat{\mathbf{y}}\mathbf{z}_{pad}; \hat{\mathbf{z}}]$ are then used to exactly reproduce $\hat{\mathbf{x}}$. The elementwise difference $\mathbf{x} - \hat{\mathbf{x}}$ is negligible. An in-depth discussion of this figure is deferred until Section 4.2.

Beyond providing explanations for predictions, a model should ideally be able to provide an answer to the question "What do you consider representative examples for a concept of interest?". Taking first steps towards producing models that are able to derive semantic information from the input, *and* are able to answer this question, we explore invertible neural networks (INNs) with respect to interpretability of predictions, their potential to identify biases and confounding factors inherent in the training dataset, and ability to generate samples for a semantic concept of interest.

Additionally, we consider ways in which these models could enable us to locate ambiguous or uncertain predictions on unlabeled data, to provide eventual users of the MIR model with hints on where manual postprocessing of the predictions might be advisable. We choose to conduct our investigation in the context of polyphonic piano transcription and provide a first glimpse at the capabilities of INNs in Figure 1. The input $\mathbf{x}$ to the INN is a magnitude spectrogram of an excerpt from a polyphonic piano piece, the output is split into a semantic part $\mathbf{y}$ containing variables of interest, and a nuisance part $\mathbf{z}$, optimistically containing all other factors of variation that are irrelevant for the MIR task the model was trained for. From these two output vectors, a hypothetical, perfectly converged in-

vertible model can faithfully reproduce the input, down to a negligible numerical difference.

Invertible neural networks are parametrized, nonlinear and bijective functions, trainable from matched pairs, similar to any other neural network in a supervised learning task. The architectures we consider here are all constructed in such a way that the inverse is available in closed form.

Networks designed in this fashion have a few desirable properties. They are both discriminative and generative models unified in one function, sharing one set of parameters. To put this into context, training a transcription system also yields a synthesizer, and vice versa training a synthesizer yields a transcription system. The term "synthesizer" is used rather loosely here, as the transcription system is trained with magnitude spectrograms.

This setup enables a direct interpretation of predictions by looking at what samples the model produces, conditioned on the predictions. This can potentially be extended until after eventual postprocessing steps, to see whether the generated samples are still close to the input in data space.

Furthermore, in order for a practitioner to understand whether the discriminatively trained network has learned to distinguish multiple concepts reasonably well, she can directly obtain samples from the model for each different concept. As an illustrative example, we choose the task of transcribing polyphonic audio into a symbolic format. This is a multi-label problem, assigning multiple note labels to each (quantized) point in time. Transcription systems based on neural networks are commonly learned from corpora containing large amounts of polyphonic music. Due to having the inverse available to us in closed form, we are able to sample all different single notes from the network to directly see whether the concept of single, isolated notes could be learned by training on our polyphonic corpus, or if multiple notes have been "smeared" together, and could not be disentangled from each other, or if the concept could not be learned at all. To the best of our knowledge, this is still an open problem that mostly affects polyphonic transcription systems based on neural networks, as discussed in [20].

2. RELATED WORK

Invertible neural networks were first introduced in [6] and rediscovered in [2]. They define a nonlinear, bijective mapping between inputs and outputs. They can be used to transform arbitrarily complex distributions into simple, factorized distributions. This concept became more widely known as normalizing flows, introduced in [11], generalized in [33], and has been used in [8] for density estimation, and improving variational inference in [26]. Various types of (more expressive) normalizing flows have been introduced in [9,34,35]. In [21] normalizing flows are employed as generative models for high resolution samples comparable to those produced by high resolution generative adversarial networks (GANs), e.g., [18].

With a greater focus on the invertibility aspect, [1] uses bijective architectures to approximate physical processes with a well defined forward model, in order to obtain the posterior distribution over inputs conditioned on desired outputs. We adopt parts of their terminology and training procedure. The differences will be discussed in more detail in Section 3. In [17] injective and bijective i-RevNets are introduced, architectures similar to ResNets [14], which are invertible up to the last layer. In [16], fully invertible RevNets in conjunction with a new objective function are used to train classifiers which are more robust against adversarial attacks. We borrow their term "nuisance" variables to describe what information is supposed to end up in the output vector $\mathbf{z}$.

Distribution matching in this work is done using the sliced Wasserstein distance. Introduced in [4, 25] as a distance measure for texture synthesis in a computer graphics setting, it has been used for encouraging the codes of autoencoders to follow a proposal distribution [22], and has also been directly applied to generative modeling of images, replacing the domain regressor in GANs [7].

Finally, we draw inspiration from [13] where a transcription-resynthesis system was introduced, consisting of three separately trained parts, a transcription system, a language model and a (neural) synthesizer.

3. METHOD

We adhere closely to the invertible neural network architectures described in [1], with a minor modification to the training procedure that will be outlined after the formal introduction of invertible neural networks. Our notation also loosely follows the one used in [1]. Given a data space $\mathcal{X}$, a label space $\mathcal{Y}$ and a nuisance space $\mathcal{Z}$, we consider a directly invertible neural network as a parametrized function $f_\theta : \mathcal{X} \to \mathcal{Y} \times \mathcal{Z}$ where we have access to its closed form inverse $f_\theta^{-1} : \mathcal{Y} \times \mathcal{Z} \to \mathcal{X}$.

The function f_θ maps the input into a label space that carries semantic information that we are interested in, and maps the rest of information into a nuisance space. Given both the semantic and nuisance information, we are able to obtain the input again via f_θ^{-1}.

There are a few different ways such a function can be implemented in practice, and they all come with various different architectural constraints. We will first define a small invertible building block with relatively weak capabilities. These building blocks are then used to construct a more expressive function.

A necessary structural restriction to a bijective block is that the dimensionality of the input must match the dimensionality of the output. Another restriction concerns the inner workings of blocks, so their inverse is available in closed form. We adopt the affine coupling layer design in [1], which is a more expressive version of the one in [9]. Its internal structure can be seen in Figure 2. The layer takes as input a vector $\mathbf{u}$, whose dimensions are first shuffled with a fixed random permutation matrix and then split into two halves $\mathbf{u}_1, \mathbf{u}_2$. Dimension shuffling causes the splits to be different from layer to layer and facilitates interaction between components whose indices might be far apart in the input vector. The permutation matrix is inverted by simply transposing it.

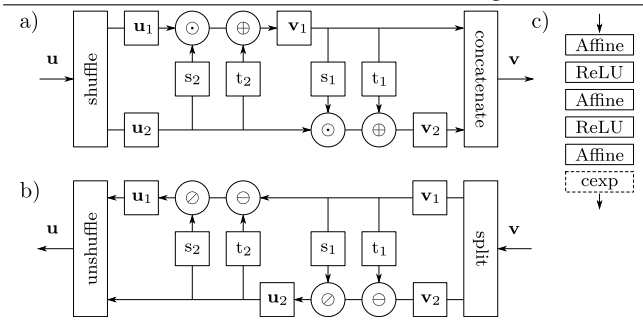


Figure 2: This sketch depicts the structure of the particular version of affine coupling layers we use. **a**) The operations as they are applied in the forward direction. **b**) The operations as they are applied in the backward direction. **c**) The parametrization of the $\mathrm{s}_{1|2}$ and $\mathrm{t}_{1|2}$ transforms. The cexp function is only applied after the $\mathrm{s}_{1|2}$ transforms.

Different operations are then applied to each half, after which the halves are concatenated again to yield the output $\mathbf{v}$. Equations (1) – (4) show the exact expressions used to compute results in both directions.

$$\mathbf{v}_1 = \mathrm{cexp}(\mathrm{s}_2(\mathbf{u}_2)) \odot \mathbf{u}_1 \oplus \mathrm{t}_2(\mathbf{u}_2) \tag{1}$$
$$\mathbf{v}_2 = \mathrm{cexp}(\mathrm{s}_1(\mathbf{v}_1)) \odot \mathbf{u}_2 \oplus \mathrm{t}_1(\mathbf{v}_1) \tag{2}$$
$$\mathbf{u}_2 = (\mathbf{v}_2 \ominus \mathrm{t}_1(\mathbf{v}_1)) \oslash \mathrm{cexp}(\mathrm{s}_1(\mathbf{v}_1)) \tag{3}$$
$$\mathbf{u}_1 = (\mathbf{v}_1 \ominus \mathrm{t}_2(\mathbf{u}_2)) \oslash \mathrm{cexp}(\mathrm{s}_2(\mathbf{u}_2)) \tag{4}$$

Operations $\oplus, \ominus, \odot, \oslash$ (addition, subtraction, multiplication, division) are applied elementwise. The function cexp is defined as $\mathrm{cexp}(x) = \exp(c \cdot \mathrm{atan}(x))$ with $c > 0$ being a hyperparameter. Its purpose is to constrain the output to a reasonable range, and to prevent runaway growth of activations. The transforms $\mathrm{s}_{1|2}$ and $\mathrm{t}_{1|2}$ are arbitrarily parametrizable functions, modeling input dependent scaling and translation respectively. All transforms are implemented as standard neural networks, and are not required to be invertible, because the transformed half of the output vector can be inverted using the untransformed half. The network structures we use are shown in Figure 2c.

If the dimensionalities of input and output vectors do not match, the vectors are padded with zeros during inference, or small scale Gaussian noise during training, to encourage the network to ignore the additional padding dimensions, as done in [1].

Each update of the model involves three passes, one forward pass, and two backward passes. Each pass has its own set of objective functions. The joint objective function to be minimized consists of a weighted sum of these terms. We specify the following notation: vectors are in boldface, writing vectors in square brackets separated by semicolons $[\mathbf{a}; \mathbf{b}]$ denotes concatenation. The vector $\mathbf{x}$ is the input to the model, $\mathbf{y}$ is the semantic part of the groundtruth, and $\mathbf{z}$ is a sample from a proposal distribution, which we choose to be $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The padding vectors used during training are denoted as $\mathbf{x}_{pad}$ and $\mathbf{yz}_{pad}$ respectively, and are drawn from $\mathcal{N}(\mathbf{0}, \varepsilon)$ for each update, with $\varepsilon > 0$ a hyperparameter. Symbols with a circumflex always refer to model outputs with a direct counterpart in the groundtruth. We denote a zero vector of a size appropriate in the context it appears in as $\mathbf{0}$. A sample from the model will be written as $\mathbf{x}_{sam}$. Equation (5) fully specifies all inputs and outputs for an invertible neural network used in the forward direction, equation (6) does the same in the backward direction, and (7) specifies how samples are drawn.

Algorithm 1 Sliced Wasserstein Distance $d_{SWD}(\mathbf{A}, \mathbf{B})$

Let $S \leftarrow 0$ and $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times d}$ (two samples)
For $1 \,..\, m$ **do**
  $\mathbf{p} \leftarrow \mathbf{p}'/\|\mathbf{p}'\|$ such that $\mathbf{p}' \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathbf{p}' \in \mathbb{R}^{d \times 1}$
  $\mathbf{a} \leftarrow \mathrm{sort}[\mathbf{Ap}]$; $\mathbf{b} \leftarrow \mathrm{sort}[\mathbf{Bp}]$
  $S \leftarrow S + \|\mathbf{a} - \mathbf{b}\|_2^2 / n$
Return S/m

$$[\hat{\mathbf{z}}; \hat{\mathbf{yz}}_{pad}; \hat{\mathbf{y}}] = f([\mathbf{x}; \mathbf{x}_{pad}]) \tag{5}$$
$$[\hat{\mathbf{x}}; \hat{\mathbf{x}}_{pad}] = f^{-1}[\hat{\mathbf{z}}; \mathbf{yz}_{pad}; \hat{\mathbf{y}}] \tag{6}$$
$$[\mathbf{x}_{sam}; \hat{\mathbf{x}}_{pad}] = f^{-1}[\mathbf{z}; \mathbf{0}; \mathbf{y}] \tag{7}$$

Having defined these quantities, we can now proceed with defining the individual loss terms that will make up the joint objective function. Mean squared error (8) is used to fit the labels from the groundtruth, and the reconstruction of the input (9). We deviate from [1] and use the sliced Wasserstein distance (d_{SWD}) [25] instead of the maximum mean discrepancy (d_{MMD}), to measure the distance between distributions, as we found it to be better behaved for high dimensional data. The intuition behind d_{SWD} is to decompose the high dimensional optimal transport problem into m 1-dimensional ones, by randomly projecting samples $\mathbf{A}$ and $\mathbf{B}$ onto lines, allowing the resulting 1-dimensional problems to be solved by computing the distance between sorted components. In equation (10), d_{SWD} is used to minimize the distance between samples from the joint distribution over the outputs $[\hat{\mathbf{y}}; \hat{\mathbf{z}}]$ and samples from the joint distribution over the labels and the proposal distribution $[\mathbf{y}; \mathbf{z}]$. Please note that following the advice laid out in [1], no gradient information from this objective is propagated back over $\hat{\mathbf{y}}$, to not unduly disturb the label fitting process. Informally stated, the purpose of including $\mathbf{y}$ and $\hat{\mathbf{y}}$ in the distribution matching process is to "group" samples together for which $\hat{\mathbf{z}}$ needs to follow a Gaussian distribution, resulting in the distributions $p(\hat{\mathbf{y}})$ and $p(\hat{\mathbf{z}})$ to gradually decouple, and become independent of each other, with the side effect of erasing label information from $\hat{\mathbf{z}}$. d_{SWD} is also used in (11), to minimize the distance between the distribution of samples generated from the model and the groundtruth.

$$\mathcal{L}_{\mathbf{y}}(\mathbf{y}, \hat{\mathbf{y}}) = \mathrm{MSE}(\mathbf{y}, \hat{\mathbf{y}}) \tag{8}$$
$$\mathcal{L}_{\hat{\mathbf{x}}}(\mathbf{x}, \hat{\mathbf{x}}) = \mathrm{MSE}(\mathbf{x}, \hat{\mathbf{x}}) \tag{9}$$
$$\mathcal{L}_{\mathbf{yz}}([\mathbf{y}; \mathbf{z}], [\hat{\mathbf{y}}; \hat{\mathbf{z}}]) = \mathrm{SWD}([\mathbf{y}; \mathbf{z}], [\hat{\mathbf{y}}; \hat{\mathbf{z}}]) \tag{10}$$
$$\mathcal{L}_{\mathbf{x}_{sam}}(\mathbf{x}, \mathbf{x}_{sam}) = \mathrm{SWD}(\mathbf{x}, \mathbf{x}_{sam}) \tag{11}$$
$$\mathcal{L}_{\mathbf{x}_{pad}}(\mathbf{x}_{pad}, \hat{\mathbf{x}}_{pad}) = \mathrm{MSE}(\mathbf{x}_{pad}, \hat{\mathbf{x}}_{pad}) \tag{12}$$
$$\mathcal{L}_{\mathbf{yz}_{pad}}(\mathbf{yz}_{pad}, \hat{\mathbf{yz}}_{pad}) = \mathrm{MSE}(\mathbf{yz}_{pad}, \hat{\mathbf{yz}}_{pad}) \tag{13}$$

Finally, the padding dimensions are taken care of with mean squared error terms (12) and (13), to encourage the network to disregard information in these dimensions. Following advice in [1], the individual loss terms that sum up to the joint objective are weighted such that their magnitudes are approximately equal to each other, by restarting the optimization process multiple times and adjusting the weights until this condition is met.

4. EXPERIMENTS

This section is split into multiple parts, starting out with a description of the data preparation procedure, followed by an empirical assessment of the usability of INNs for practitioners in subsection 4.1, a critical examination of the interpretability of a trained model in subsection 4.2, and finally an analysis of how well the concept of single notes could be learned from a polyphonic corpus in subsection 4.3.

All model training, testing and generative sampling experiments were carried out with the MUS subset of the MAPS corpus [10]. This subset contains 210 polyphonic piano pieces rendered with 7 sample based synthesizers, and 60 recordings of a YAMAHA Disklavier in two different recording conditions. After removing all synthesized pieces that also occur in the set of recordings, we are left with 139 pieces for training, and the 60 Disklavier recordings for testing, according to the procedure outlined in [12]. Evaluation measures are computed individually for each piece in the test set, and the mean over all pieces is reported. Groundtruth information is available as temporally aligned MIDI files. Sustain pedal control values are quantized, and the pedal considered fully engaged if its MIDI control value exceeds 64. All offsets of notes that are sounding while sustain is in effect are extended in time, until the pedal is released again.

The label information $\mathbf{y}$ that the model has to learn is derived from the MIDI groundtruth and consists of 3 parts: the note phase, its velocity and instrument information. For each piano key, the temporal evolution each note is modeled with an exponentially decaying curve, defined as $\text{curve}(\tau) = 0.99^{\tau} \cdot 5$, with $0 \leq \tau < \text{duration}$. It starts at the onset of a note, lasts for its duration, and drops off immediately after the offset. The velocity part is derived from the MIDI velocity value scaled into the interval $[0, 1]$. This procedure is also outlined in Figure 3, and repeated for each of the 88 piano keys. Finally, instruments are numbered from 0 to 8, corresponding to one of the 7 sample banks or alternatively one of the two microphone conditions for the Disklavier recordings, and are one-hot encoded. For each (quantized) point in time t all three parts are concatenated into the vector $\mathbf{y}_t$, having $9 + 88 + 88 = 185$ components. This particular label vector derivation is chosen so that $\mathbf{y}_t$ contains all necessary information to generate spectrogram frames for different instruments and notes at the right volume and the right stage of a notes' temporal evolution without any additional context information from neighboring frames. The length of $\mathbf{z}_t$ was treated as a hyperparameter, and selected via cross validation on a small subset of the training set. Its length appears to have negligible influence given all the other settings, and was set to 9 for all models subsequently used. The corresponding data $\mathbf{x}_t$ are magnitude spectrograms processed by a semi logarithmic filterbank, and the resulting bins $\mathbf{b}_t$ are elementwise processed by the function $\log(1 + \mathbf{b}_t)$, approximating human loudness perception to finally yield a vector $\mathbf{x}_t$ of length 144. All spectral feature extraction and filtering is done with the `madmom` library [3]. The frame rate at which pairs $(\mathbf{x}_t, \mathbf{y}_t)$ are extracted from the audio and MIDI files is 25 frames per second. As all input is processed in a framewise fashion everywhere, we omit the subscript t, denoting time in frames, for all plots and most equations to not add additional clutter. To increase the capacity of the INN, we add zero padding vectors to both input ($\mathbf{x}_{pad}$) and output dimensions ($\mathbf{yz}_{pad}$), so the number of components in the padded vectors sum up to 256 in total. Training follows the procedure outlined in Section 3, and all code is released [1] to facilitate reproducability.

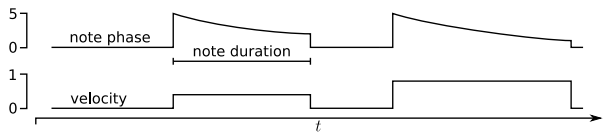


Figure 3: This illustration shows how the note phase and velocity part of the label information $\mathbf{y}$ is derived for multiple notes played by a single key.

4.1 Usability for MIR tasks

As a kind of quantitative viability test, the capability of INNs (in combination with simple temporal models) to produce predictions useful to MIR practitioners, is examined. We train small recurrent networks (RNNs) on the framewise predictions $\hat{\mathbf{y}}$ obtained from the INN, in the hope to obtain cleaner, denoised framewise predictions $\hat{\mathbf{y}}'$. The types of RNN cells we use are either LSTM [15] or GRU [5] cells. In a first attempt, RNNs with very limited capacity - 4 hidden units / cells for all keys - are employed. An input sequence to the RNN consists of the note phase and the velocity part of a *single* key over the whole length of the piece, leading to an input dimension of 2. The RNNs should output a smoothed, denoised version of the note phase and velocity sequence, and an additional framewise note activity indicator between 0 and 1, and thus all have 3 outputs. The binary piano roll used to compute framewise performance measures is obtained by thresholding the note activity indicator output of the RNNs at 0.5. Training proceeds one full sequence at a time, picked uniformly at random from all $139 \cdot 88$ single key sequences derived from all pieces in the training set. The models with highest F_1-measure on a subset of the training set are then evaluated on the test set.

In order to evaluate framewise performance for a piece, we produce framewise transcriptions for all keys with the INN. Each key is then separately smoothed, denoised, and

[1] https://github.com/rainerkelz/ISMIR19

Method	P	R	F_1
INN + GRU (S)	79.74	63.73	70.84
INN + LSTM (S)	80.12	63.91	71.10
INN + biGRU (L)	81.72	64.81	72.29
CNN only [12, 19]	81.18	65.07	71.60
CNN + RNN-NADE [12, 29]	71.99	73.32	72.22
CNN + LSTM [12]	88.53	70.89	78.30

Table 1: Framewise performance of different combinations of acoustic and temporal models on the testset.

its activity is inferred over the length of the piece. We report the results for the small models in Table 1, suffixed with "(S)". We would like to note that there was next to no hyperparameter tuning done, aside from getting the learn rates for the two different RNN cell types approximately in the right regime. A slightly larger version uses 3 layers of bi-directional GRU cells with 8 hidden units, and dropout [30] with a probability of 0.5, applied to the output of each recurrent layer, before it is passed on to the next. Results in the table for this type of RNN are suffixed with "(L)". The INN has 5 invertible layers, and 990.720 parameters in total. The parameter counts for the recurrent model variants "GRU (S)", "LSTM (S)" and "biGRU (L)" are 111, 143 and 3123 respectively.

We can see that the combination framewise INN + biGRU performs on par with the CNN + RNN-NADE combination in terms of framewise performance, and slightly outperforms the standalone CNN. The last three rows in Table 1 are taken from [12], who re-implemented the approaches in [19] and [29], and ostensibly performed additional hyperparameter tuning to improve upon the original results. They also provide the current state of the art results for this train and test protocol in the last row, achieved by supplying an additional onset target to the network during training.

4.2 Interpretability of results

This section considers how the ability to modify the output of the model, and then using it in the backward direction, can assist the practitioner in determining the causes in the data that led to a particular prediction. We start with a thought experiment, and get closer to reality step by step. Let us assume the model works perfectly, and given an input $\mathbf{x}$, the model routes all semantic information (note phases, velocities, instrument) into the $\hat{\mathbf{y}}$ vector, all nuisance information (other acoustic variability, such as microphone characteristics, room reverberation or actual noise) ends up in $\hat{\mathbf{z}}$ which is distributed as $\mathcal{N}(\mathbf{0}, \mathbf{I})$, and the padding vector $\hat{\mathbf{yz}}$ is exactly zero. Sampling $\mathbf{z}_s \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and using $f^{-1}([\hat{\mathbf{y}}, \mathbf{0}, \mathbf{z}_s])$ to obtain the corresponding input $\mathbf{x}_s$ will change *only nuisance characteristics* in the input. This would also mean that we have *full control* over the semantic content of the input. We could add or delete notes *in the input* simply by adding or zeroing them out in $\hat{\mathbf{y}}$, much like we can insert or delete a symbolic MIDI note, the implication being that every output has a *directly* interpretable correspondence in the input.

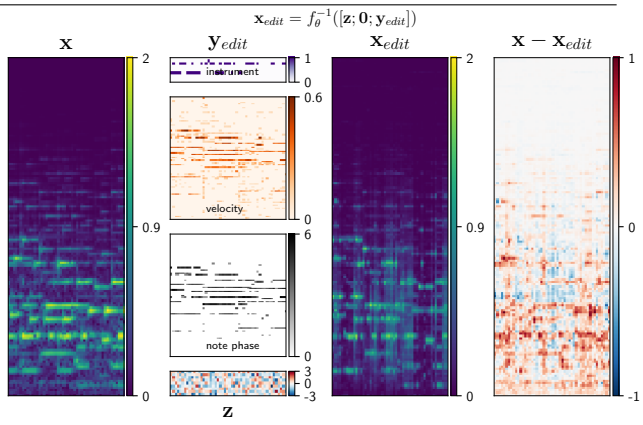


Figure 4: Gradually denoising the predictions with simple, ad-hoc rules, zero padding and sampling $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. In practice, this would be done iteratively, always keeping an eye on how the overall structure of the input is affected.

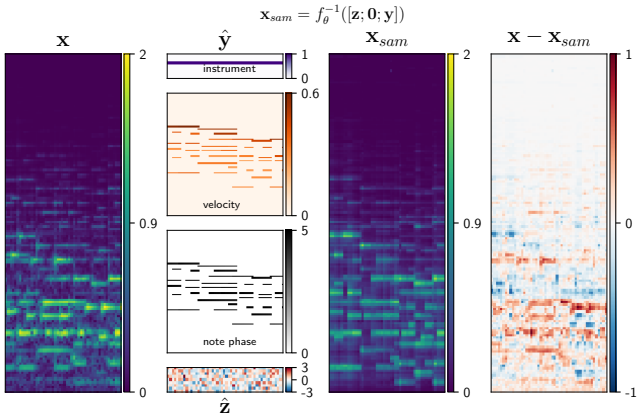


Figure 5: The hypothetical case where all predictions could be perfectly denoised. For purely demonstrational purposes, this was accomplished through consultation of an oracle, but one could imagine this to be achievable through interaction with the system. Although quite a bit of detail is missing, the majority of the structure in the original input is nonetheless recognizable in the generated sample.

A closer look at Figure 1 reveals what can realistically be achieved just by training a framewise INN with a rather limited amount of polyphonic data. It is immediately noticeable that $\hat{\mathbf{z}}$ does *not* appear to be normally distributed. There are still patterns discernible, making it apparent that it still contains semantic information. Similar patterns also exist in $\hat{\mathbf{yz}}_{pad}$ (not shown).

It is also observable that the information that is routed into $\hat{\mathbf{y}}$ is somewhat noisy. We can now attempt to "separate the wheat from the chaff" by using the INN in the backward direction with cleaned up predictions. Figure 4 shows what happens when the predictions are partially denoised by setting all predictions below a certain threshold to zero, ignoring the padding vector by zeroing it out, and sampling $\mathbf{z}$ from a unit normal distribution. These simple, ad-hoc rules cannot get rid of all the noise and discontinuities in the predictions, but are useful to determine which outputs can be ignored by observing their (collective) im-

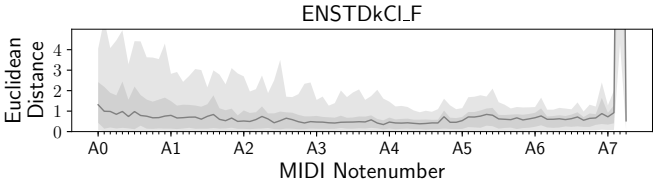


Figure 6: Interquantile ranges of the framewise Euclidean distances between isolated reference notes and samples from the model. The reference notes stem from an unseen instrument.

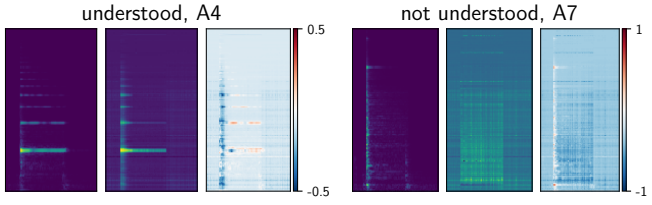


Figure 7: Samples from the model for single notes.

pact on the sample. Figure 5 depicts what can be generated by the model, assuming that the denoising process of the predictions were perfect, by consulting an oracle about the true contents of $\mathbf{y}$. In all figures discussed in this section, the same excerpt from the test set was used, meaning the model has never seen any of the examples during training.

4.3 Concept Understanding

Returning to a question raised in the introduction, in this section the model will be systematically queried about specific semantic concepts. Arguably, a polyphonic transcription system should be able to transcribe isolated notes. The MAPS dataset provides both renderings and recordings of isolated notes, which we utilize to formulate our queries. For each of the 88 keys, 30 samples are drawn from the model, using the groundtruth $\mathbf{y}$ paired with the reference recording for the key and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This ensures that each sample has the same length as the reference. For each frame at time t in a sample, the Euclidean distance to the corresponding frame of the reference recording is measured, and p-quantiles are computed on the resulting lists of framewise distances, with $p \in \{0.05, 0.25, 0.5, 0.75, 0.95\}$. Figure 6 depicts the interquantile range $[0.05, 0.95]$ as light gray, the range $[0.25, 0.75]$ in a darker shade, and the median as a black line. It becomes immediately apparent that samples for rarely occuring (possibly omitted) notes, such as those in the lower and higher octaves, are highly dissimilar from the reference recordings, and indicate that these particular isolated notes could not be learned by the network (Figure 7). Admittedly, this question could have been answered for a regular feedforward network as well, but would have necessitated *more labeled* reference data of the *same* instrument. The ability to sample from the model allows us to sidestep the rather cumbersome way of aggregating prediction errors, as was necessary in [20], to arrive at a similar conclusion.

4.4 Improving Models with temporal context

The invertible models investigated so far all take single frames as input, without temporal context information. We trained fully invertible RevNets [16, 17] on a variety of different context lengths, but were not yet able to observe either quantitative or qualitative improvements over the framewise models. RevNets tend to become rather large in terms of the number of parameters, input and output padding is not as straightforward as for framewise models, the input and output space dimensionality is much larger, making the sliced Wasserstein distance gradually less effective due to an increase in necessary computational resources, which in turn further slows down training. Finally, the amount of training data we use may simply be insufficient for higher capacity models. However, we believe that all these issues have appropriate remedies. An immediate next step would be to apply the same models to the much larger MAESTRO dataset [13]. We leave these steps for future work though.

5. CONCLUSION

The viability of invertible neural networks for a selected MIR task was shown quantitatively in terms of transcription performance and a brief numerical analysis of single concept understanding. A qualitative investigation of the direct interpretability of outputs back in input space was conducted. There is ample room for improvement, such as using an adversarial distance for distribution matching in both input and output space, or alternatively using the independent cross-entropy objective from [16] in latent space. The objective for the semantic part of the output space could be similarly augmented to encourage the disentanglement of (predictions for) individual notes. Another obvious improvement would be to skip the computation of filtered spectrograms altogether, and feed in waveforms to obtain models that can in turn generate waveforms we can directly listen to.

Beyond the interpretability aspect, we are confident that invertible neural networks will prove to be useful for other MIR tasks as well, such as musical content-aware style transfer (this is already doable with the models used in this work, by simply changing the instrument encoding when sampling, although changing from one piano to a different one is not as exciting as changing it into a trumpet). These models could also be adapted for (blind) source separation, to name only two examples.

6. ACKNOWLEDGEMENTS

We would like to express our deep gratitude to Lynton Ardizzone for pointing us in the right direction and Hamid Eghbal-zadeh for noting the pitch content-aware style transfer angle. This research has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 670035, project CON ESPRESSIONE). The Tesla K40 used for this research was donated by the NVIDIA Corporation.

7. REFERENCES

[1] Lynton Ardizzone, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Analyzing Inverse Problems with Invertible Neural Networks. In *International Conference on Learning Representations*, 2019.

[2] L. Baird, D. Smalenberger, and S. Ingkiriwang. One-step neural network inversion with PDF learning and emulation. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 966–971 vol. 2, July 2005.

[3] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, 10 2016.

[4] Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and Radon Wasserstein Barycenters of Measures. *Journal of Mathematical Imaging and Vision*, 51(1):22–45, 2015.

[5] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734, 2014.

[6] Gustavo Deco and Wilfried Brauer. Nonlinear higher-order statistical decorrelation by volume-conserving neural architectures. *Neural Networks*, 8(4):525–535, 1995.

[7] Ishan Deshpande, Ziyu Zhang, and Alexander G. Schwing. Generative Modeling Using the Sliced Wasserstein Distance. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3483–3491, 2018.

[8] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[9] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[10] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(6):1643–1654, 2010.

[11] Esteban G. Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences - COMMUN MATH SCI*, 8, 03 2010.

[12] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and Frames: Dual-Objective Piano Transcription. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27*, 2018.

[13] Curtis Hawthorne, Andrew Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset. In *International Conference on Learning Representations*, 2019.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[16] Joern-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive Invariance Causes Adversarial Vulnerability. In *International Conference on Learning Representations*, 2019.

[17] Jörn-Henrik Jacobsen, Arnold W. M. Smeulders, and Edouard Oyallon. i-RevNet: Deep Invertible Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[19] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the Potential of Simple Framewise Approaches to Piano Transcription. In *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*, pages 475–481, 2016.

[20] Rainer Kelz and Gerhard Widmer. An Experimental Analysis of the Entanglement Problem in Neural-Network-based Music Transcription Systems. In *AES*

International Conference Semantic Audio 2017, Erlangen, Germany, June 22-24, 2017, 2017.

[21] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 10236–10245, 2018.

[22] Soheil Kolouri, Phillip E. Pope, Charles E. Martin, and Gustavo K. Rohde. Sliced Wasserstein Auto-Encoders. In *International Conference on Learning Representations*, 2019.

[23] Saumitra Mishra, Bob L. Sturm, and Simon Dixon. Local Interpretable Model-Agnostic Explanations for Music Content Analysis. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pages 537–543, 2017.

[24] Gregory Plumb, Denali Molitor, and Ameet S. Talwalkar. Model Agnostic Supervised Local Explanations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 2520–2529, 2018.

[25] Julien Rabin, Gabriel Peyré, Julie Delon, and Marc" Bernot. Wasserstein Barycenter and Its Application to Texture Mixing. In Alfred M. Bruckstein, Bart M. ter Haar Romeny, Alexander M. Bronstein, and Michael M. Bronstein, editors, *Scale Space and Variational Methods in Computer Vision*, pages 435–446, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[26] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1530–1538, 2015.

[27] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[28] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 618–626, 2017.

[29] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An End-to-End Neural Network for Polyphonic Piano Music Transcription. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 24(5):927–939, 2016.

[30] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[31] B. L. Sturm. A Simple Method to Determine if a Music Information Retrieval System is a "Horse". *IEEE Transactions on Multimedia*, 16(6):1636–1644, Oct 2014.

[32] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3319–3328, 2017.

[33] E. G. Tabak and Cristina V. Turner. A Family of Nonparametric Density Estimation Algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

[34] Jakub M. Tomczak and Max Welling. Improving Variational Auto-Encoders using Householder Flow. *CoRR*, abs/1611.09630, 2016.

[35] Rianne van den Berg, Leonard Hasenclever, Jakub M. Tomczak, and Max Welling. Sylvester Normalizing Flows for Variational Inference. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 393–402, 2018.

LEARNING TO GENERATE MUSIC WITH SENTIMENT

Lucas N. Ferreira
University of California, Santa Cruz
Department of Computational Media

Jim Whitehead
University of California, Santa Cruz
Department of Computational Media

ABSTRACT

Deep Learning models have shown very promising results in automatically composing polyphonic music pieces. However, it is very hard to control such models in order to guide the compositions towards a desired goal. We are interested in controlling a model to automatically generate music with a given sentiment. This paper presents a generative Deep Learning model that can be directed to compose music with a given sentiment. Besides music generation, the same model can be used for sentiment analysis of symbolic music. We evaluate the accuracy of the model in classifying sentiment of symbolic music using a new dataset of video game soundtracks. Results show that our model is able to obtain good prediction accuracy. A user study shows that human subjects agreed that the generated music has the intended sentiment, however negative pieces can be ambiguous.

1. INTRODUCTION

Music Generation is an important application domain of Deep Learning in which models learn musical features from a dataset in order to generate new, interesting music. Such models have been capable of generating high quality pieces of different styles with strong short-term dependencies [1] [2]. A major challenge of this domain consists of disentangling these models to generate compositions with given characteristics. For example, one can't easily control a model trained on classical piano pieces to compose a tense piece for a horror scene of a movie. Being able to control the output of the models is specially important for the field of Affective Music Composition, whose major goal is to automatically generate music that is perceived to have a specific emotion or to evoke emotions in listeners [19]. Applications involve generating soundtracks for movies and video-games [18], sonification of biophysical data [3] and generating responsive music for the purposes of music therapy and palliative care [9].

Recently, Radford et al. [13] showed that a generative Long short-term memory (LSTM) neural network can learn an excellent representation of sentiment (positive-negative) on text, despite being trained only to predict the next character in the Amazon reviews dataset [6]. When combined to a Logistic Regression, this LSTM achieves state-of-the-art sentiment analysis accuracy on the Stanford Sentiment Treebank dataset and can match the performance of previous supervised systems using 30-100x fewer labeled examples. This LSTM stores almost all of the sentiment signal in a distinct "sentiment neuron", which can be used to control the LSTM to generate sentences with a given sentiment. In this paper, we explore this approach with the goal of composing symbolic music with a given sentiment. We also explore this approach as a sentiment classifier for symbolic music.

In order to evaluate this approach, we need a dataset of music in symbolic format that is annotated by sentiment. Even though emotion detection is an important topic in music information retrieval [7], it is typically studied on music in audio format. To the best of our knowledge, there are no datasets of symbolic music annotated according to sentiment. Therefore, we created a new dataset composed of 95 MIDI labelled piano pieces (966 phrases of 4 bars) from video game soundtracks. Each piece is annotated by 30 human subjects according to a valence-arousal (dimensional) model of emotion [15]. The sentiment of each piece is then extracted by summarizing the 30 annotations and mapping the valence axis to sentiment. The same dataset also contains another 728 non-labelled pieces, which were used for training the generative LSTM.

We combine this generative LSTM with a Logistic Regression and analyse its sentiment prediction accuracy against a traditional classification LSTM trained in a fully-supervised way. Results showed that our model (generative LSTM with Logistic Regression) outperformed the supervised LSTM by approximately 30%. We also analysed the generative capabilities of our model with a user study. Human subjects used an online annotation tool to label 3 pieces controlled to be negative and 3 pieces controlled to be positive. Results showed human annotators agree the generated positive pieces have the intended sentiment. The generated negative pieces appear to be ambiguous, having both negative and positive parts.

We believe this paper is the first work to explore sentiment analysis in symbolic music and it presents the first disentangled Deep Learning model for music generation with sentiment. Another contribution of this paper is a labelled dataset of symbolic music annotated according to sentiment. These contributions open several direction for

[1] Supporting strong long-term dependencies (music form) is still an open problem.

future research, specially music generation with emotions as both a multi-class problem and as a regression problem. Moreover, these methods could be applied to create soundtrack generation systems for films, video games, interactive narratives, audio books, etc.

2. RELATED WORK

This paper is related to previous work on Affective Algorithmic Music Composition, more specifically to works that process music in symbolic form in order to generate music with a given emotion. A common approach for this problem consists of designing a rule-based system to map musical features to a given emotion in a categorical or dimensional space [19]. For example, Williams et al. [18] propose a system to generate soundtracks for video games where each game's scene graph (defining all the possible branching of scenes in the game) is annotated according to a valence-arousal model. A second-order Markov model is used to learn melodies from a dataset and are then transformed by a rule-based system to fit the annotated emotions in the graph. Davis and Mohammad [4] follow a similar approach in TransPose, a system that composes piano melodies for novels. TransPose uses a lexicon-based approach to automatically detect emotions (categorical model) in novels and a rule-based technique to create piano melodies with these emotions.

There are a few other approaches in the literature to compose music with a given emotion. Scirea et al. [16] recently presented a framework called MetaCompose designed to create background music for games in real-time. MetaCompose generates music by (i) randomly creating a chord sequence from a pre-defined chord progression graph, (ii) evolving a melody for this chord sequence using a genetic algorithm and (iii) producing an accompaniment for the melody/chord sequence combination. Monteith et al. [10] approaches Affective Algorithmic Music Composition from a Machine Learning perspective to learn melodies and rhythms from a corpus of music labeled according to a categorical model of emotion. Individual Hidden Markov models and n-grams are trained for each category to generate pitches and underlying harmonies, respectively. Rhythms are sampled randomly from examples of a given category.

Deep Learning models have recently achieved high-quality results in music composition with short-term dependencies [2]. These models normally are trained on a corpus of MIDI files to predict the next note to be played based on a given note. In general, these models can't be manipulated to generate music with a given emotion. For example, in the system DeepBach, Hadjeres et al. [5] use a dependency network and a Gibbs-like sampling procedure to generate high-quality four-part chorales in the style of Bach. Roberts et at. [14] train recurrent variational autoencoder (VAEs) to reproduce short musical sequences and with a novel hierarchical decoder they are able to model long sequences with musical structure for both individual instruments and a three-piece band (lead, bass, and drums).

The majority of the deep learning models are trained to generate musical scores and not performances. Oore et al. [11] tackles this problem by training an LSTM with a new representation that supports tempo and velocity events from MIDI files. This model was trained on the Yamaha e-Piano Competition [1], which contains MIDI captures of ~1400 performances by skilled pianists. With this new representation and dataset, Oore et al. [11] generated more human-like performances when compared to previous models.

3. MODEL

We propose a Deep Learning method for affective algorithmic composition that can be controlled to generate music with a given sentiment. This method is based on the work of Radford et al. [13] which generates product reviews (in textual form) with sentiment. Radford et al. [13] used a single-layer multiplicative long short-term memory (mLSTM) network [8] with 4096 units to process text as a sequence of UTF-8 encoded bytes (character-based language modeling). For each byte, the model updates its hidden state of the mLSTM and predicts a probability distribution over the next possible byte.

This mLSTM was trained on the Amazon product review dataset, which contains over 82 million product reviews from May 1996 to July 2014 amounting to over 38 billion training bytes [6]. Radford et al. [13] used the trained mLSTM to encode sentences from four different Sentiment Analysis datasets. The encoding is performed by initializing the the states to zeros and processing the sequence character-by-character. The final hidden states of the mLSTM are used as a feature representation. With the encoded datasets, Radford et al. [13] trained a simple logistic regression classifier with L1 regularization and outperformed the state-of-the-art methods at the time using 30-100x fewer labeled examples.

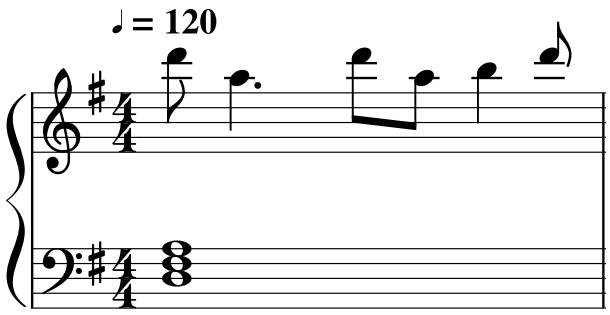
By inspecting the relative contributions of features on various datasets, Radford et al. [13] discovered a single unit within the mLSTM that directly corresponded to sentiment. Because the mLSTM was trained as a generative model, one can simply set the value of the sentiment unit to be positive or negative and the model generates corresponding positive or negative reviews.

3.1 Data Representation

We use the same combination of mLSTM and logistic regression to compose music with sentiment. To do this, we treat the music composition problem as a language modeling problem. Instead of characters, we represent a music piece as a sequence of words and punctuation marks from a vocabulary that represents events retrieved from the MIDI file. Sentiment is perceived in music due to several features such as melody, harmony, tempo, timbre, etc [7]. Our data representation attempts to encode a large part of these features [2] using a small set of words:

- "n_[pitch]": play note with given pitch number: any integer from 0 to 127.

[2] Constrained by the features one can extract from MIDI data.




```
t_120 v_76 d_whole_0 n_50 n_54 n_57
v_92 d_eighth n_86 . . v_84
d_quarter_1 n_81 . .
```

Figure 1: A short example piece encoded using our proposed representation. The encoding represents the first two time steps of the shown measure.

- "d_[duration]_[dots]": change the duration of the following notes to a given duration type with a given amount of dots. Types are breve, whole, half, quarter, eighth, 16th and 32nd. Dots can be any integer from 0 to 3.
- "v_[velocity]": change the velocity of the following notes to a given velocity (loudness) number. Velocity is discretized in bins of size 4, so it can be any integer in the set $V = 4, 8, 12, \ldots, 128$.
- "t_[tempo]": change the tempo of the piece to a given tempo in bpm. Tempo is also discretized in bins of size 4, so it can be any integer in the set $T = 24, 28, 32, \ldots, 160$.
- ".": end of time step. Each time step is one sixteenth note long.
- "\n": end of piece.

For example, Figure 1 shows the encoding of the first two time steps of the first measure of the Legend of Zelda - Ocarina of Time's Prelude of Light. The first time step sets the tempo to 120bpm, the velocity of the following notes to 76 and plays the D Major Triad for the duration of a whole note. The second time step sets the velocity to 84 and plays a dotted quarter A5 note. The total size of this vocabulary is 225 and it represents both the composition and performance elements of a piece (timing and dynamics).

4. SENTIMENT DATASET

In order to apply the Radford et al. [13] method to compose music with sentiment, we also need a dataset of MIDI files to train the LSTM and another one to train the logistic regression. There are many good datasets of music in MIDI format in the literature. However, to the best of our knowledge, none are labelled according to sentiment. Thus, we created a new dataset called VGMIDI which is composed of 823 pieces extracted from video game soundtracks in MIDI format. We choose video game soundtracks because they are normally composed to keep the player in a certain affective state and thus they are less subjective pieces. All the pieces are piano arrangements of the soundtracks and they vary in length from 26 seconds to 3 minutes. Among these pieces, 95 are annotated according to a 2-dimensional model that represents emotion using a valence-arousal pair. Valence indicates positive versus negative emotion, and arousal indicates emotional intensity [17].

We use this valence-arousal model because it allows continuous annotation of music and because of its flexibility—one can directly map a valence-arousal (v-a) pair to a multiclass (happy, sad, surprise, etc) or a binary (positive/negative) model. Thus, the same set of labelled data permits the investigation of affective algorithmic music composition as both a classification (multiclass and/or binary) and as a regression problem. The valence-arousal model is also one of the most common dimensional models used to label emotion in music [17].

Annotating a piece according to the v-a model consists of continuously listening to the piece and deciding what valence-arousal pair best represents the emotion of that piece in each moment, producing a time-series of v-a pairs. This task is subjective, hence there is no single "correct" time-series for a given piece. Thus, we decided to label the pieces by asking several human subjects to listen to the pieces and then considering the average time-series as the ground truth. This process was conducted online via Amazon Mechanical Turk, where each piece was annotated by 30 subjects using a web-based tool we designed specifically for this task. Each subject annotated 2 pieces out of 95, and got rewarded USD $0.50 for performing this task.

4.1 Annotation Tool and Data Collection

The tool we designed to annotate the video game soundtracks in MIDI format is composed of five steps, each one being a single web-page. These steps are based on the methodology proposed by Soleymani et al. [17] for annotating music pieces in audio waveform. First, participants are introduced to the annotation task with a short description explaining the goal of the task and how long it should take in average. Second, they are presented to the definitions of valence and arousal. In the same page, they are asked to play two short pieces and indicate whether arousal and valence are increasing or decreasing. Moreover, we ask the annotators to write two to three sentences describing the short pieces they listened to. This page is intended to measure their understanding of the valence-arousal model and willingness to perform the task. Third, a video tutorial was made available to the annotators explaining how to use the annotation tool. Fourth, annotators are exposed to the main annotation page.

This main page has two phases: calibration and annotation. In the calibration phase, annotators listen to the first 15 seconds of the piece in order to get used to it and to define the starting point of the annotation circle. In the anno-

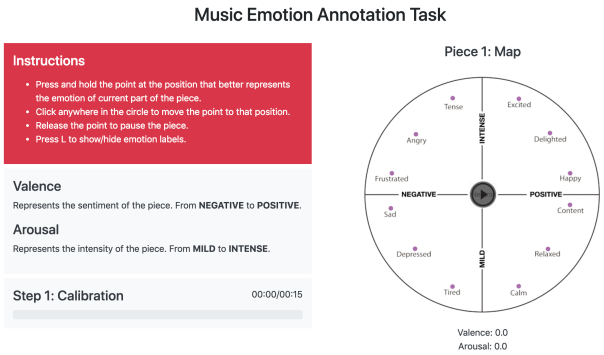


Figure 2: Screenshot of the annotation tool.

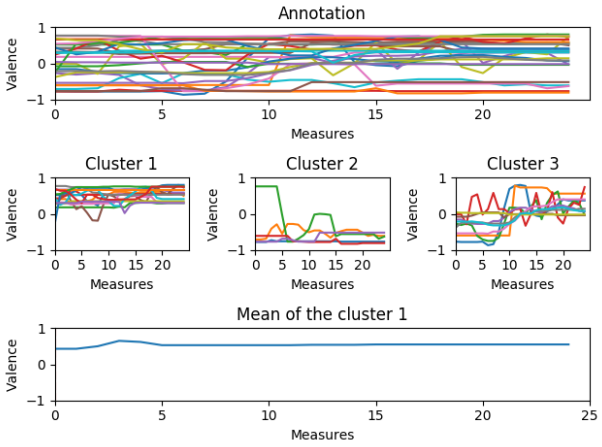


Figure 3: Data analysis process used to define the final label of the phrases of a piece.

tation phase they listen to the piece from beginning to end and label it using the annotation circle, which starts at the point defined during the calibration phase. Figure 2 shows the annotation interface for valence and arousal, where annotators click and hold the circle (with the play icon) inside the v-a model (outer circle) indicating the current emotion of the piece. In order to maximize annotators' engagement in the task, the piece is only played while they maintain a click on the play circle. In addition, basic instructions on how to use the tool are showed to the participants along with the definitions of valence and arousal. A progression bar is also showed to the annotators so they know how far they are from completing each phase. This last step (calibration and annotation) is repeated for a second piece. All of the pieces the annotators listened to are MIDI files synthesized with the "Yamaha C5 Grand" soundfont. Finally, after the main annotation step, participants provide demographic information including gender, age, location (country), musicianship experience and whether they previously knew the pieces they annotated.

4.2 Data Analysis

The annotation task was performed by 1425 annotators, where 55% are female and 42% are male. The other 3% classified themselves as transgender female, transgender male, genderqueer or choose not to disclose their gender. All annotators are from the United States and have an average age of approximately 31 years. Musicianship experience was assessed using a 5-point Likert scale where 1 means "I've never studied music theory or practice" and 5 means "I have an undergraduate degree in music". The average musicianship experience is 2.28. They spent on average 12 minutes and 6 seconds to annotate the 2 pieces.

The data collection process provides a time series of valence-arousal values for each piece, however to create a music sentiment dataset we only need the valence dimension, which encodes negative and positive sentiment. Thus, we consider that each piece has 30 time-series of valence values. The annotation of each piece was preprocessed, summarized into one time-series and split into "phrases" of same sentiment. The preprocessing is intended to remove noise caused by subjects performing the task randomly to get the reward as fast as possible. The data was preprocessed by smoothing each annotation with moving average and clustering all 30 time-series into 3 clusters (positive, negative and noise) according to the dynamic time-warping distance metric.

We consider the cluster with the highest variance to be noise cluster and so we discard it. The cluster with more time series among the two remaining ones is then selected and summarized by the mean of its time series. We split this mean into several segments with the same sentiment. This is performed by splitting the mean at all the points where the valence changes from positive to negative or vice-versa. Thus, all chunks with negative valence are considered phrases with negative sentiment and the ones with positive valence are positive phrases. Figure 3 shows an example of this three-steps process performed on a piece. All the phrases that had no notes (i.e. silence phrases) were removed. This process created a total of 966 phrases: 599 positive and 367 negative.

5. SENTIMENT ANALYSIS EVALUATION

To evaluate the sentiment classification accuracy of our method (generative mLSTM + logistic regression), we compare it to a baseline method which is a traditional classification mLSTM trained in a supervised way. Our method uses unlabelled MIDI pieces to train a generative mLSTM to predict the next word in a sequence. An additional logistic regression uses the hidden states of the generative mLSTM to encode the labelled MIDI phrases and then predict sentiment. The baseline method uses only labelled MIDI phrases to train a classification mLSTM to predict the sentiment for the phrase.

The unlabelled pieces used to train the generative mLSTM were transformed in order to create additional training examples, following the methodology of Oore et al. [11]. The transformations consist of time-stretching (making each piece up to 5% faster or slower) and transposition (raising or lowering the pitch of each piece by up to a major third). We then encoded all these pieces and transformations according to our word-based representation (see Section 3.1). Finally, the encoded pieces were shuffled and

90% of them were used for training and 10% for testing. The training set was divided into 3 shards of similar size (approximately 18500 pieces each – 325MB) and the testing set was combined into 1 shard (approximately 5800 pieces – 95MB).

We trained the generative mLSTM with 6 different sizes (number of neurons in the mLSTM layer): 128, 256, 512, 1024, 2048 and 4096. For each size, the generative mLSTM was trained for 4 epochs using the 3 training shards. Weights were updated with the Adam optimizer after processing sequences of 256 words on mini-batches of size 32. The mLSTM hidden and cell states were initialized to zero at the beginning of each shard. They were also persisted across updates to simulate full-backpropagation and allow for the forward propagation of information outside of a given sequence [13]. Each sequence is processed by an embedding layer (which is trained together with the mLSTM layer) with 64 neurons before passing through the mLSTM layer. The learning rate was set to $5 * 10^{-6}$ at the beginning and decayed linearly (after each epoch) to zero over the course of training.

We evaluated each variation of the generative mLSTM with a forward pass on test shard using mini-batches of size 32. Table 1 shows the average [3] cross entropy loss for each variation of the generative mLSTM.

mLSTM Neurons	Average Cross Entropy Loss
128	1.80
256	1.61
512	1.41
1024	1.25
2048	1.15
4096	1.11

Table 1: Average cross entropy loss of the generative mLSTM with different amount of neurons.

The average cross entropy loss decreases as the size of the mLSTM increases, reaching the best result (loss 1.11) when size is equal to 4096. Thus, we used the variation with 4096 neurons to proceed with the sentiment classification experiments.

Following the methodology of Radford et al. [13], we re-encoded each of the 966 labelled phrases using the final cell states (a 4096 dimension vector) of the trained generative mLSTM-4096. The states are calculated by initializing them to zero and processing the phrase word-by-word. We plug a logistic regression into the mLSTM-4096 to turn it into a sentiment classifier. This logistic regression model was trained with regularization "L1" to shrink the least important of the 4096 feature weights to zero. This ends up highlighting the generative mLSTM neurons that contain most of the sentiment signal.

We compared this generative mLSTM + logistic regression approach against our baseline, the supervised mLSTM. This is an mLSTM with exactly the same architecture and size of the generative version, but trained in a fully supervised way. To train this supervised mLSTM, we used the word-based representation of the phrases, but we padded each phrase with silence (the symbol ".") in order to equalize their length. Training parameters (learning rate and decay, epochs, batch size, etc) were the same ones of the the generative mLSTM. It is important to notice that in this case the mini-batches are formed of 32 labelled phrases and not words. We evaluate both methods using a 10-fold cross validation approach, where the test folds have no phrases that appear in the training folds. Table 2 shows the sentiment classification accuracy of both approaches.

Method	Test Accuracy
Gen. mLSTM-4096 + Log. Reg.	89.83 $\pm$3.14
Sup. mLSTM-4096	60.35 $\pm$3.52

Table 2: Average (10-fold cross validation) sentiment classification accuracy of both generative (with logistic regression) and supervised mLSTMs.

The generative mLSTM with logistic regression achieved an accuracy of 89.83%, outperforming the supervised mLSTM by 29.48%. The supervised mLSTM accuracy of 60.35% suggests that the amount of labelled data (966 phrases) was not enough to learn a good mapping between phrases and sentiment. The accuracy of our method shows that the generative mLSTM is capable of learning, in an unsupervised way, a good representation of sentiment in symbolic music.

This is an important result, for two reasons. First, since the higher accuracy of generative mLSTM is derived from using unlabeled data, it will be easier to improve this over time using additional (less expensive) unlabeled data, instead of the supervised mLSTM approach which requires additional (expensive) labeled data. Second, because the generative mLSTM was trained to predict the next word in a sequence, it can be used as a music generator. Since it is combined with a sentiment predictor, it opens up the possibility of generating music consistent with a desired sentiment. We explore this idea in the following section.

6. GENERATIVE EVALUATION

To control the sentiment of the music generated by our mLSTM, we find the subset of neurons that contain the sentiment signal by exploring the weights of the trained logistic regression model. Since each of the 10 generative models derived from the 10 fold splits in Table 2 are themselves a full model, we use the model with the highest accuracy. As shown in Figure 4, the logistic regression trained with regularization "L1" uses 161 neurons out of 4096. Unlike the results of Radford et al. [13], we don't have one single neuron that stores most of the sentiment signal. Instead, we have many neurons contributing in a more balanced way. Therefore, we can't simply change the values of one neuron to control the sentiment of the output music.

We used a Genetic Algorithm (GA) to optimize the weights of the 161 L1 neurons in order to lead our mL-

[3] Each mini-batch reports one loss.

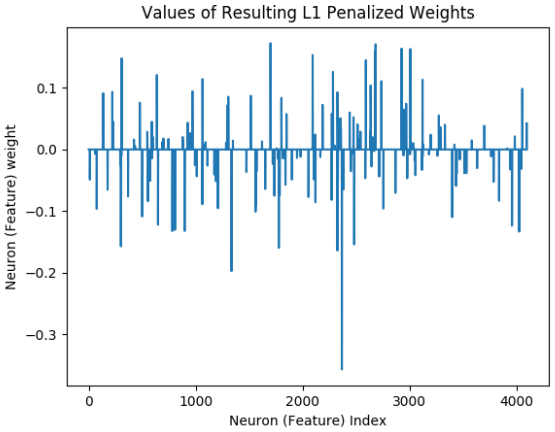


Figure 4: Weights of 161 L1 neurons. Note multiple prominent positive and negative neurons.

STM to generate only positive or negative pieces. Each individual in the population of this GA has 161 real-valued genes representing a small noise to be added to the weights of the 161 L1 neurons. The fitness of an individual is computed by (i) adding the genes of the individual to the weights (vector addition) of the 161 L1 neurons of the generative mLSTM, (ii) generating P pieces with this mLSTM, (iii) using the logistic regression model to predict these P generated pieces and (iv) calculating the mean squared error of the P predictions given a desired sentiment $s \in S = \{0, 1\}$.

The GA starts with a random population of size 100 where each gene of each individual is an uniformly sampled random number $-2 \leq r \leq 2$. For each generation, the GA (i) evaluates the current population, (ii) selects 100 parents via a roulette wheel with elitism, (iii) recombines the parents (crossover) taking the average of their genes and (iv) mutates each new recombined individual (new offspring) by randomly setting each gene to an uniformly sampled random number $-2 \leq r \leq 2$.

We performed two independent executions of this GA, one to optimize the mLSTM for generating positive pieces and another one for negative pieces. Each execution optimized the individuals during 100 epochs with crossover rate of 95% and mutation rate of 10%. To calculate the fitness of each individual, we generated P=30 pieces with 256 words each, starting with the symbol "." (end of time step). The optimization for positive and negative generation resulted in best individuals with fitness 0.16 and 0.33, respectively. This means that if we add the genes of the best individual of the final population to the weights of the generative mLSTM, we generate positive pieces with 84% accuracy and negative pieces with 67% accuracy.

After these two optimization processes, the genes of the best final individual of the positive optimization were added to the weights of the 161 L1 neurons of the trained generative mLSTM. We then generated 30 pieces with 1000 words starting with the symbol "." (end of time step) and randomly selected 3 of them. The same process was repeated using the genes of the best final individual of the negative execution. We asked annotators to label this 6 generated pieces via Amazon MTurk, using the the same methodology described in Section 4.1. Figure 5 shows the average valence per measure of each of the generated pieces.

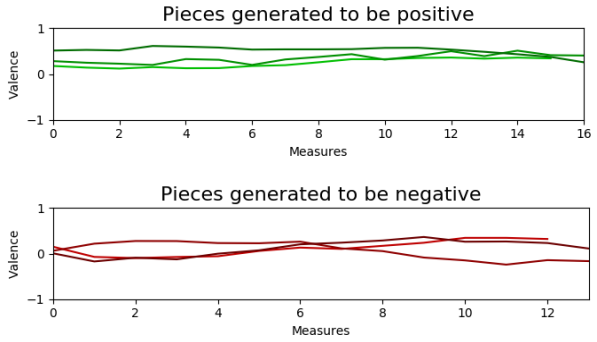


Figure 5: Average valence of the 6 generated pieces, as determined by human annotators. with least variance.

We observe that the human annotators agreed that the three positive generated pieces are indeed positive. The generated negative pieces are more ambiguous, having both negative and positive measures. However, as a whole the negative pieces have lower valence than the positive ones. This suggests that the best negative individual (with fitness 0.33) encountered by the GA wasn't good enough to control the mLSTM to generate complete negative pieces. Moreover, the challenge to optimize the L1 neurons suggests that there are more positive pieces than negative ones in the 3 shards used to train the generative mLSTM.

7. CONCLUSION AND FUTURE WORK

This paper presented a generative mLSTM that can be controlled to generate symbolic music with a given sentiment. The mLSTM is controlled by optimizing the weights of specific neurons that are responsible for the sentiment signal. Such neurons are found plugging a Logistic Regression to the mLSTM and training the Logistic Regression to classify sentiment of symbolic music encoded with the mLSTM hidden states. We evaluated this model both as a generator and as a sentiment classifier. Results showed that our model obtained good classification accuracy, outperforming a equivalent LSTM trained in a fully supervised way. Moreover, a user study showed that humans agree that our model can generate positive and negative music, with the caveat that the negative pieces are more ambiguous.

In the future, we plan to improve our model to generate less ambiguous negative pieces. Another future work consists of expanding the model to generate music with a given emotion (e.g. happy, sad, suspenseful, etc.) as well as with a given valence-arousal pair (real numbers). We also plan to use this model to compose soundtracks in real-time for oral storytelling experiences [12].

8. ACKNOWLEDGMENTS

We would like to thank Dr. Levi Lelis for the great feedback and Dr. Leonardo N. Ferreira for the support on the time series analysis. This research was supported by CNPq (200367/2015-3).

9. REFERENCES

[1] International e-piano competition. `http://www.piano-e-competition.com`. Accessed: 2019-04-12.

[2] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. Deep learning techniques for music generation-a survey. *arXiv preprint arXiv:1709.01620*, 2017.

[3] Sixian Chen, John Bowers, and Abigail Durrant. 'ambient walk': A mobile application for mindful walking with sonification of biophysical data. In *Proceedings of the 2015 British HCI Conference*, British HCI '15, pages 315–315, New York, NY, USA, 2015. ACM.

[4] Hannah Davis and Saif M Mohammad. Generating music from literature. *Proceedings of the 3rd Workshop on Computational Linguistics for Literature (CLfL)*, pages 1–10, 2014.

[5] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. Deepbach: a steerable model for bach chorales generation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1362–1371. JMLR. org, 2017.

[6] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 507–517, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee.

[7] Youngmoo E Kim, Erik M Schmidt, Raymond Migneco, Brandon G Morton, Patrick Richardson, Jeffrey Scott, Jacquelin A Speck, and Douglas Turnbull. Music emotion recognition: A state of the art review. In *Proc. ISMIR*, volume 86, pages 937–952. Citeseer, 2010.

[8] Ben Krause, Iain Murray, Steve Renals, and Liang Lu. Multiplicative LSTM for sequence modelling. *ICLR Workshop track*, 2017.

[9] Eduardo R Miranda, Wendy L Magee, John J Wilson, Joel Eaton, and Ramaswamy Palaniappan. Brain-computer music interfacing (bcmi): from basic research to the real world of special needs. *Music & Medicine*, 3(3):134–140, 2011.

[10] Kristine Monteith, Tony R Martinez, and Dan Ventura. Automatic generation of music for inducing emotive response. In *International Conference on Computational Creativity*, pages 140–149, 2010.

[11] Sageev Oore, Ian Simon, Sander Dieleman, and Doug Eck. Learning to create piano performances. In *NIPS 2017 Workshop on Machine Learning for Creativity and Design*, 2017.

[12] Rafael R Padovani, Lucas N Ferreira, and Levi HS Lelis. Bardo: Emotion-based music recommendation for tabletop role-playing games. In *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2017.

[13] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

[14] Adam Roberts, Jesse Engel, and Douglas Eck, editors. *Hierarchical Variational Autoencoders for Music*, 2017.

[15] James A Russell. A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161, 1980.

[16] Marco Scirea, Julian Togelius, Peter Eklund, and Sebastian Risi. Affective evolutionary music composition with metacompose. *Genetic Programming and Evolvable Machines*, 18(4):433–465, 2017.

[17] Mohammad Soleymani, Micheal N. Caro, Erik M. Schmidt, Cheng-Ya Sha, and Yi-Hsuan Yang. 1000 songs for emotional analysis of music. In *Proceedings of the 2Nd ACM International Workshop on Crowdsourcing for Multimedia*, CrowdMM '13, pages 1–6, New York, NY, USA, 2013. ACM.

[18] Duncan Williams, Alexis Kirke, Joel Eaton, Eduardo Miranda, Ian Daly, James Hallowell, Etienne Roesch, Faustina Hwang, and Slawomir J Nasuto. Dynamic game soundtrack generation in response to a continuously varying emotional trajectory. In *Audio Engineering Society Conference: 56th International Conference: Audio for Games*. Audio Engineering Society, 2015.

[19] Duncan Williams, Alexis Kirke, Eduardo R Miranda, Etienne Roesch, Ian Daly, and Slawomir Nasuto. Investigating affect in algorithmic composition systems. *Psychology of Music*, 43(6):831–854, 2015.

BACKTRACKING SEARCH HEURISTICS FOR SOLVING THE ALL-PARTITION ARRAY PROBLEM

Brian Bemman
Aalborg University
Aalborg, Denmark
bb@create.aau.dk

David Meredith
Aalborg University
Aalborg, Denmark
dave@create.aau.dk

ABSTRACT

Recent efforts to model the compositional processes of Milton Babbitt have yielded a number of computationally challenging problems. One of these problems, known as the *all-partition array problem*, is a particularly hard variant of set covering, and several different approaches, including mathematical optimization, constraint satisfaction, and greedy backtracking, have been proposed for solving it. Of these previous approaches, only constraint programming has led to a successful solution. Unfortunately, this solution is expensive in terms of computation time. We present here two new search heuristics and a modification to a previously proposed heuristic, that, when applied to a greedy backtracking algorithm, allow the all-partition array problem to be solved in a practical running time. We demonstrate the success of our heuristics by solving for three different instances of the problem found in Babbitt's music, including one previously solved with constraint programming and one Babbitt himself was unable to solve. Use of the new heuristics allows each instance of the problem to be solved more quickly than was possible with previous approaches.

1. INTRODUCTION

Milton Babbitt (1916–2011) was a composer of serial music, whose work constituted a substantial contribution to 12-tone music theory and composition [2–6]. His works and the compositional techniques he developed have been studied extensively by music theorists and noted for their complexity [10, 13, 15, 18]. Recent computational work has shed further light on this complexity by looking, in particular, at a 12-tone structure and method of composition that Babbitt developed, known as the *all-partition array* [7, 9, 19, 20].

Satisfying all the constraints necessary to construct an all-partition array is challenging, not least because it involves solving a difficult variant of the set-cover problem [7, 8, 12]. In this paper, we present an improvement to a previous method, based on greedy backtracking, that applies two new search heuristics and a modification of a third heuristic that was originally proposed in [9]. These heuristics aim to facilitate the satisfaction of the most challenging of the all-partition array constraints when using a backtracking algorithm (to be discussed in section 4).

In section 2, we review the all-partition array and follow this with an overview in section 3 of recent computational work on solving the all-partition array problem. We focus in particular on the procedural greedy backtracking approach proposed in [9] to which the heuristics proposed in this paper have been applied. In section 4, we present our new heuristics and give pseudo-code for one possible implementation. In section 5, we demonstrate the effectiveness of the new heuristics by using them to discover solutions to three instances of the all-partition array problem. We report and compare the average running times as well as the number of required backtracks for these three solutions. We also provide a complete solution to one of these instances as evidence of correctness. Finally, in section 6, we summarize our results and propose some possible directions for future work.

2. THE ALL-PARTITION ARRAY

Constructing an all-partition array starts with an $I \times J$ matrix, A, in which each entry is an integer between 0 and 11, representing a pitch class, and where each row contains $J/12$ twelve-tone rows. In this paper, we focus on matrices where $I = 6$ and $J = 96$ because these figure prominently in Babbitt's music [14], and so far have proved difficult to generate [7, 9]. The resulting set of 48 tone rows in the matrix must be closed under any combination of transposition, inversion and retrograde. [1] Matrix A will therefore contain 48 occurrences of each of the integers from 0 to 11. It is important to note that not all organizations of these pitch classes in A will prove successful in constructing an all-partition array, but a desirable trait is for pitch classes to

[1] The exact instance of the type of 6×96 matrix shown throughout this paper has the following form:

$$A = \begin{bmatrix} \mathrm{R}_5 & \mathrm{I}_4 & \mathrm{RI}_7 & \mathrm{P}_2 & \mathrm{R}_{11} & \mathrm{P}_8 & \mathrm{RI}_1 & \mathrm{I}_{10} \\ \mathrm{RI}_4 & \mathrm{P}_{11} & \mathrm{R}_2 & \mathrm{I}_1 & \mathrm{RI}_{10} & \mathrm{I}_7 & \mathrm{R}_8 & \mathrm{P}_5 \\ \mathrm{P}_3 & \mathrm{R}_0 & \mathrm{I}_{11} & \mathrm{RI}_8 & \mathrm{P}_9 & \mathrm{R}_6 & \mathrm{I}_5 & \mathrm{RI}_2 \\ \mathrm{RI}_5 & \mathrm{I}_2 & \mathrm{R}_3 & \mathrm{P}_0 & \mathrm{RI}_{11} & \mathrm{I}_8 & \mathrm{R}_9 & \mathrm{P}_6 \\ \mathrm{I}_6 & \mathrm{R}_1 & \mathrm{P}_4 & \mathrm{RI}_3 & \mathrm{I}_0 & \mathrm{RI}_9 & \mathrm{P}_{10} & \mathrm{R}_7 \\ \mathrm{P}_7 & \mathrm{R}_{10} & \mathrm{I}_9 & \mathrm{R}_{10} & \mathrm{P}_1 & \mathrm{R}_4 & \mathrm{I}_3 & \mathrm{RI}_6 \end{bmatrix},$$

where $\mathrm{P}_0 = \langle 0, 1, 6, 8, 2, 7, 10, 11, 3, 5, 4, 9 \rangle$.

1	2	3	4	5	6	7	8	9	10	11	12
2	9	10	8	4	3	0	7	1	11	6	5
7	0	11	1	5	6	9	2	8	10	3	4
3	4	9	11	5	10	1	2	6	8	7	0
8	1	0	2	6	7	10	3	9	11	4	5
6	5	0	10	4	11	8	7	3	1	2	9
7	8	1	3	9	2	5	6	10	0	11	4

Figure 1: A 6×12 excerpt from a 6×96 pitch-class matrix with a single region defined by a partition (in dark grey) whose "shape" is represented as the integer composition, $\mathrm{IntComp}_{12}(3, 3, 2, 2, 2, 0)$.

1	2	3	4	5	6	7	8	9	10	11	12
2	9	10	8	4	3	0	7	1	11	6	5
7	0	11	1	5	6	9	2	8	10	3	4
3	4	9	11	5	10	1	2	6	8	7	0
8	1	0	2	6	7	10	3	9	11	4	5
6	5	0	10	4	11	8	7	3	1	2	9
7	8	1	3	9	2	5	6	10	0	11	4

Figure 2: A 6×12 excerpt from a 6×96 pitch-class matrix with two regions defined by distinct integer partitions. Note that the region formed by the second composition, $\mathrm{IntComp}_{12}(1, 0, 4, 3, 0, 4)$ (in light grey), overlaps two locations (rows 1 and 3) from the first region.

be evenly distributed [7]. However, the exact organizations which prove successful are still unknown [16, p. 284].

As shown in [20], a complete all-partition array is a *covering* of matrix, A, by K sets, each of which is a partition of the set $\{0, 1, \ldots, 11\}$ whose parts (1) contain consecutive row elements from A and (2) have cardinalities equal to the summands in one of the K distinct integer partitions of $L = 12$ (e.g., $6+6$ or $5+4+2+1$) containing I or fewer unordered summands greater than zero. [2] Figure 1 shows a 6×12 excerpt from a 6×96 pitch-class matrix, A, and one set forming a region in A containing every pitch class exactly once and corresponding to an integer partition, whose exact "shape" is more precisely represented as the *integer composition*, $\mathrm{IntComp}_{12}(3, 3, 2, 2, 2, 0)$ [9]. [3]

There are a total of 58 distinct integer partitions of 12 into 6 or fewer non-zero summands [14]. This means that the number of pitch classes required of an all-partition array having K regions, exceeds the number of entries in its matrix by $(K \cdot 12) - (I \cdot J)$. When $I = 6$ and $J = 96$, this difference is 120. On the musical surface, these 120 additionally required pitch classes are found through horizontal repetitions of at most one in each row from any one contiguous region to the next. As shown in [20], these horizontal row repetitions can be found instead through horizontal overlaps. These overlaps greatly simplify any computational model for generating an all-partition array because the matrix can remain fixed in size.

Figure 2 shows the same 6×12 excerpt from Figure 1, now with a second region corresponding to a distinct partition defined by the integer composition, $\mathrm{IntComp}_{12}(1, 0, 4, 3, 0, 4)$ (in light grey), with two overlaps shared with the first region. Note how the second region (in light grey) formed by its composition shares two overlapped locations (in rows 1 and 3) which lie at the rightmost column of the first region. A complete all-partition array of the type considered here is based on an $I = 6$ and $J = 96$ matrix containing $K = 58$ distinct regions with 120 overlaps.

The constraints of the all-partition array are motivated by Babbitt's desire for *maximal diversity* [14], which is the exhaustive presentation of as many musical parameters as possible (e.g., all 12 pitch classes, 48 tone rows, and K partitions). This makes the all-partition array problem appropriate for methods used in combinatorial optimization or constraint satisfaction, which often rely on either maximizing some objective function or strictly satisfying a set of constraints, respectively.

3. PREVIOUS COMPUTATIONAL WORK ON THE ALL-PARTITION ARRAY PROBLEM

Efforts to solve the all-partition array problem have resulted in a number of different approaches [9, 19, 20]. In [20], an integer programming (IP) model expressed the problem as a set of linear (in)equalities and managed to solve significantly smaller instances of the problem for matrices with six rows and up to 24 columns (requiring $(J + 2)/2$ regions which fix the number of overlaps to be 12). Solving for a matrix of this size required in excess of 30 minutes (Gurobi Optimizer v6.0 solver [1] running on a 2 GHz Intel Core i7 laptop with 8 GB RAM), and the computational time drastically increased with an increase in the number of columns. This suggests that solving for larger matrix sizes, such as the ones considered in this paper, would prove intractable for this IP model.

The first computational method to automatically generate an all-partition array from a pitch class matrix was a constraint programming (CP) model, described in [19]. This model solved a 4×96 matrix (requiring 34 regions and 24 overlaps) by splitting the whole of the matrix in half and solving each smaller sub-matrix before re-joining them to form a complete solution. Solving the second sub-matrix was made easier by discovering certain "easy-to-find" partitions and excluding these from the first sub-matrix. Finding a solution still required over 30 minutes (Sugar v2-1-0 solver [17] running on a 2 GHz Intel Core i7 laptop with 8 GB RAM) and this method of splitting the matrix unfortunately excludes possible solutions. Moreover, it is still unclear whether this model could be used to efficiently solve other instances of the all-partition array problem.

The heuristics proposed in this paper are intended to be used with the procedural greedy backtracking algorithm originally proposed in [9]. Figure 3 gives pseudo-code for a simplified version of the main backtracking procedure originally presented in [9].

[2] As in [20], we denote an integer partition of a positive integer, L, by $\mathrm{IntPart}_L(s_1, s_2, \ldots, s_I)$ and define it to be an ordered set of non-negative integers, $\langle s_1, s_2, \ldots, s_I \rangle$, where $L = \sum_{i=1}^{I} s_i$ and $s_1 \geq s_2 \geq \cdots \geq s_I$.

[3] As in [20], we define an *integer composition* of a positive integer, L, denoted by $\mathrm{IntComp}_L(s_1, s_2, \ldots, s_I)$, to be an ordered set of I non-negative integers, $\langle s_1, s_2, \ldots, s_I \rangle$, where $L = \sum_{i=1}^{I} s_i$. Unlike an integer partition, however, the summands in an integer composition need not be in descending order of size.

```
BACKTRACKINGBABBITT()
1   C ← ⨁_{i=1}^{K} ⟨⟨⟩⟩        ▶ Lists of candidates
2   k ← 1
3   while 0 < k ≤ K
4     if C[k] is empty
5       P ← FINDUNUSEDPARTITIONS(...)
6       C[k] ← FINDCANDIDATES(P, ...)
7       if C[k] is empty
8         k ← k − 1        ▶ Backtrack
9       else
10        k ← k + 1        ▶ Proceed
11    else        ▶ Backtracked to previously visited k
12      Select next overlaps for current candidate in C[k]
13      if current overlaps for current candidate is nil
14        Current candidate becomes next candidate in C[k]
15        if current candidate is nil
16          C[k] ← ⟨⟩        ▶ Make empty
17          k ← k − 1
18        else
19          Select first overlaps (if any) for current candidate
20          k ← k + 1
21      else
22        k ← k + 1
23  return C
```

Figure 3: Pseudo-code for a simplified version of the BACKTRACKINGBABBITT algorithm originally presented in [9]. Note that $\bigoplus_{i=1}^{n}\langle x_i\rangle = \langle x_1, x_2, \ldots x_n, \rangle$, the assignment operator is denoted by '←' and scoping is indicated by indentation.

The algorithm shown in Figure 3 works from left to right in a given matrix. For each region, indexed by k, it first finds the partitions that have not yet been used in positions 1 to k (line 5) and then finds a list of candidate compositions (i.e., $\mathbf{C}[k]$) from these unused partitions (line 6) that form valid regions in the matrix according to the constraints discussed in section 2. If there are no candidate compositions, the algorithm backtracks by decrementing k by 1 (lines 7–8), otherwise it proceeds by incrementing k by 1 (line 10). In the event that the algorithm has backtracked to a previously visited k containing candidates (line 11), the next set of overlaps for the current candidate in $\mathbf{C}[k]$ is chosen (line 12). If there are no overlaps remaining (line 13) then the next candidate in $\mathbf{C}[k]$ is chosen. The algorithm then backtracks if there are no remaining candidates (lines 16–17) or proceeds after selecting the first set of overlaps (if any are available) for this new current candidate (lines 19–20). Note that we have provided here only those details necessary for understanding how our heuristics have been implemented (see section 4). [4]

Even with the help of the search heuristics described in [9], the algorithm above proved unable to generate a complete solution to the all-partition array problem after $100,000$ backtracks when tested on a 6×96 matrix. In the following section, we discuss why this failure likely occurred.

[4] More detailed pseudo-code for the original backtracking algorithm can be found in [9] and an improved implementation (written in Julia v.1.1.0 [11]) with our proposed heuristics can be found in the following repository: `https://github.com/brianm2b/generate-all-partition-arrays`.

4. PROPOSED BACKTRACKING SEARCH HEURISTICS

As noted in [9], a greedy deterministic backtracking algorithm for solving the all-partition array problem based on a depth-first search procedure which finds candidate regions from left to right in a given matrix will incur considerable computational cost in terms of time without sufficiently good heuristics for limiting the amount of backtracking required. When attempts were made to use this algorithm to solve for a 6×96 matrix, much of the backtracking was concentrated towards the far right of the matrix after the algorithm had already successfully discovered most of the required partitions. This means that the algorithm was unable to use the few remaining unused partitions to find compositions capable of forming successful regions (according to the constraints discussed in section 2) from the remaining matrix elements. If one relaxes the constraint that missing pitch classes from the final region must come from overlaps from previous contiguous regions and, instead, allows these pitch classes to be added to the end of the matrix, then the problem becomes significantly more tractable. As it turns out, this is exactly what Babbitt was forced to do in many of his works [7, 13]. It is these matrices, found for example in Babbitt's *About Time* (1982) and *Arie da Capo* (1974), that are the focus of this paper, as no known solution which satisfies all constraints exists. [5]

Figure 4 illustrates two scenarios for forming a final region in a nearly complete all-partition array where only one partition remains to be used. In Figure 4(b) the final region fails to cover one entry in the matrix due to an overlap of pitch class 9 (row 2) with the previous region. In Figure 4(c), every entry is covered, but pitch class 9 is added to the end (row 3) instead of overlapping with the previous region.

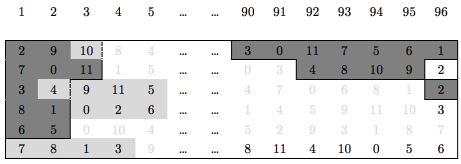
4.1 First Proposed Search Heuristic

The first of our proposed search heuristics is based on a simple assumption regarding the difficulty of finding a final region noted above. By excluding from the left-to-right search those partitions that successfully form regions in the final position K at the far right-hand side of A, it will be easier not only to form a complete matrix covering (Figure 4(c)), but also to avoid violating the constraint that all missing pitch classes from any one region must be overlaps (Figure 4(b)).

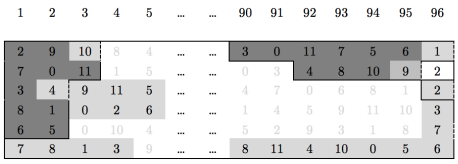
Let us suppose P_k, $1 \leq k \leq K$, is the set of all unused partitions not found in the sequence of selected candidate compositions from 1 up to k and R is the set of partitions that have been found to successfully form regions at K. The modified set of all unused partitions, P'_k, from 1 up to k is given by

$$P'_k = \begin{cases} P_k \setminus R, \text{if } (|P_k \cap R| = r \wedge \\ \qquad\qquad |P_k| > r); \text{and} \\ P_k, \text{otherwise,} \end{cases} \tag{1}$$

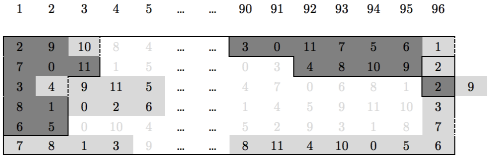
[5] See [13] for an example of an all-partition array of this type that forms an incomplete solution to the all-partition array problem.



(a) A nearly covered matrix and complete all-partition array with 9 uncovered pitch classes and one unused partition, $\mathrm{IntPart}_{12}(7,1,1,1,1,1)$, remaining.



(b) An unsuccessful matrix covering where a single pitch-class 2 remains uncovered by the final region. Note that all missing pitch classes from the uncovered elements in (a) are overlaps with previous contiguous regions.



(c) A successful matrix covering by the final region where pitch-class 2 from (b) has been covered, requiring that the missing pitch class 9 is added to the end of the matrix instead of overlapping with the previous region (as in (b)).

Figure 4: A 6×96 pitch-class matrix with 57 of its $K = 58$ partitions in (a) and two possible ways in (b) and (c) to ensure that the final unused partition, $\mathrm{IntComp}_{12}(1,1,1,1,1,7)$ (in light grey), forms a region containing every pitch class exactly once—both of which result in an incomplete solution to the all-partition array problem. For clarity, greyed out pitch classes belong to regions formed by partitions that have not been shown.

where r, $0 \leq r \leq |R|$, denotes a specified number of partitions in R to exclude from P_k. The first case in Eqn (1) states that the modified set of all unused partitions, P'_k, is the set difference of all unused partitions from 1 up to k and those found in R when (1) the number of partitions at the intersection of P_k and R is equal to r, and (2) the number of unused partitions is greater than r. When either of these two conditions are not met, P'_k is simply all unused partitions remaining (i.e., P_k). Collectively, these cases allow for partitions from R to be freely chosen so long as at least r partitions from R remain unused until there are r regions left to be found.

4.2 Second Proposed Search Heuristic

The second of our proposed heuristics is based in part on a modification to one originally proposed in [9]. A central feature of both heuristics, however, is that they work on the assumption that, since the matrix from which an all-partition array is constructed is regular (i.e., not ragged), regions should be chosen at each k so as to minimize the "raggedness" of their right hand column locations in each row. We make two significant improvements to this original heuristic which allow us to (1) work with a modification of the problem in which additionally required pitch classes appear as overlaps and not horizontal repetitions [20] and (2) minimize the raggedness of regions at each k in a way which takes into account both how far off from and in which direction their right hand column locations are from "ideal" locations specified in (1) while correcting for this same error found in the previous $k-1$ region.

For the first of our improvements, let us suppose we have a list of candidate compositions, $C_k = \langle c_{k,1}, c_{k,2}, ...c_{k,N} \rangle$ (corresponding to e.g., line 6 in Figure 3), at position k, where $1 \leq k \leq K$. Ideally, after choosing a composition for k, the rightmost column location of each row in this composition's region would be $J \cdot k/K$. This rightmost column location *after* choosing $c_{k,n}$ from C_k we denote $l'_{k,n,i}$ for a matrix row, i. For example, if we let $L'_{k,n}$ be equal to $\langle l'_{k,n,1}, \ldots, l'_{k,n,I} \rangle$, then the second region shown in Figure 2 would be $L'_{k,n} = \langle 3,3,5,5,2,4 \rangle$. To measure the raggedness or degree of difference in a region's rightmost column locations, $D_{k,n}$, that results from choosing $c_{k,n}$ in a fixed-size matrix, we use the following formula, based on city-block distance:

$$D_{k,n} = \sum_{i=1}^{I} \left| l'_{k,n,i} - \tfrac{J \cdot k}{K} \right|, \qquad (2)$$

where $|x|$ denotes the absolute value of x. The term, $J \cdot k/K$, specifies for any given region and row at k an "ideal" column location in a fixed-size matrix containing overlaps rather than in a potentially ragged matrix containing horizontal repetitions as in the original construction of an all-partition array. [6] This modification simplifies the modeling of the problem and aligns it with other proposed models [19, 20].

Our second improvement is based on the observation that Eqn (2) computes the magnitude and not the *direction* of the difference in each row between a region's right hand column location and its ideal location. This means, for example, that it is not possible to distinguish between two regions that are equally ragged according to $D_{k,n}$ but where one may be short of its ideal column locations and the other is longer. For this reason, we propose the use of an *adjustment* which captures for a given k how far off and in which direction the region chosen at $k-1$ is from its ideal column locations defined by the right hand term in Eqn (2). We can express this adjustment by defining for a region at k the rightmost column location for a matrix row, i, *before* choosing $c_{k,n}$ from C_k, which we denote $l_{k,i}$. For example, if we let $L_{k,i}$ be equal to $\langle l_{k,1}, \ldots, l_{k,I} \rangle$, then, for the second region shown in Figure 2, $L_{k,i} = \langle 3,3,2,2,2,0 \rangle$. Our second heuristic then is given by

$$S_{k,n} = \sum_{i=1}^{I} \left| \left(l'_{k,n,i} - \tfrac{J \cdot k}{K} \right) + \left(l_{k,i} - \tfrac{J \cdot (k-1)}{K} \right) \right|, \qquad (3)$$

where the adjustment, expressed as the difference $l_{k,i} - \frac{J \cdot (k-1)}{K}$, has been added to the difference shown in Eqn (2).

[6] In [9], this "ideal" column location was expressed as $12k/n$, where n is the number of matrix rows, due to the use of horizontal repetitions.

The absolute value of this sum is then taken and the resulting positive value is summed over all rows of i to form the final measure of adjusted raggedness, $S_{k,n}$. Use of Eqn (3) has the effect of preventing the accumulation of regions having the same directional error, either too short or too far past the ideal column locations for each row.

4.3 Modified backtracking algorithm with improved search heuristics

Whether only the magnitude of difference from an ideal location (Eqn (2)), or the magnitude and direction of this difference (Eqn (3)), is used, the goal of both heuristics is to minimize the degree of raggedness in the column locations of any one region formed by a candidate composition at k. We propose using a greedy strategy, for implementation with the backtracking algorithm proposed in [9], in which, for each k, we choose the candidate composition, c_n, in C_k that minimizes either $D_{k,n}$ or $S_{k,n}$. As a given region may have more than one possible set of overlaps (or none), we sort each region's sets of overlaps for each c_n according to whichever set results in the smallest value for the single heuristic, either $D_{k,n}$ or $S_{k,n}$, used globally throughout the search. The first of our heuristics shown in Eqn (1) can be implemented in two parts: one which occurs before the backtracking search begins and the other during the search when unused partitions from 1 up to k are found. Our two other heuristics shown in Eqn (2) and Eqn (3) can be implemented simply during the search when sorting the list of discovered candidate compositions in each C_k.

In Figure 5, asterisks indicate our modifications to the original backtracking algorithm in Figure 3 required to implement the new heuristics. Prior to the start of the backtracking search, the FINDCANDIDATES function finds the set of partitions, $\mathbf{R}$, that prove successful in forming regions at K, as required in Eqn (1) (see line 2 in Figure 5). These partitions are then passed to the function for finding unused partitions from 1 up to k in line 6, which returns a sorted set of lexicographically ordered compositions grouped by partition, $\mathbf{P}'$. In line 7, the pool of candidates for the kth region is chosen from this returned set. In line 11, the set of candidates found at k is sorted according to the value assigned to each candidate by either Eqn (2) or (3) (but not both).

```
BACKTRACKINGBABBITT*()
1   C ← ⊕_{i=1}^{K} ⟨⟨⟩⟩        ▶ Lists of candidates
2*  R ← FINDCANDIDATES(...)
3   k ← 1
4   while 0 < k ≤ K
5     if C[k] is empty
6*      P′ ← FINDUNUSEDPARTITIONS(R, ...)
7*      C[k] ← FINDCANDIDATES(P′, ...)
8       if C[k] is empty
9         k ← k − 1        ▶ Backtrack
10      else
11*       C[k] ← SORTBYHEURISTICS(C[k])
12        k ← k + 1        ▶ Proceed
13    else        ▶ Backtracked to previously visited k
14      Select next overlaps for current candidate in C[k]
15      if current overlaps for current candidate is nil
16        Current candidate becomes next candidate in C[k]
17        if current candidate is nil
18          C[k] ← ⟨⟩        ▶ Make empty
19          k ← k − 1
20        else
21          Select first overlaps (if any) for current candidate
22          k ← k + 1
23      else
24        k ← k + 1
25  return C
```

Figure 5: Pseudo-code for a simplified version of modifications to the BACKTRACKINGBABBITT algorithm originally posed in [9] required to implement our new heuristics, expressed in Eqn (1), Eqn (2), and Eqn (3). The '*' denotes modified lines to the original implementation shown in Figure 3.

5. SOLUTIONS

As evidence of the correctness for the modified backtracking algorithm in Figure 5 and a demonstration of its performance using our proposed heuristics, we solved for three different instances of matrices of two sizes found in Babbitt's works. Table 2 shows the approximate times in seconds and fewest number of backtracks required by our heuristics to solve these three matrices (Julia v.1.1.0 [11] running on a 2.4 GHz Intel Core i5 laptop with 8 GB RAM).

For the purposes of bench marking, all solving times shown in Table 2 were averaged over three runs and terminated after 2×10^7 backtracks if no solution was found. The reported solutions were found by running the algorithm with the given set of heuristics for all values of the parameter, r, from 1 to $|R|$ (where $|x|$ denotes cardinality) and selecting the one which resulted in the fewest number of respective backtracks. The first of these matrices was constructed manually by Babbitt and no known complete solution has existed prior to our solving it here using P' ($r = 11$) and D, and P' ($r = 13 = |R|$) and S—the latter of which resulted in a fewer number of backtracks. The complete solution to this matrix using the best combination of heuristics appears in Table 1 below. The second matrix in Table 2 was constructed manually by a student of Babbitt named David Smalley [9] and at least one known solution (discovered by Smalley) existed prior to our solving it here using P' ($r = 22 = |R|$) and D, and P' ($r = 18$) and S. The final matrix was previously solved in [19] using constraint programming in ≈ 30 minutes, however, the combination of P' ($r = 1$ where $|R| = 7$) and S discovered a solution in ≈ 22 minutes. Overall, this matrix required the highest number of backtracks (over 5 million in the best case) of all matrices tested here. In all cases, we have used P', as solving using either D or S alone proved infeasible within the specified backtracking limit. Similarly, using P' alone also proved unsuccessful.

The significant difference in solving times and number of required backtracks in Table 2 for each of the matrices using one set of heuristics or another is interesting to note. The only difference between the first and second matrices, for example, is their organization of pitch classes, as both are the same size and require the same number of regions and overlaps. However, using P' and D solves the second matrix in approximately 1 second and 2377 backtracks but

	29A843		-30	-071	-1		-1B6	-6	5	43	
	70B	-B	15	-56	9		-92	-28A34B0			
349B5A	1	2	-2		-2	6870	-0		9		
8102	6	-6	-67A	-A39B4	-4	-45	-5	-5	21860743B	-B9A50786	21A
6	5	-50A4	-4,B,8	-8	-873	-3129A	-A		-A		5640B8397
7		-78139	-9	2	56A0B	-B	4387	-791		-12	
641^2	631^3	541^3	$3^2 2^2 1^2$	5321^2	531^4	5421	4321^3	731^2	91^3	82^2	93

-3		A82		9	651		-1B07	-7	-7A32489		
	-0	-05716	9		-9A2438B		6		-6	-67	51094A8
	-9		453BA72	861		-10BA	-10539	4	1	08	67
	5	B943	0168	27AB3		-35492768	-8		0	14935A	B
-721	4						-4	5A06B	-B	-B2	-23
5A46B098	-83172BA6			450	-07		2	-23198	5		
831	81^4	543	741	$53^2 1$	732	84	$4^2 1^4$	$5^2 1^2$	71^5	62^3	$72^2 1$

	-9	-9			05B1	67	-72	38A	-A490		157
321	075	-5		B632A	-A8	-894	-4	165	-5	7B03	824
	2B43	-3			-3		59A160	-027	8	-89	-9A3
		87	2061A	95			3	4B	61	-12	
798	-816	-6BA04	-45	817	-792	-230B	-B			64A5	
-5064BA	-A	12	7938B	04	-46	5A1	8	9	732B		-B60
63^2	$43^2 1^2$	$52^2 1^3$	$5^2 2$	532^2	432^2	$43^2 2$	621^4	$3^3 21$	$4^2 21^2$	$4^2 2^2$	3^4

6B	8342	-2A	96	175	0B8	924	-4A36	-67	-7	-7	
9A	76	-61	-1B	-B		-B5	-5098				
-35	-5	B4780	-02	-2	1	63		AB9	5	-5	
0874	B	53	-3A	-A96		70		-02		-2	
21	-19	-9	78	-830	546A	-A	B271	-13	-38	-89AB460	58913274B0A6
	-0A		54	-4	3297	18		-854	-40AB6921	-13	
42^4	$42^3 1^2$	$52^3 1$	2^6	$3^3 1^3$	$4^2 31$	$32^4 1$	4^3	$3^2 2^3$	821^2	721^3	12

B105498A23	-36B5701A9	-9	-94283		0			-0B7561	
	42	3A	-A501B76		3A429856B17	-70	34	8A92	
		418207		6		54B93A	7620185A9B	34	706812
						-281			459BA3
-6		-65		293187					
7	8	B		-B4A05		6			
10 1^2	921	$62^2 1^2$	75	651	11 1	6321	10 2	642	6^2

Table 1: A generated all-partition array which solves one instance of the all-partition array problem based on a 6×96 matrix having 58 distinct regions and 120 overlaps. Each box contains the elements in A belonging to a region formed by a distinct integer partition, where a dash indicates those that overlap. Note that partitions are denoted using a shorthand notation, e.g., 4^3, where the base indicates the length of a part and the exponent denotes its number of occurrences. The integers 10 and 11 are the letters A and B, respectively.

		P', D			P', S		
	Matrix A	time (s)	backtracks	r	time (s)	backtracks	r
1.	Babbitt$(6, 96)$	176.92	583724	11	3.88	5909	13
2.	Smalley$(6, 96)$	1.25	2377	22	96.03	231933	18
3.	Babbitt$(4, 96)$	3791.94	14224645	1	1321.09	5390388	1

Table 2: Approximate times and fewest number of required backtracks for solving three different matrices using the search heuristics, P', D, and S.

using P' and S is significantly more costly. The reverse is true of these sets of heuristics in the first matrix. This result appears to support the findings reported in [20], that each matrix represents a unique problem space, which, in our case, may require the use of heuristics with different considerations. Contrary to what one might expect, however, the smaller third matrix actually proved more difficult than the larger and more combinatorially expansive matrices (i.e., requiring more regions) with a best-case solving time and number of backtracks roughly 3 orders of magnitude greater. Finally, it is important to note that while the best solving times and number of backtracks for the first two matrices are low, their use of values greater than 1 for the parameter, r, in P' means that some potential solutions are excluded, namely, those in which one or more of these r partitions in P' appear at positions, k, where $k \leq K - r$. This is not true, however, in the third matrix, where only one of the possible partitions in P' was excluded (i.e., $r = 1$).

6. CONCLUSION

In this paper, we provided improvements to an existing backtracking algorithm in the form of three search heuristics which proved successful in solving the all-partition array problem for three different instances found in Babbitt's music. Our findings demonstrate that, when used together, our proposed heuristics allow the backtracking approach to outperform other approaches. However, it is also apparent from our results that the solving time and number of required backtracks required is highly dependent on the specific matrix given as input. In future work, it would prove useful to investigate methods of analyzing the organization of pitch classes in a matrix prior to searching and using these findings to modify the ideal column locations accordingly during the search.

7. REFERENCES

[1] *Gurobi Optimizer Reference Manual version 6.0*. Gurobi Optimization, Inc., Houston, TX, 2015.

[2] Milton Babbitt. Some aspects of twelve-tone composition. *The Score and I.M.A. Magazine*, 12:53–61, 1955.

[3] Milton Babbitt. Twelve-tone invariants as compositional determinants. *Journal of Music Theory*, 46(2):246–259, 1960.

[4] Milton Babbitt. Set structure as a compositional determinant. *Journal of Music Theory*, 5(1):72–94, 1961.

[5] Milton Babbitt. Twelve-tone rhythmic structure and the electronic medium. *Perspectives of New Music*, 1(1):49–79, 1962.

[6] Milton Babbitt. Since Schoenberg. *Perspectives of New Music*, 12(1/2):3–28, 1973.

[7] Brian Bemman. *Computational Problems in Modeling the Compositional Process of Milton Babbitt*. Ph.d. diss., Aalborg University, 2017.

[8] Brian Bemman and David Meredith. Exact cover problem in Milton Babbitt's all-partition array. In Tom Collins, David Meredith, and Anja Volk, editors, *Mathematics and Computation in Music: Fifth International Conference, MCM 2015, London, UK, June 22–25, 2015, Proceedings*, volume 9110 of *Lecture Notes in Artificial Intelligence*, pages 237–242. Springer, Berlin, 2015.

[9] Brian Bemman and David Meredith. Generating Milton Babbitt's all-partition arrays. *Journal of New Music Research*, 45(2):1–21, 2016.

[10] Zachary Bernstein. The problem of completeness in Milton Babbitt's music and thought. *Music Theory Spectrum*, 38(2):241–264, 2017.

[11] J. Bezanson, A. Edelman, S. Karpinski, and V. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[12] Richard M. Karp. Reducibility among combinatorial problems. In Miller R.E., Thatcher J.W., and Bohlinger J.D., editors, *Complexity of Computer Computations*, The IBM Research Symposia Series. Springer, 1972.

[13] Andrew Mead. About About Time's time: A survey of Milton Babbitt's recent rhythmic practice. *Perspectives of New Music*, 25:182–235, 1987.

[14] Andrew Mead. *An Introduction to the Music of Milton Babbitt*. Princeton University Press, Princeton, NJ., 1994.

[15] Andrew Mead. Still being an american composer: Milton Babbitt's at eighty. *Perspectives of New Music*, 35(2):101–126, 1997.

[16] Robert Morris. Mathematics and the twelve-tone system: Past, present, and future. In Noll T. (eds) In: Klouche T., editor, *Mathematics and Computation in Music, MCM 2009 Proceedings*, volume 37 of *Communications in Computer and Information Science*, pages 266–288. Springer, Berlin, Heidelberg, 2009.

[17] Tamura. Naoyuki and Mutsunori Banbara. Sugar: A csp to sat translator based on order encoding. In *Proceedings of the 2nd International CSP Solver Competition*, Communications in Computer and Information Science, pages 65–69. 2008.

[18] Daniel Starr and Robert Morris. A general theory of combinatoriality and the aggregate, part 2. *Perspectives of New Music*, 16(2):50–84, 1978.

[19] Tsubasa Tanaka, Brian Bemman, and David Meredith. Constraint programming approach to the problem of generating Milton Babbitt's all-partition arrays. In Michael Rueher, editor, *International Conference on Principles and Practice of Constraint Programming, CP2016, Toulouse, France, September 5–9, 2016 Proceedings*, volume 9892 of *Lecture Notes in Computer Science*, pages 802–810. Springer, Berlin.

[20] Tsubasa Tanaka, Brian Bemman, and David Meredith. Integer programming formulation of the problem of generating Milton Babbitt's all-partition arrays. In *17th International Society for Music Information Retrieval Conference, August 7–11, 2016, New York, NY.*

MODELING AND LEARNING STRUCTURAL BREAKS IN SONATA FORMS

Laurent Feisthauer **Louis Bigo** **Mathieu Giraud**
CRIStAL, UMR 9189, CNRS, Université de Lille, France
{laurent,louis,mathieu}@algomus.fr

ABSTRACT

Expositions of Sonata Forms are structured towards two cadential goals, one being the Medial Caesura (MC). The MC is a gap in the musical texture between the Transition zone (TR) and the Secondary thematic zone (S). It appears as a climax of energy accumulation initiated by the TR, dividing the Exposition in two parts. We introduce high-level features relevant to formalize this energy gain and to identify MCs. These features concern rhythmic, harmonic and textural aspects of the music and characterize either the MC, its preparation or the texture contrast between TR and S. They are used to train a LSTM neural network on a corpus of 27 movements of string quartets written by Mozart. The model correctly locates the MCs on 14 movements within a leave-one-piece-out validation strategy. We discuss these results and how the network manages to model such structural breaks.

1. INTRODUCTION

1.1 Sonata Form

The classical sonata form shaped many musical works in the classical and the romantic period. It began to appear in the second half of the 18th century but was not formalized until the early 19th century. Recent theories on sonata forms emerged in the last decades, with various points of views [5, 7, 13], but nevertheless agree on its higher-level structure, involving *Exposition*, *Development* and *Recapitulation* sections, and optional *Introduction* or *Coda* sections.

According to Hepokoski and Darcy, an Exposition may be either a *two-part exposition*, featuring two contrasting thematic zones, or a *continuous exposition*, with only one thematic zone [13]. The two-part exposition is characterized by two strong punctuation breaks (Figure 1). The first one is the *Medial Caesura (MC)* which closes the first part of the exposition. The second one is the *Essential Expositional Closure (EEC)*, which is a *Perfect Authentic Cadence (PAC)* that concludes the Secondary thematic zone (S). The lack of a clearly articulated MC is the main difference between the two-part exposition and the continuous exposition [12]. Figure 1 shows the two-part exposition of a sonata form with its two punctuation points.

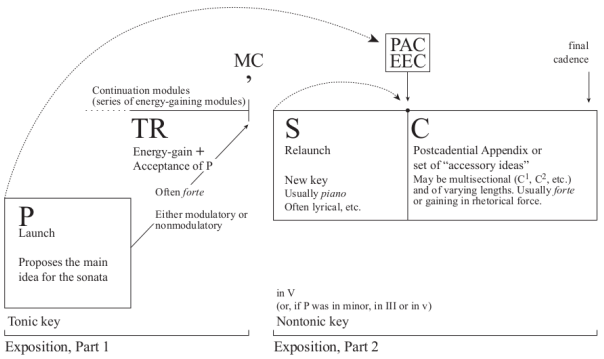


Figure 1. Structure of the two-part exposition in a sonata form, from Hepokoski and Darcy [13].

1.2 Formalizing Medial Caesuras

The MC is a "break in texture" [21] or a "textural change" [12] built around a cadence (most of the time, a half cadence (HC)), that acts as a boundary between TR and S. According to Hepokoski and Darcy, the MC has two fonctions [12, 13]: It closes the first part of the Exposition, concluding a process of energy gain initiated during the TR. It makes the second part available, opening a space for S thanks to that energy accumulated in TR. [1] Whether or not TR is modulatory, the MC is the point when the music reaches a structural dominant. [2] This dominant may be prolonged by neighbor motion, a repeated dominant pedal and a strong *forte*, and may be emphasized by *hammer strokes* (or *hammer blows*), that are repetitions of the final dominant chord. The whole process is concluded by the actual articulation of the MC. There can be a *general pause* on all voices, but there can also be a *caesura-fill*, that is a small melodic pattern or a sustained note or chord that "bridges the gap" to the S zone.Two examples of MCs are presented on Figure 2 and Figure 3.

[1] Hepokoski and Darcy suggest that the MC "may be thought of as metaphorically analogous to the moment of the opening of elevator doors onto a higher floor." [13]

[2] This structural dominant is the *arrival point* of the associated HC [6], what follows being considered as *post-cadential*.

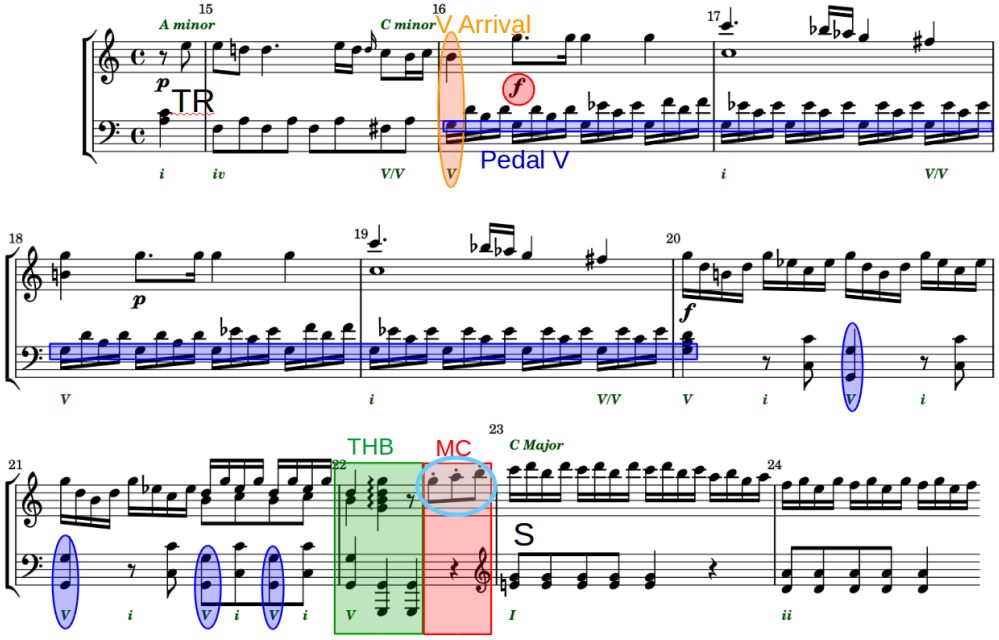


Figure 2. Mozart, *Piano sonata in A minor*, K310, 1st movement, mm15–24. The arrival point of the half cadence is on the downbeat of m16 (circled in orange). The measures 16 to 21 have a prolongational function. They are built on a *dominant pedal* (in blue) and a speeding-up harmonic oscillation between dominant and minor tonic of C minor. The dominant tension is reinforced by the *forte*/*piano*/*forte* contrast at m16, m18, and m20. This leads to a *triple hammer blow* (THB) at m22 (green) and the actual articulation of the MC on the fourth beat of m22 (red). Then *caesura-fill* at the right hand (lighter blue) leads to S and its new thematic unit.

Hepokoski and Darcy classify MCs according to the cadence occuring before the break and their position inside the Exposition [12]. A *first-level default* MC is associated with a HC in the secondary key [3] and can be denoted by V:HC MC (III:HC MC or v:HC MC in minor mode). This MC generally occurs between 25 and 50% of the length of the exposition, and sometimes at 60%. A *second-level default* MC considers a HC in the primary key (I:HC MC, between 15 and 45% of the length). A *third-level default* MC is caracterized by a PAC in the secondary key (V:PAC MC, III:PAC MC, or v:PAC MC), and occurs between 50 and 70%, sometimes at 75% of the length. The least encountered option, *fourth-level default*, is a PAC or an imperfect authentic cadence (IAC) in the primary key (I:PAC MC or IAC MC), generally at the end of P. In this case, S follows P without TR.

Another point of view on MCs is given by Richards [20], who identifies in 2013 seven *signals*, underlying the beginning of the secondary theme (S): tonic harmony of the new key, beginning function, preparation by a phrase-ending chord, textural gap of a medial caesura, change in texture, change in dynamics and characteristic melodic material. Each signal can be encountered in a *strong* or *weak* form. For example, the "Tonic harmony of new key" signal is strong when the chord encountered on the first downbeat is a tonic chord in the secondary key. It is weak when that chord is not the tonic of the secondary key, or when the chord is the tonic of another key due to a temporary modulation.

Somehow, a "textbook" MC, like the one displayed on Figure 2 is heard when all these signals, either on MC or on start of S, are strong. Of course, MCs in the classical repertoire do not strictly follow these rules and are rather heard with many deformations, like the one presented on Figure 3.

1.3 Medial Caesura, Sonata Form and MIR

Working on high-level music structures is a challenge for Music Information Retrival (MIR) research [19]. To our knowledge, no previous study in MIR has specifically targeted MC. However, several authors worked on sonata forms and designed algorithms to model or retrieve parts of its structure, on audio signals [15, 26] and on symbolic data [2, 22]. Sears and colleagues worked especially on cadences, by demonstrating that terminal events from cadential context are the most predictable thanks to a finite-context model (IDyOM) [23]. We previously worked on sonata form structure identification, modeling the MC as one state in Hidden Markov Models with 14 or 18 states and using a Viterbi algorithm to find back the sections from symbolic features [1, 4]. We also worked on PAC and HC detection thanks to features extraction and a SVM model. While PAC detection seems satisfying, HC detection was kind of disappointing due to the lack of characteristic feature for this cadence type [3].

[3] The secondary key is often the dominant major (V) for major mode primary key and the relative major (III) or the dominant minor (v) for minor mode primary key.

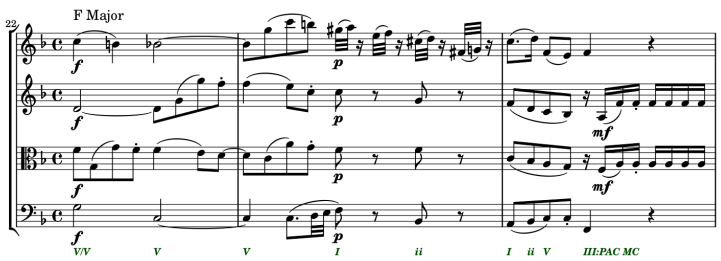


Figure 3. Mozart, *String Quartet in D minor*, K421, 1st movement, mm22-24. This MC is weak: No dominant arrival, no lock on degree V, nor hammer strokes, and energy depletion rather than a gain. This MC weakness can be due to a I:HC MC denial on the third beat of m14 not shown on this figure (dominant arrival at m12). The composer delays the arrival of S and continues into TR, maybe to create surprise.

Modeling MCs is a challenging subject even amongst music theorists. As the MC is a striking event, playing a role in the high-level structure, one may wonder whether it is possible to model and predict MCs with computational musicology methods. This study tries thus to model features relevant to identify structural breaks such as the MC. We propose such features that may be specific to MCs, based on music theorist works [13, 20] (Section 2) and present a neural network model that we train on a corpus of expositions in Mozart string quartets (Section 3). We finally discuss the occurences of the features, detail how the network manages to model the MC, and propose perspectives to further their study (Sections 4 and 5).

2. FEATURES INDICATING THE MEDIAL CAESURA

We mostly here introduce features to model high-level signals leading to the Medial Caesura, taking inspiration from Hepokoski and Darcy as well as from Richards [13,20]. To find the MC, we want to model the MC, but also its preparation and the textural contrast with the beginning of S. We also use low-level features inspired by dedicated extraction software as jSymbolic [18]. This section details 13 *features* that are estimated on each beat of the music piece. The features are then used in the next section to train a neural network to model the MC. In contrast to frequent uses of neural networks that consist in automatically identify most relevant features, this research aims to validate the efficiency of a set of pre-determined theory driven features to model medial caesura.

In the following, given a onset b, the $[b, b+1[$ interval means that we consider each note actually sounding in that interval, including notes whose onset is b (or before, but still sounding on b) but excluding notes whose onset is $b+1$.

2.1 Rhythm, energy and textural features

Preparations of MCs are expected to be passages of high rhythmic intensity as a consequence of the repetition of the pedal and the *forte*. It might even be the most obvious way of gaining musical energy. We also expect a change of rythmic density between TR and S and a beat filled with silence on the articulation of the MC.

Modeling textural changes with precision is a challenging topic in computational musicology [11]. To try to capture these "breaks in texture" between the end of TR and the beginning of S, we implement the following low-level features, for each beat b:

- *f-rhythm-density* counts the number of notes in $[b, b+1[$,
- *f-rest* counts the number of voices not sounding on b.

We add these features, as the textural/energy change can be seen on the range of voices:

- *f-mean-pitch* is the mean of the MIDI values of pitches in $[b, b+1[$,
- *f-range-pitch* is the difference between the maximum and minimum MIDI values of pitches in $[b, b+1[$.

We also keep track of the position in the piece:

- *f-time* is the current beat number divided by the total number of beat in the score.

We propose one high-level feature specific to MCs:

- *f-hammer-blow* tracks double, triple or more hammer strokes (see THB on Figure 2). To estimate this feature, the set of pitches on b are compared to sets of pitches on previous beats. This feature reaches its maximum when the set of pitches is the same for b for at least 2 previous beats.

2.2 Harmonic Features

We expect a (functional) dominant lock during the few beats before the articulation of the MC (whether it is a dominant on primary key or on secondary key).

Moreover, whether a TR is modulatory or not, we expect to encounter accidentals during TR or at least the beginning of S due to neighbor motion during the phase of prolongation of dominant and use of the dominant of the secondary key.

Algorithms for detecting local tonalities work well [17, 24] but tend to average variations over a window (often a few beats or measures). We propose rather two sets of features adapted to the detection of the MC.

Functional harmony compatibility. Functional harmonic analysis is a challenging problem in itself [9, 25]. The idea

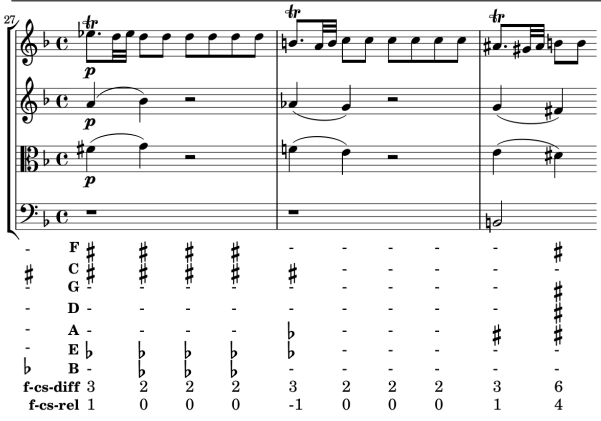


Figure 4. Estimation of the *current diatonic scale* on mm27-29 of Mozart's 1st movement of String Quartet No. 13 in D minor (K173). The D minor harmonic scale is given on the left. **f-cs-rel** stands for *f-current-scale-relative*.

here is to assert how compatible is a current harmony to a harmonic function, but without actually classifying the harmonies.

For a given functional harmony (as for example *f-predominant*, that may be either a ii or a IV/iv), we define a *compatibility profile* $h : P \rightarrow [-1, 1]$ that asserts how compatible a pitch $p \in P$ is to the given harmony. The pitch is given relative to the tonic of the primary key. Given a set of notes $c = \{p_1, p_2, ...\}$ from a given offset, the feature computes $\sum_{p \in c} h(p)$. The actual computation weights notes by their length in the given beat.

The compatibility profile h could be learned as profiles used in tonality detection [17]. We selected here a simpler approach and encoded two pitches lists coming for musical knowledge, one with relevant pitches, the other with irrelevant pitches. We define five such features, each one with a particular list of relevant and irrelevant pitches:

	relevant $h(p) = 1$	irrelevant $h(p) = -1$
f-maj-tonic	1, 3, 5	♯4, 7
f-min-tonic	1, ♭3, 5	♯4, 7
f-predominant	2, 4, ♯4, ♭6, 6, 1, ♭2	3, 5, 7
f-dominant-of-dominant	2, ♯4, ♭6, 6, 1, ♭3	3, 4, 5, ♯5, 7
f-dominant	5, 7, 2, 4, 6, ♭6	1, ♯5

These pitch lists suppose that we have a pitch space with the pitch spelling information. Pitches p that are not listed count for $h(p) = 0$.

Harmony landscape and current scale. To better capture the occurrence of new accidentals, we model the *current scale*. It is a diatonic scale containing the seven pitches with, for each of them, the last accidental encountered (Figure 4). We expect it to be different of its initial state at the beginning of the piece and to vary a lot just before the MC. We estimate two features on this scale :

- *f-current-scale-diff* counts the number of pitches differences in the current scale in $]b - 1, b]$ related to the initial current scale.

	Tonality	Cadence	MC	Tempo
K80.1	G Major	V:HC	1	Adagio
K80.2	G Major	V:HC	1	Allegro
K156.1	G Major	I:HC	2	Presto
K156.2	E minor	III:HC	1	Adagio
K157.1	C Major	I:HC	2	N.A.
K157.2	C minor	III:IAC	4	Andante
K158.1	F Major	V:HC	1	Allegro
K159.1	B♭ Major	V:HC	1	Andante
K159.2	G minor	III:PAC	3	Allegro
K168.1	F Major	I:HC	2	Allegro
K168.2	F minor	V:HC	1	Andante
K169.1	A Major	I:PAC	4	Molto Allegro
K171.3	C minor	v:HC	1	Andante
K171.4	E♭ Major	V:HC	1	Allegro assai
K172.1	B♭ Major	V:HC	1	Allegro spiritoso
K172.2	E♭ Major	V:HC	1	Adagio
K172.4	B♭ Major	V:PAC	3	Allegro assai
K173.1	D minor	v:HC	1	Allegro moderato
K387.1	G Major	V:HC	1	Allegro vivace assai
K421.1	D minor	III:PAC	3	Allegro
K428.1	E♭ Major	V:PAC	3	Allegro non troppo
K428.2	A♭ Major	I:HC	2	Andante con moto
K465.1	C Major	V:HC	1	Adagio + Allegro
K465.4	C Major	V:HC	1	Allegro
K499.3	G Major	V:HC	1	Adagio
K589.1	B♭ Major	V:PAC	3	Allegro
K590.1	F Major	V:HC	1	Allegro moderato

Table 1. The corpus contains 27 expositions (21 in major, 6 in minor) in Mozart String Quartets. "MC" denotes the MC type as designed by [12].

- *f-current-scale-relative* further weights this count by $+1$ when the scale "gains" a sharp (or "loses" a flat) and by -1 in the other case.

For example, on the Figure 4, the current scales are compared with the D minor harmonic scale (primary key, with B♭ and C♯). On the downbeat of measure 28, three pitches are changed (A♭, E♭, B♮), so *f-current-scale-diff* is 3 and *f-current-scale-relative* is $1 - 2 = -1$.

3. LEARNING STRATEGY

3.1 Network Layout

A Long Short-Term Memory neural network (LSTM) is built to predict the position of the Medial Caesura in the pieces of the corpus (Figure 5).

The network takes vectors of values describing the beats of the pieces as input. The identification of a Medial Caesura at a particular beat requires also to look at several past beats and possibly future beats. This is partly taken into account by the LSTM, but we further directly provide to the network the feature values over a time window. A

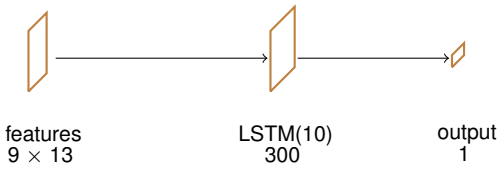


Figure 5. Network layout

vector describing a beat includes thus the 13 feature values corresponding to this beat but also those corresponding to the p previous beats and the n next beats. In this experiment, we set $p = 4$ and $n = 4$ which results in input vectors of size $9 \times 13 = 117$.

The input layer is fully connected to a recurrent hidden layer including 300 units. The time step of the LSTM is set to 10, meaning that 10 consecutive vectors are used to compute the probability of a MC occuring at one beat. Finally, the hidden layer is fully connected to the output layer that is a single unit. A sigmoid function scales the output value as a probability in the interval $[0, 1]$.

3.2 Model Training

To avoid overfitting, the position of the Medial Caesura in a piece of the corpus is predicted with a model that has been trained on the whole corpus minus the piece itself. This is referred as *leave-one-piece-out validation* process.

The 13 features are computed at every beat of every piece in the training set. Each piece is represented as a sequence of *feature vectors* of size 117, each vector being associated with a specific beat.

Every feature vector of the training set is associated with a *label* having a value 1 (presence of an annoted MC in the next 5 beats) or 0 (otherwise). During the training, pairs *(feature vector, label)* are presented to the network by batches of size 200. An Adam optimization algorithm updates the unit weights to minimize a binary cross-entropy loss function over 60 epochs.

Given the small number of medial caesura in the corpus, there was no preliminary selection of a separated test set. This research primarily focuses on the validation of the musical features rather than the classifier itself. For these reasons, the number and sizes of hidden layers, the batch size and the number of epochs as the optimization algorithm were selected among the most common values given the dimension and the quantity of input datas, with minimum optimization process in order to avoid over-fitting.

4. EVALUATION

4.1 Two-part expositions in Mozart's String Quartets

Mozart wrote 23 string quartets totaling 86 movements, including 42 in sonata form [16]. Many of these quartets are encoded as `.krn` Humdrum files [14] that we downloaded from the `humdrum-mozart-quartets` repository at `github.com/musedata/`. Some of these movements were left out because of unavailable clean encodings or of other technical inconsistencies including the absence of Medial Caesura. The corpus used finally contains 27 two-part expositions totaling 4179 beats. Medial Caesura annotations were taken from the sonata form annotation dataset we proposed in [1] and available at `www.algomus.fr/data`. These annotations include P, TR, MC, S, and C sections. Table 1 lists these 27 movements with their MC level default. We denote by K171.4 the 4th movement of K171.

	P	TR	MC	S	C
f-rhythm-density	0.442	0.517	0.524	**0.557**	**0.577**
f-hammer-blow	0.080	0.087	**0.143**	0.070	0.059
f-rest	0.223	0.194	0.099	**0.229**	0.181
f-mean-pitch	0.770	0.768	**0.786**	0.782	0.766
f-range-pitch	0.373	0.434	**0.508**	0.420	0.441
f-time	0.053	0.150	0.189	0.270	**0.350**
f-maj-tonic	**0.679**	0.621	0.593	0.575	0.595
f-min-tonic	**0.650**	0.598	0.581	0.561	0.588
f-predominant	0.568	0.553	**0.576**	0.545	0.521
f-dominant-of-dominant	0.493	0.537	**0.594**	0.561	0.525
f-dominant	0.629	0.670	**0.737**	0.699	0.719
f-current-scale-diff	0.065	0.182	**0.262**	0.228	0.237
f-current-scale-relative	0.008	0.069	0.094	**0.146**	0.134

Table 2. Average value of the features according to the section on the whole corpus.

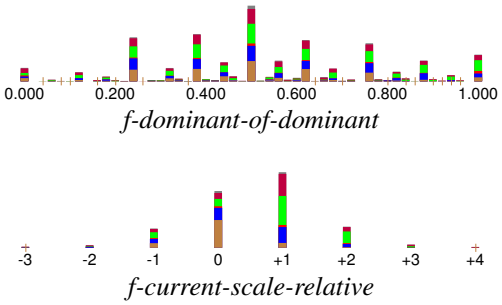


Figure 6. Distribution of two features relevant for MC identification. From bottom to top, P (brown), TR (blue), MC (red), S (green), and C (purple).

4.2 Implementation

We encoded feature extraction within the Python music21 framework [10]. The pitch space is Base40, modeling full pitch spelling information. Features described in Section 2 are computed at each beat of each piece and have their values scaled between 0 and 1 through min-max normalization. These features values are available as open data at `www.algomus.fr/data`. The neural network has been implemented with the Python framework Keras [8].

4.3 Features distribution

Table 2 shows the average values of each feature depending on the sections, and Figure 6 details the distribution of two relevant features. Several features have larger values on the MC, notably *f-dominant-of-dominant* and *f-dominant*. As expected, the features on tonic harmonies have higher values on P and TR, while features on dominant harmonies have higher values on the MC and S and C sections. The *current-scale* features are very low on P (initial tonal stability), but are then mostly activated on the other sections as the music moves to another key, and preferably one with more sharps. The *f-current-scale-diff* is maximal around the MC, reflecting the harmonic oscillations and tonal instability at this place. This behaviour is less visible on *f-current-scale-relative*. Indeed, we observe sometimes here a modal instability (as in Figure 2) that means more flats before the MC in major mode. Other relevant features

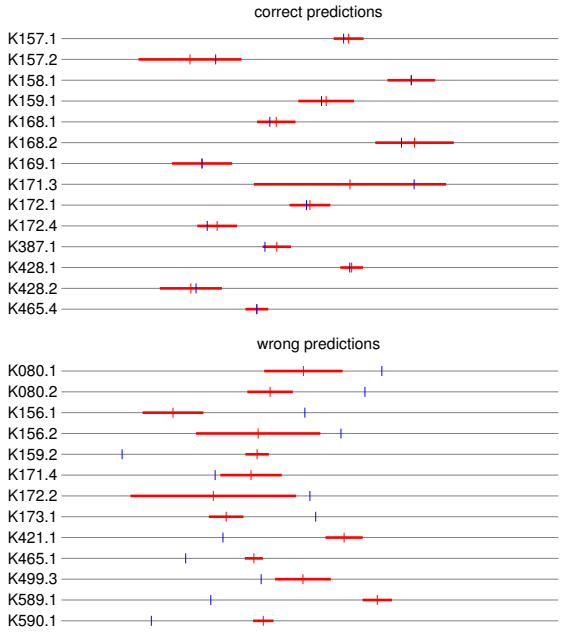


Figure 7. MC in the reference annotation (red marks on bold segments, representing ± 4 beats) and predicted by the model (blue marks).

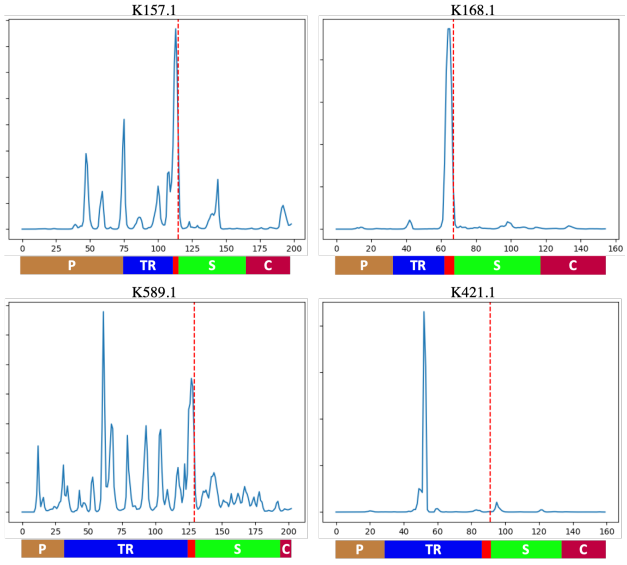


Figure 8. Probability curves for each beat to be an MC along four pieces of the corpus, two with correct MC predictions (top) and two with wrong MC predictions (bottom). Below each curve is the structure of the exposition in the reference annotation (P/TR/MC/S/C), and the red dashed lines emphasize the position of the MCs.

for the MC are, as expected, *f-hammer-blow*, and *f-range-pitch*, emphasizing the octave leap down often found on hammer blows.

4.4 MC prediction

In order to predict the position of the MC in an unseen piece, the sequence of vectors representing its beats are presented to the network. The position of the MC is identified at the offset where the network gives the maximum probability. We consider that a prediction is correct when the predicted MC is less than 4 beats before or after the annotated MC which seems reasonable given the progressive aspect of the MC phenomena.

Figure 7 displays the location at which the MC is identified as the MC original annotations for each piece of the corpus. The network correctly locates the MC of 14 of the 27 pieces of the corpus. This is a improvement from our previous work [1] where the model found only 8 MCs out of the same 27 pieces. Due to the small size of the corpus, we did not find any significant correlation between the accuracy of the prediction and the piece mode, tempo, or MC type. For example, MCs are correctly estimated in 11 out of 21 pieces in major mode and in 3 out of 6 pieces in minor mode.

Figure 8 displays the estimation of the probability of having a MC at each beat of four pieces of the corpus. These probabilities are computed by different models that have been trained on the whole corpus, except on the piece on which the prediction is performed. The model works well on some pieces. In K157.1, the highest peak predicts well the MC. The second highest peak, on beat 76, is also noteworthy as it is an HC ending the P section. Other peaks are not well explained. In K168.1, there is a unique peak at the correct position of the MC. The model fails to predict the MC in other pieces. In K421.1 (see also Figure 3), a MC is wrongly predicted with high confidence about 40 beats too early, at measure 14. This false prediction is actually a *denied MC*. In K589.1, there are more candidates for the MC location, but with low estimated probabilities, under 0.2. Another peak triggers the detection around beat 60, where there are two beats with only rests.

5. CONCLUSION AND PERSPECTIVES

We proposed theory driven features modeling structural breaks such as the Medial Caesura. Trained with only these features and without any other note information, the model succeeds in identifying about half of the MCs of the corpus. This is notable given the diversity of the realisations of MCs in such a small corpus. The success of the predictions does not seem to be correlated with the tempo, mode or the MC type.

The model might probably be improved both by enlarging the corpus and by taking into account additional elements that can not be retrieved from the files used in these experiments, such as dynamics. Features were selected based on music theory. It could be worth learning also more lower-level elements used in their computations, such as note pitches and durations. Furthermore, the performance of the model might be improved by considering additional musical features that have been proposed for other MIR tasks such as cadence detection or sonata form retrieval.

6. REFERENCES

[1] Pierre Allegraud, Louis Bigo, Laurent Feisthauer, Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. Learning sonata form structure on Mozart's string quartets. *Transactions of the International Society for Music Information Retrieval*, in revision.

[2] Adriano Baratè, Goffredo Haus, and Luca A Ludovico. Music analysis and modeling through Petri nets. In *International Symposium on Computer Music Modeling and Retrieval (CMMR 2005)*, pages 201–218, 2005.

[3] Louis Bigo, Laurent Feisthauer, Mathieu Giraud, and Florence Levé. Relevance of musical features for cadence detection. In *International Society for Music Information Retrieval Conference (ISMIR 2018)*, Paris, France, 2018.

[4] Louis Bigo, Mathieu Giraud, Richard Groult, Nicolas Guiomard-Kagan, and Florence Levé. Sketching sonata form structure in selected classical string quartets. In *International Society for Music Information Retrieval Conference (ISMIR 2017)*, pages 752–759, 2017.

[5] William E. Caplin. *Classical Form: A Theory of Formal Functions for the Instrumental Music of Haydn, Mozart, and Beethoven*. Oxford University Press, 1998.

[6] William E. Caplin. The classical cadence: Conceptions and misconceptions. *Journal of the American Musicological Society*, 57:51–117, 2004.

[7] William E. Caplin, James Hepokoski, and James Webster. *Musical Form, Forms & Formenlehre – Three Methodological Reflections*. Leuven University Press, 2009.

[8] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[9] Nathaniel Condit-Schultz, Yaolong Ju, and Ichiro Fujinaga. A flexible approach to automated harmonic analysis: Multiple annotations of chorales by bach and prætorius. In *International Society for Music Information Retrieval Conference (ISMIR 2018)*, pages 66–73, 2018.

[10] Michael Scott Cuthbert and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In *International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 637–642, 2010.

[11] Mathieu Giraud, Florence Levé, Florent Mercier, Marc Rigaudière, and Donatien Thorez. Modeling texture in symbolic data. In *International Society for Music Information Retrieval Conference (ISMIR 2014)*, pages 59–64, 2014.

[12] James Hepokoski and Warren Darcy. The medial caesura and its role in the eighteenth-century sonata exposition. *Music Theory Spectrum*, 19(2):115–154, 1997.

[13] James Hepokoski and Warren Darcy. *Elements of Sonata Theory: Norms, Types, and Deformations in the Late-Eighteenth-Century Sonata*. Oxford University Press, 2006.

[14] David Huron. Music information processing using the Humdrum toolkit: Concepts, examples, and lessons. *Computer Music Journal*, 26(2):11–26, 2002.

[15] Nanzhu Jiang and Meinard Müller. Automated methods for analyzing music recordings in sonata form. In *International Society for Music Information Retrieval Conference (ISMIR 2013)*, pages 595–600, 2013.

[16] Alec Hyatt King. *La Musique de chambre de Mozart*. Arles : Actes Sud, 1968.

[17] Carol L. Krumhansl and Edward J. Kessler. Tracing the dynamic changes in perceived tonal organisation in a spatial representation of musical keys. *Psychological Review*, 89(2):334–368, 1982.

[18] Cory McKay, Julie Cumming, and Ichiro Fujinaga. JSymbolic 2.2: Extracting features from symbolic music for use in musicological and MIR research. In *International Society for Music Information Retrieval Conference (ISMIR 2018)*, pages 348–354, 2018.

[19] David Meredith. *Computational Music Analysis*. Springer, 2015.

[20] Mark Richards. Sonata form and the problem of second-theme beginnings. *Music Analysis*, 32(1):3–45, 2013.

[21] Charles Rosen. *Sonata Forms*. W. W. Norton, 1980.

[22] David Sears, William E. Caplin, and Stephen McAdams. Perceiving the classical cadence. *Music Perception*, 31(5):397–417, 2014.

[23] David R. W. Sears, Marcus T. Pearce, William E. Caplin, and Stephen McAdams. Simulating melodic and harmonic expectations for tonal cadences using probabilistic models. *Journal of New Music Research*, 47(1):29–52, 2016.

[24] David Temperley. What's key for key ? the Krumhansl-Schmuckler key-finding algorithm reconsidered. *Music Perception*, 17(1):65–100, 1999.

[25] David Temperley and Daniel Sleator. Modeling meter and harmony: A preference-rule approach. *Computer Music Journal*, 23(1):10–27, 1999.

[26] Christof Weiß and Meinard Müller. Quantifying and visualizing tonal complexity. In *Conference on Interdisciplinary Musicology (CIM 2014)*, pages 184–188, 2014.

AUTO-ADAPTIVE RESONANCE EQUALIZATION USING DILATED RESIDUAL NETWORKS

Maarten Grachten
Contractor for Sony CSL Paris, France

Emmanuel Deruty
Sony CSL Paris, France

Alexandre Tanguy
Yascore, Paris, France

ABSTRACT

In music and audio production, attenuation of spectral resonances is an important step towards a technically correct result. In this paper we present a two-component system to automate the task of resonance equalization. The first component is a dynamic equalizer that automatically detects resonances, to be attenuated by a user-specified factor. The second component is a deep neural network that predicts the optimal attenuation factor based on the windowed audio. The network is trained and validated on empirical data gathered from a listening experiment. We test two distinct network architectures for the predictive model and find that an agnostic network architecture operating directly on the audio signal is on a par with a network architecture that relies on hand-designed features. Both architectures significantly improve a baseline approach to predicting human-preferred resonance attenuation factors.

1. INTRODUCTION AND RELATED WORK

Equalization is part of the audio mixing and mastering process. It is a redistribution of the energy of the signal in different frequency bands. The process has been traditionally performed by skilled sound engineers or musicians who determine the proper equalization given the characteristics of the input audio. Recently methods have been developed for semi-automatic and automatic equalization. These methods include automatic detection of frequency resonances [1], equalization derived from expert practices [7], and conformation to a target spectrum [15]. Equalization profiles may also be derived from semantic descriptors [5]. Appropriate equalization settings can be found through different means, for example by comparing the input source to previously equalized content [20], or by formulating equalization as an optimization problem where inter-track masking is used as the cost function [10]. Some automated equalization functionalities are featured in commercial products [1] [2].

[1] www.izotope.com/en/products/mix/neutron

[2] www.soundtheory.com/home

The use of machine learning, in particular neural networks, to solve audio production related tasks is recent. Automatic mixing tasks that have been addressed in this way include automatic reverbation [6], dynamic range compression [18], and demixing/remixing of tracks [17]. To our knowledge, there is no documented example of the use of neural networks for automatic equalization.

A specific form of equalization used both in mixing and mastering is the attenuation of resonating or salient frequencies, *i.e.* frequencies that are substantially louder than their neighbors [2]. Salient frequencies may originate from different phenomena, such as the acoustic resonances of a physical instrument or an acoustic space. They are considered a deficiency in the sense that they may mask the content of other frequency regions. One particular difficulty in resonance attenuation (RA) is finding the right amount of attenuation. For example, too much attenuation may unmask noise that would otherwise remain unheard, or flatten the spectrum to the point of garbling the original audio.

The subject of this paper is the automation of the RA process using machine learning. We limit our study to neural networks as the state of the art machine learning technique. Our method fully automates the RA process. It includes 1) a 0.5s windowed RA process that can be controlled with a single parameter—the *resonance attenuation factor* (RAF), 2) a deep neural network that predicts the attenuation factor from the input audio, making the process auto-adaptive [21].

For the training and validation we conduct a listening experiment determining optimal RAFs for a set of tracks, as chosen by sound engineers. We compare a neural network architecture that operates directly on the audio signal to a more traditional approach that includes a feature-extraction stage yielding a set of features commonly used in MIR. Results show that both approaches perform equally well, and significantly outperform a baseline.

The paper is organized as follows. Section 2 describes the RA process. The listening experiment is described in Section 3. The design, training, and evaluation of the predictive models is presented in Section 4, and conclusions are presented in Section 5.

2. RESONANCE EQUALIZATION

Traditionally RA has been a manual task where a sound engineer determines the resonating frequencies by ear or using a graphical tool, in order to reduce the energy of the

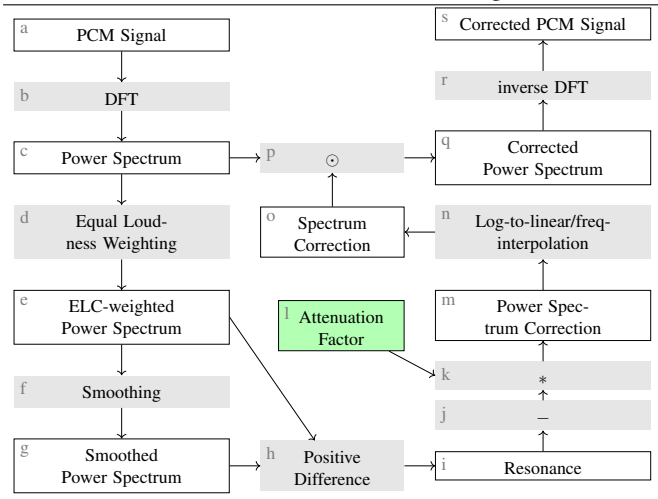


Figure 1. Resonance equalization block diagram; White and gray blocks represent data and processes respectively; The green block depicts the single user-controlled parameter; The symbols $\odot$, $*$ and $-$ represent elementwise vector/vector multiplication, elementwise scalar/vector multiplication, and unary negation respectively.

signal in those frequencies by an appropriate amount [16]. In this section, we describe a procedure that identifies resonating frequencies autonomously, and reduces the energy in those frequencies by a factor that is controlled by the user. The procedure works on overlapping audio windows that must be large enough to allow for spectral analysis at a high frequency resolution.

Figure 1 displays a block diagram of the RA process, where each element is denoted by a letter. We will use these letters to refer to the corresponding elements in the diagram. First the audio signal is used to compute a power spectrum weighted by Equal-Loudness Contours (ELC) [12] at a fixed monitoring level of 80 phon (Figure 1, element d) to reflect the perceptual salience of the signal energy at different frequency bands. The value of 80 phon is chosen in relation to the procedure detailed in Section 3.3. The ELC-weighted power spectrum (e) consists of 400 log-scaled frequency bands.

Resonances (i) are determined by smoothing ELC-weighted power spectrum [3] (e) to obtain (g) and computing the elementwise differences (e) minus (g), setting negative elements to zero (h). The negative of the resonances is then scaled by the user defined RAF (l), transformed back to a linear scale and converted back to the shape of the original spectrum using interpolation (h). The result (o) is a vector of scaling factors (one per DFT bin). Multiplying the original spectrum (c) with the scaling factors gives the corrected spectrum (q) which—by the inverse DFT (r)—yields the corrected audio signal (s).

3. LISTENING EXPERIMENT

A listening experiment was carried out to obtain ground truth in terms of optimal RAFs for a set of audio tracks. In the experimental design of the listening test it proved unpractical to ask subjects to set a varying RAF. Therefore we chose relatively homogeneous music fragments (excluding transitions between different sections of songs) and asked subjects for a single attenuation factor for the whole fragment, ensuring relatively homogeneous sound fragments.

3.1 Participants

A group of 15 subjects was recruited for the experiment, around Paris (France). All subjects are recognized professionals in the industry. Nine subjects specialize in studio recording (Classic/Jazz/Pop/Rock/Movie Music, audio post-production), three are experts in live music, and three are composer/music producers. The average age was 32 (min: 24, max: 42). The subjects were recruited and paid as if they were working on a commercial project.

3.2 Data

A set of 150 audio tracks was used for the listening experiment. The tracks are excerpts from longer pieces, with a mean duration of 46 seconds and a standard deviation of 16 seconds. All tracks were processed using Nugen AMB R128 [4] so that they were aligned to the same median loudness. The set comprised contemporary pop and rock music, as well as film scores. Of this set, 131 tracks were unique recordings, while the remaining 19 tracks were variants of some of the unique 131 recordings, with differences in mixing. None of the tracks were previously mastered.

3.3 Procedure

The listening experiment took place in a recording studio, where participants listened to the audio tracks individually, using studio monitors, at a measured loudness of 80 dBC– a typical listening loudness during audio production. The participants were presented with a web interface in which they could listen to each track with different degrees of RA, ranging from 0 (no attenuation) to 1 in 17 steps. They could select their preferred degree of RA, or alternatively decline to select any version, indicating that none of the versions sounded acceptable. The tracks were separated from each other by 10 seconds of pink noise surrounded by a short silence to give the participants a fixed reference. Sessions of 50 tracks were alternated with breaks.

3.4 Results and discussion

Basic statistics of the results per subject are given in Table 1. Subject 13 stands out because of the number of missing ratings (21 versus a median of 1 over all subjects). Subjects 1 and 15 have abnormally high rates of 0.0 ratings (72 and 58 respectively, versus a median of 16 over all subjects). Finally, Subject 7 stands out in terms of median rating (0.469 versus a median of 0.188 over all subjects).

To see how strongly the ratings are linearly related among subjects, we compute the Pearson correlation for

[3] using zero-phase low-pass filtering over the spectral bins

[4] nugenaudio.com/amb

Subject	# No rating	# 0.0	Min	Median	Max
s01	1	**72**	0.0	0.062	0.750
s02	0	2	0.0	0.188	0.812
s03	2	25	0.0	0.125	0.812
s04	2	7	0.0	0.250	1.000
s05	4	25	0.0	0.156	0.750
s06	0	22	0.0	0.125	0.875
s07	0	2	0.0	**0.469**	0.875
s08	8	9	0.0	0.312	1.000
s09	0	9	0.0	0.250	0.750
s10	1	17	0.0	0.188	0.812
s11	1	16	0.0	0.250	1.000
s12	2	25	0.0	0.188	1.000
s13	**21**	13	0.0	0.312	1.000
s14	0	3	0.0	0.250	0.875
s15	0	**58**	0.0	0.188	1.000

Table 1. Rating statistics per subject. Outlying values are highlighted in bold (see text).

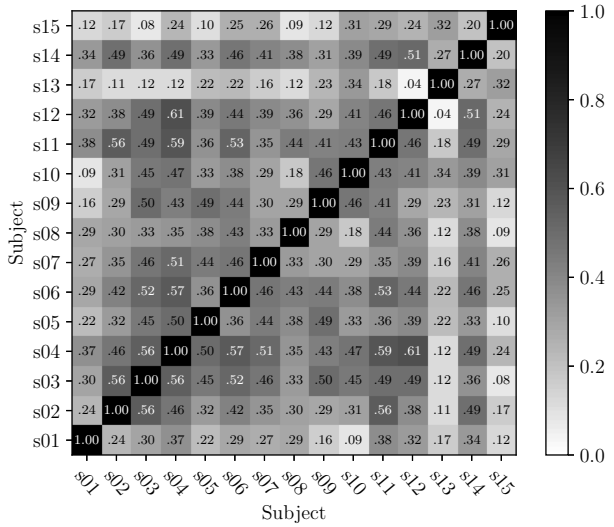


Figure 2. Subject rating correlation coefficients.

each pair of subjects (Figure 2). Apart from Subjects 13 and 15 (and to a lesser degree Subject 1) who appear to have different rating patterns from the majority of the subjects, the figure shows weak to moderate positive correlations between all subjects. This suggests that in spite of different preferred rating ranges, subjects made their judgments according to common criteria.

4. PREDICTING OPTIMAL ATTENUATION

In this section we describe the design and experimental validation of two modeling approaches to predict optimal attenuation factors from audio. The models are ultimately intended to be used in a real-time plugin for audio workstations. Although the real-time implementation is beyond the scope of this paper, it does guide some important design decisions for the modeling. Most importantly it implies a causal design in which the track cannot be analyzed as a whole in order to estimate the optimal attenuation factor. On the other hand, the audio latency upper-bound for real-time operation (maximum observed audio latency in real-time commercial plug-ins is 4096 samples) is too low for accurate prediction of the attenuation factor. This implies that the attenuation factor that will be applied at time t will be estimated from a windowed part of the signal (immediately) before t. Whether the predicted attenuation factor is still approximately valid for the signal at time t depends on the length of the window in relation to how quickly the resonance characteristics of the signal can change. Our aim is to target long range phenomena—3 seconds or longer, the time scale used for *short-term loudness* in EBU R128 [9]. From this perspective we consider a window size of 0.5s a good trade-off, offering sufficient data for an informed prediction and at the same time being short enough to adapt to changes in resonance characteristics at the 3s time-scale.

We describe and test two alternative neural network approaches to the problem of predicting optimal attenuation factors. The first is a more traditional approach in which a feature set is computed from an audio window, from which the attenuation factor is predicted. The second approach skips the intermediate feature representation. Instead, it takes the stereo PCM signal directly as input to a neural network that predicts the attenuation factor.

4.1 Feature-based prediction (FFN)

Performing regression or classification tasks on audio using feature descriptions of the audio has been the predominant approach for the past decades, and is based on the intuition that the prediction is determined by characteristics of the signal that can be defined explicitly and computed from the audio. These descriptors often capture spectral characteristics of the signal, but may also approximate perceptual characteristics, such as loudness. Many audio descriptors that have been proposed in the literature over time are implemented in a software library called Essentia [4]. The descriptors used in this study are listed in Table 2. All descriptors are available in the Essentia library, except *harmonics-to-noise ratio* [3] and *stereo width* (two descriptors, computed as the correlation between channels and absolute difference in RMS between channels, respectively), for which we used our own implementation.

The features are computed on shorter timescales (typically 1024 samples) than the 0.5s audio window for which our prediction will be made. Thus the feature computation stage returns a vector of values for each feature. We summarize each of these vectors by 7 statistics: the *mean*, *median*, *standard deviation*, *skew*, *kurtosis*, the 10^{th} *percentile*, and the 90^{th} *percentile*. This yields a total of 679 values per data instance, based on which a prediction must be made.

The network consists in a stack of linear layers (also called *dense*, or *fully connected*), each of which is followed by a *batch normalization* (BN) layer and a layer of *rectified linear* units (ReLU). The BN layer transforms the distribution of the output activations of the preceding linear layer to zero mean and unit variance by keeping track of mean and variance during the training of the model. The ReLU layer performs a non-linear transformation by setting negative output activations of the preceding layer to

MFCC (13 values)	zero-crossing rate
GFCC (13 values)	spectral flatness dB
inharmonicity	high frequency coefficient
pitch	barkbands (30 values)
pitch salience	pitch instantaneous confidence
spectral complexity	silence rate (at 20/30/60dB)
spectral crest	odd-to-even harmonic energy ratio
spectral decrease	spectral energy band (4 values)
spectral energy	tristimulus (3 values)
spectral flux	spectral contrast (6 values)
spectral rms	spectral valley (6 values)
spectral rolloff	stereo width (2 values)
spectral strong peak	harmonics-to-noise ratio

Table 2. List of audio descriptors used in the FFN.

zero. The number of linear layers and their sizes are not fixed in advance but determined using a hyper-parameter optimization scheme (Section 4.3). A final linear layer is added after the last ReLU layer. This layer has a single output—the predicted RAF.

4.2 Signal-based prediction (DRN)

In this section we describe a convolutional neural network that takes slices of a stereo PCM signal of the audio as input and provides an estimate of the optimal attenuation factor. Note that even for a window of moderate size and sample rate this quickly leads to tens of thousands of samples to be taken as model inputs. As opposed to a feature vector however, the inputs are ordered along a meaningful dimension (time), in which patterns can be identified, and are thus amenable to *convolution*. This approach, which was pioneered in [14], exploits the fact that such data display local patterns that may occur at different locations in the data. The strength of convolutional networks is that they learn to recognize patterns independently of their absolute location, and at the same time the convolution operation is much more space efficient than the "fully-connected" matrix dot product using in feed-forward neural networks, allowing for larger models. By stacking convolutional layers on top of each other it is possible to detect patterns of increasing size, and by interleaving the convolution operation with so-called *pooling* or *sub-sampling* operations, the patterns become somewhat invariant to local deformations.

However promising, the potential of traditional convolutional networks has been limited by a number of factors. Two limitations have been addressed by recent extensions of the traditional convolutional network approach, namely *dilated convolution*, and *residual networks*. We integrate both extensions in our convolutional network for predicting RA, and discuss each of them briefly before we describe the global architecture of the model.

4.2.1 Dilated convolution

An approach often used with traditional convolution in order to create high-level feature representations of data is pooling using *max* or *average* aggregation functions. For instance, max-pooling sub-samples the input by selecting maximal elements in a sliding window, typically using a sliding step (*stride*) equal to the size of the pooling window. Stacking convolution/pooling operations leads to features with increasing *receptive fields*, meaning that the features can describe patterns of increasing size. However, it comes at the cost of resolution loss: The relative position of features becomes less precise as their size increases.

On the contrary, dilated convolution achieves high-level features without loss of resolution. Rather than increasing stride, it increases the receptive field of the features by "dilating" the convolution kernels. A normal convolution of the kernel k with the signal s involves multiplying kernel elements with contiguous signal samples (τ is a discrete variable that increases in steps of 1):

$$(k * s)(t) = \sum_{\tau=-\infty}^{\infty} k(\tau)\, s(t-\tau) \tag{1}$$

In convolution with dilation factor $d \in \mathbb{Z}^+$ on the other hand the kernel elements are multiplied with signal samples that are equally spaced at d samples:

$$(k *_d s)(t) = \sum_{\tau=-\infty}^{\infty} k(\tau)\, s(t-d\tau) \tag{2}$$

By stacking convolutional layers with increasing dilation factors the higher level filter kernels aggregate information over input ranges of exponentially increasing size, even if the size the kernels (in terms of parameters) does not increase, and the resolution remains intact. This approach has proven successful in image processing tasks such as semantic segmentation [23].

4.2.2 Residual blocks

Another issue with convolutional networks is that as they grow deeper in order to capture higher level patterns, it becomes harder to optimize the lower level convolutional layers. This is directly related to the fact that for low level feature activations to influence the output of the model, they must pass through multiple layers of convolutions. Sometimes however, it is desirable for low level features to be able to directly influence the output of the model, not just to figure as a building block for higher level features.

This observation has led to the proposal of the *residual block* as a sub-structure used in deep networks [11, 24], an adaptation of which is depicted in Figure 3. In this structure the information flows from input to output through two pathways in parallel. The left pathway involves a typical convolution layer with configurable parameters: the kernel size k, number of kernels n, and dilation factor d. The right pathway convolution uses kernels of size 1 (the dilation factor is thus irrelevant), and thus does not compute any features from the input. Instead, it outputs n linear combinations of the input to ensure shape compatibility for elementwise addition to the n *feature maps* of the left pathway. The pathways further include batch normalization operations. After the elementwise sum of the pathways a rectified linear unit allows for a non-linear response.

The term "residual" refers to the fact that the left pathway only needs to account for variance in the output that

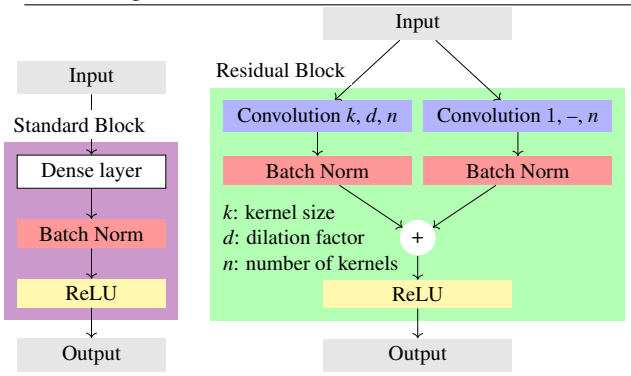


Figure 3. Building blocks for the FFN and DRN models. Left: Standard block (See Section 4.1); Right: Residual block (See Section 4.2.2).

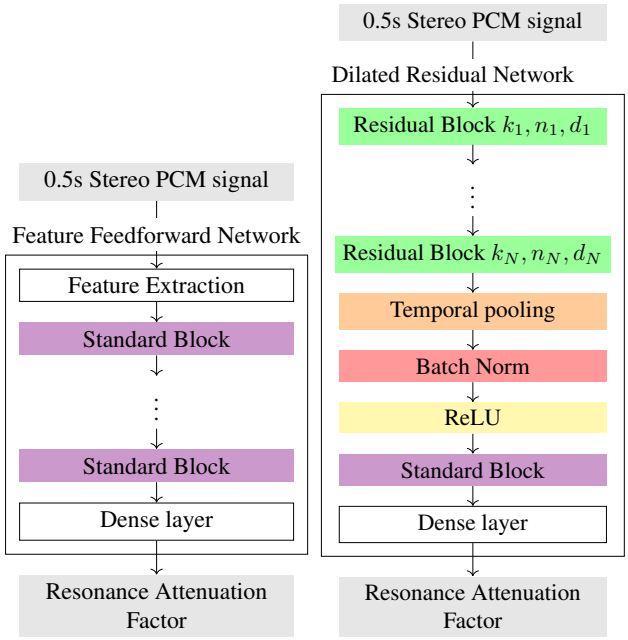


Figure 4. FFN and DRN architectures.

cannot be modeled as linear combinations of the input—the right pathway. Thus, increasing the number of layers does not hamper the ability of the network to account for its output in terms of lower level features.

4.2.3 DRN architecture

Figure 4 shows the complete model consisting of multiple residual blocks. Note that the residual blocks maintain the original temporal dimension of the data, which amounts to a size of 11025 for 0.5s of audio sampled at 22050Hz. The temporal pooling operation reduces this number by down-sampling the output using window-wise averaging, and is followed by two dense layers with intermediate batch-normalization and non-linearity in order to produce an estimated RAF.

4.3 Experiments

In this section we describe the training and evaluation procedure of both model architectures described above. We use the human ratings of the 150 tracks gathered in the listening experiment to train and evaluate both architectures.

4.3.1 Procedure

Evaluation Criterion Predicting optimal RAFs for given tracks is a regression problem, suggesting the mean squared error of the predictions with respect to the optimal value (the *target*) as an objective to be minimized. However, given the variance in the ratings across subjects in the ground truth, it may be hard to determine a unique optimal value per track. Using the mean or median of the ratings per track as a target has the drawback that the mean squared error objective does not differentiate between tracks with different degrees of rater consensus. Ideally, we wish to impose a lower penalty on errors from the mean rating when the rater consensus is low. We do so by generalizing the mean squared error objective as follows. Rather than defining the objective function to be minimal only for a single value, we define it to be minimal whenever the prediction lies within a specified interval that varies from one data instance to another. For a given data instance consisting of an audio track $A \in \mathbf{A}$ and a set of ratings $\mathbf{F}$ we define the zero penalty interval as $[\, P_l(\mathbf{F}),\ P_h(\mathbf{F})\,]$, where $P_l(\mathbf{F})$ and $P_h(\mathbf{F})$ denote the l-th and h-th percentiles of the ratings $\mathbf{F}$, respectively, with $l \leq h$. We refer to this objective (which is novel to the best of our knowledge) as the *mean squared bounds error* with bounds l, h, or $MSBE^{(l,h)}$. We use $l = 35$ and $h = 65$ throughout the experiments.

Formally, given a dataset D consisting of pairs $(A, \mathbf{F})$, the $MSBE^{(l,h)}$ of a model $f : \mathbf{A} \to \mathbb{R}$ is defined as:

$$MSBE^{(l,h)}(f, D) = \frac{1}{|D|} \sum_{(A,\,\mathbf{F})\,\in\,D} L^{(l,h)}_{A,\mathbf{F}}(f), \qquad (3)$$

where

$$L^{(l,h)}_{A,\mathbf{F}}(f) = \Big([\, f(A) - P_h(\mathbf{F})\,]^+ + [\, f(A) - P_l(\mathbf{F})\,]^-\Big)^2. \qquad (4)$$

The brackets $[\,\cdot\,]^+$ and $[\,\cdot\,]^-$ denote the positive and negative parts respectively.

Hyper-parameter optimization We use Bayesian optimization to find the optimal hyper-parameters for each of the models, most importantly the depth of the networks and the hidden layer sizes. This is a heuristic to speed up the search for appropriate hyper-parameter values compared to an exhaustive grid search. The particular form of optimization we use is based on a gaussian process approximation of the loss as a function of the hyper-parameters. This approximation gives rise to the *upper confidence bound* [13], which estimates the expected loss for hyper-parameter settings that have not yet been tested, and is used as a guide to search the space of hyper-parameters [22].

Apart from the depth of the models and the hidden layer sizes, the optimization involved hyper-parameters to control the training procedure: the *learning rate*, and the thresholds for *early stopping*, and *learning rate reduction*.

	FFN	DRN
Depth	3 Std. Blocks	10 Res. Blocks
Blk. Size (Low/Mid/High)	500 / 250 / 250	100 / 100 / 300
Temporal Pooling	–	300
Final Std. Block Size	–	10

Table 3. Optimal configuration for the FFN and the DRN architectures as found by hyper-parameter optimization.

			95% CI	
Model	Mean	Std. dev.	Low	High
Baseline	0.237	0.103	0.194	0.280
FFN	0.159	0.082	0.124	0.194
DRN	0.154	0.080	0.121	0.188

Table 4. Means, standard deviations, and the 95% confidence interval (CI) for the mean $MSBE^{(35,\,65)}$ per model.

Cross-validation To perform the hyper-parameter optimization we use two partitions of the dataset into a test set (10 tracks), a validation set (10 tracks), and a train set (130 tracks). For each of the test tracks we compute the $MSBE^{(35,\,65)}$ loss on 100 randomly selected 0.5s frames. The criterion used to optimize the hyper-parameters is the average frame-wise loss across both test sets.

With the best hyper-parameters found for the FFN and DRN architectures, respectively, we perform a further five fold cross-validation. To this end, we use the same dataset, but exclude the 20 test tracks used for hyper-parameter optimization. We repeat the five fold cross validation five times using different random seeds to reduce the effect of partitioning of the data into folds and model parameter initializations on the result.

Baseline We define a baseline approach as a reference for evaluating the FFN and DRN architectures. It consists in computing the mean RAF over all tracks in the training set, and using this value as a prediction for the test set, irrespective of the input audio.

4.3.2 Results and discussion

The optimal configuration for FFN and DRN architectures, as found by hyper-parameter optimization, are shown in Table 3. Figure 5 shows the results of these architectures on repeated five fold cross validations. A one-way repeated measures ANOVA reveals a significant effect of model on $MSBE^{(35,\,65)}$ ($F_{2,\,72} = 6.55$, $p = 0.002$). A post-hoc Tukey HSD test at $\alpha = 0.05$ indicates that DRN and FFN differ significantly from the baseline. The effect size of DRN over baseline corresponds to Cohen's $d = 0.88$, whereas the FFN over baseline effect size is $d = 0.82$. The difference between DRN and FFN is not significant.

Table 3 shows that the FFN architecture works best when it is comparatively shallow (three standard blocks, the minimal depth tested), whereas the DRN architecture performs better when it is deep (10 residual blocks, the maximal depth tested). This trend is consistent in a review of the 10 best FFN and DRN architectures as found in the hyper-parameter optimization, omitted here for the sake of brevity. The layer sizes however do not show a similarly consistent trend, and vary considerably throughout the 10 best FFN and DRN architectures.

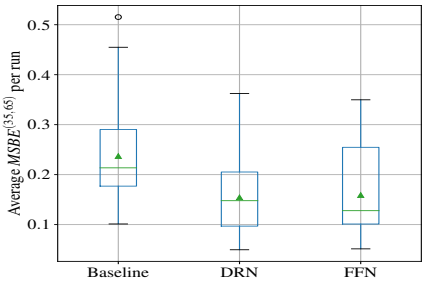


Figure 5. Boxplot of $MSBE^{(35,\,65)}$ cross-validation results for baseline, FFN, and DRN models; Horizontal lines in the boxes denote the median, triangles the mean values.

The fact that both approaches show similar prediction accuracies is in line with a general trend in the deep learning literature that computational tasks can be solved without the need for hand-designed features.

At the same time, the roughly equivalent performance of the FFN and DRN measured here is at odds with a multitude of cases where end-to-end deep networks clearly outperform prior state-of-the-art methods that rely on a hand-designed feature extraction stage, especially in image processing [25]. For audio tasks such as automatic tagging however, this does not seem to be the case however [8]—where end-to-end approaches require large training data sets in order to outperform spectrogram-based approaches [19]. Given the small data set used here—especially in combination with inter-subject variance and the non-uniform distribution of the ratings—this may explain why the DRN does not outperform the FFN.

5. CONCLUSION

We have proposed a method to attenuate automatically identified resonances by a user-controlled factor, and gathered ground-truth data for the optimal attenuation factor from sound engineers. The data—revealing general consensus in ratings among subjects—were used to train and evaluate two types of neural networks to estimate optimal resonance attenuation factors. The results show a domain-agnostic dilated residual network operating directly on the audio signal performs on a par with a neural network operating on a set of commonly used audio features. Although this does not discredit the feature-based approach to the resonance attenation problem per se, it does show that further improvements—if possible—will require specially crafted features based on expert domain knowledge.

The proposed system is a fully auto-adaptive resonance equalization system in which the attenuation factor is chosen automatically by a deep neural network. To our knowledge, this system is the first documented self-adaptive equalizer based on neural networks. Future work includes a real-time implementation of the presented model as a real-time plugin that can be used in audio work stations.

6. REFERENCES

[1] Joerg Bitzer and Jay LeBoeuf. Automatic detection of salient frequencies. In *Audio Engineering Society Convention 126*. Audio Engineering Society, 2009.

[2] Joerg Bitzer, Jay LeBoeuf, and Uwe Simmer. Evaluating perception of salient frequencies: Do mixing engineers hear the same thing? In *Audio Engineering Society Convention 124*. Audio Engineering Society, 2008.

[3] Paul Boersma. Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound. In *IFA 17*, pages 97–110, 1993.

[4] Dmitry Bogdanov, Nicolas Wack, Emilia Gomez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, Jose R. Zapata, and Xavier Serra. Essentia: an audio analysis library for music information retrieval. In *14th International Society for Music Information Retrieval Conference*, November 4-8 2013.

[5] Mark Brozier Cartwright and Bryan Pardo. Social-eq: Crowdsourcing an equalization descriptor map. In *ISMIR*, pages 395–400, 2013.

[6] Emmanouil T. Chourdakis and Joshua D. Reiss. A machine-learning approach to application of intelligent artificial reverberation. *J. Audio Eng. Soc.*, 65(1/2):56–65, 2017.

[7] Brecht De Man and Joshua D Reiss. A knowledge-engineered autonomous mixing system. In *Audio Engineering Society Convention 135*. Audio Engineering Society, 2013.

[8] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968. IEEE, 2014.

[9] EBU-R-128. BU Tech 3341-2011, Practical Guidelines for Production and Implementation in Accordance with EBU-R-128, 2011.

[10] Sina Hafezi and Joshua D Reiss. Autonomous multitrack equalization based on masking reduction. *Journal of the Audio Eng. Soc.*, 63(5):312–323, 2015.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR*, pages 770–778, 2016.

[12] International Standardization Organization. Iso 226:2003: Acoustics–normal equal-loudness-level contours. Geneva, Switzerland.

[13] T.L Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.*, 6(1):4–22, March 1985.

[14] Yann Lecun, LÃl'on Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *IEEE*, volume 6, pages 2278–2324, 1998.

[15] Zheng Ma, Joshua D Reiss, and Dawn AA Black. Implementation of an intelligent equalization tool using yule-walker for music mixing and mastering. In *Audio Engineering Society Convention 134*. Audio Engineering Society, 2013.

[16] Greg McCandless and Daniel McIntyre. *The Craft of Contemporary Commercial Music*. Routledge, 2017.

[17] Stylianos Ioannis Mimilakis, Estefanía Cano, Jakob Abeßer, and Gerald Schuller. New sonorities for jazz recordings: Separation and mixing using deep neural networks. In *2nd AES Workshop on Intelligent Music Production*, 2016.

[18] Stylianos Ioannis Mimilakis, Konstantinos Drossos, Tuomas Virtanen, and Gerald Schuller. Deep neural networks for dynamic range compression in mastering applications. In *Audio Engineering Society Convention 140*, May 2016.

[19] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik M. Schmidt, Andreas F. Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. In *Workshop on Machine Learning for Audio Signal Processing (NIPS)*, 2017.

[20] Dale Reed. A perceptual assistant to do sound equalization. In *5th International Conference on Intelligent User Interfaces*, pages 212–218, New York, NY, USA, 2000. ACM.

[21] Joshua Reiss and Øyvind Brandtsegg. Applications of cross-adaptive audio effects: automatic mixing, live performance and everything in between. *Frontiers in Digital Humanities*, 5:17, 2018.

[22] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[23] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2015.

[24] Fisher Yu, Vladlen Koltun, and Thomas A. Funkhouser. Dilated residual networks. *CoRR*, abs/1705.09914, 2017.

[25] Liang Zheng, Yi Yang, and Qi Tian. Sift meets cnn: A decade survey of instance retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 40(5):1224–1244, 2018.

Session D

ANALYZING USER INTERACTIONS WITH MUSIC INFORMATION RETRIEVAL SYSTEM: AN EYE-TRACKING APPROACH

Xiao Hu[1] **Ying Que**[1] **Noriko Kando**[2] **Wenwei Lian**[1]
[1]University of Hong Kong
{xiaoxhu, queying, lian0624}@hku.hk
[2]National Institute of Informatics, Japan
kando@nii.ac.jp

ABSTRACT

There has been little research considering eye movement as a measure when assessing user interactions with music information retrieval (MIR) systems, whereas many studies have adopted conventional user-centered measures such as user effectiveness and user perception. To bridge this research gap, this study investigates users' eye movement patterns and measures with two music retrieval tasks and two interface presentation modes. A user experiment was conducted with 16 participants whose eye movement and mouse click behaviors were recorded through professional eye trackers. Through analyzing visual patterns of eye gazes and movements as well as various metrics in prominent Areas of Interest (AOI), it is found that users' eye movement behaviors were related to task type. Besides, the results also disclosed that some eye movement metrics were related to both user effectiveness and user perception, and influenced by user characteristics. It is also found that some eye movement and user effectiveness metrics can be used to predict user perception. This study allows researchers to gain a deeper insight into user interactions with MIR systems from the perspective of eye movement measure.

1. INTRODUCTION

With the rapid development of Music Information Retrieval (MIR) as a field, user has been increasingly recognized as playing an essential or central role in the process of music retrieval [14][25]. Users' interactions with MIR systems have been drawing researchers' attention for the purposes of evaluating MIR techniques [13] and interfaces [5] from users' perspectives, as well as improving understanding of users [19].

Various approaches have been utilized to collect data of user interactions with MIR systems including listening histories [26], click streams in system logs [11], user surveys or diaries [32], etc. A series of user-centered metrics have then been proposed and used to measure users' interactions including those in user effectiveness (e.g., number of songs listened to) [11] and user perceptions (e.g., emotion, satisfaction) [32]. To facilitate this process, various tracking techniques have been employed through off-the-shelf software tools and/or self-development apps [11][32].

As an alternative approach to measuring users' interactions with various stimuli (e.g., computer systems, web interface, learning materials), eye movement has been used to explore the relationship between human's cognitive process and their behaviors [3]. It is believed that people's cognition can be reflected by their eye movement patterns [27][30]. Nowadays, fast-developing high-tech devices enable researchers to acquire eye tracking data in an accurate and reliable manner, such as eye tracking glasses and eye movement sensors. Compared to measures that rely on users' self-report, eye tracking data are objective and thus could help avoid possible response bias [4]. With current (wearable) technology, the process of collecting eye movement data is becoming less obtrusive. If relevant methodologies and techniques are appropriately applied to MIR studies, measurements of eye movement could be potentially helpful for analyzing and evaluating user interactions with MIR systems.

This study aims to explore the application of eye movement measure to investigating user interactions with a MIR system, through a user experiment involving different MIR tasks and interface modes. The relationships between eye movement measure and traditional measures such as user effectiveness and user perception are analyzed. Possible influence of user characteristics (e.g., gender) is also taken into account. The findings are expected to provide empirical evidence on exploiting eye movement data in studying user interactions in MIR.

2. RELATED WORK

2.1 User Interactions in MIR

Users' interactions with MIR systems have been mostly studied in the context of user-centered evaluation of MIR. Compared with the system-centered evaluation paradigm, user-centered evaluation is not as popular, but is increasingly adopted, with the recognition of users being the center of MIR [25]. It is advocated that user interactions provide a more direct way to understand how human perceive and use MIR systems, as serving the users is arguably the ultimate goal of MIR systems [14]. Another purpose of studying user interactions in MIR is for designing interactive systems where users' behavioral, psychological and

physiological measures could serve as input as well as feedback to systems [15][11][32].

To date, a number of user-centered measures and metrics have been applied in studying user interactions in MIR. Conventional user-centered metrics including those of user effectiveness and user perception [16] have been adopted in MIR, such as number of songs found, task completion time and user satisfaction, etc. Besides, enjoyment has been recognized as an important aspect of user perception unique in MIR [18]. Along with the line of research considering the casual-leisure nature of music retrieval, new metrics were proposed for evaluating MIR, including novelty, aesthetics, and content quality [12].

To collect and analyze the aforementioned measures and metrics, a range of methods have been employed. Traditional methods of user studies such as survey, interview and observations are included [18] and further developed to work with current technologies such as experience sampling with mobile apps [32]. Interactive behaviors are often recorded as system logs with the support of software tools that can track mouse clicks and keyword input [11]. More recently, with the development of wearable technology, studies started to collect users' physiological signals while they interact with MIR systems [22]. Nevertheless, to date there has been little research investigating user-MIR interactions with eye movement measure [29] which has been recognized as effective in capturing people's cognitive process and used in a number of domains.

2.2 Eye Movement Tracking and Cognitive Behaviors

It is recognized that eye movement can reflect human cognitive behaviors. In recent years, eye movement has been used in the domain of text retrieval, particularly on the influence of search interface design on users' search behaviors [6], as well as the relationships between document relevance and users' cognitive efforts [8]. Eye movement has also been widely employed in studies on people's reading behaviors such as identifying the sections focused on by readers during reading process [24]. In the education domain, there are many studies benefited from analyzing learners' eye movement such as computer programming [23], language learning [1], and music education [20].

In leveraging eye movement data, an important step is to find concrete measurements that can describe eye movement in an accurate and reliable manner. Fixation and saccade are two primary measures of eye movement, which are regarded as plausible evidences to detect cognitive processing during interactive tasks such as information retrieval [4]. Fixation indicates that a person stares at certain location and lasts for a period of time, presumably to process certain cognitive tasks, while saccade is a dynamic visit between two fixations [19]. Furthermore, investigators have often attempted to categorize eye movement behaviors into different patterns so as to interpret possible cognitive implications by comparing the patterns [4][8][24][30]. Relatedly, visualization of eye movement patterns such as heatmaps of eye gazes and plots of eye scan paths is also a viable and frequently adopted approach to making sense of eye movement data [4]. For instance, based on the positions of intensive fixations and the speed of saccades, authors of [30] found the correlation between eye fixation and participants' learning performance.

MIR also involves cognitively intensive activities where eye movement has great potential in studying user-system interactions. An existing research gap is that very few user studies have considered eye movement as a measurement of interactions between user and MIR systems, even though there are studies exploiting eye movement in music psychology [17] and music performance [31]. Therefore, this study aims to bridge the gap by providing empirical evidence on studying interactive MIR systems from an eye-tracking approach.

3. RESEARCH QUESTIONS

Aiming to showcase how eye movement measures can be used in MIR research, this study considers multiple constructs in typical MIR research settings: retrieval tasks, system interface modes, user-centered measures as well as user characteristics. Specifically, through conducting a user experiment involving these factors, this study focuses on three research questions as follows:

Q1. To what extent do different music retrieval tasks and system interfaces have effect on users' eye movement measure?

Q2. To what extent is eye movement measure related to user effectiveness and user perception measures?

Q3. To what extent is eye movement related to user characteristics (e.g., gender, music listening habit)?

To answer these research questions, a set of hypotheses are formulated. There are two hypotheses under Q1:

H1: eye movement measure differs in different task types.

H2: eye movement measure differs in different interface modes.

Under Q2, there are three hypotheses as well:

H3: eye movement measure is related to user effectiveness measures.

H4: eye movement measure is related to user perception measures.

H5: eye movement and user effectiveness measures can predict user perception.

Under Q3, one hypothesis is formulated:

H6: eye movement measure is related to user characteristics.

Answers to Q1 will have methodological implications on the extent to which eye movement can be used in MIR task/system design and evaluation. Answers to Q2 can reveal how eye movement measures are related to commonly used measures of user behaviors and whether they can be complimentary to one another in the context of MIR. In particular, the answer to whether or not user perception, as

a subjective measure, is predictable by other objective measures can have the potential of facilitating MIR researchers and system designers to better understand users' perceptions through objective measures. User characteristics (e.g., background in music training) have been recognized as influential in MIR process[19][25], and Q3 is to investigate their relationship with eye movement measure.

4. METHORDS

To fulfill the goals of this study, a user experiment was conducted with a mood-aware MIR system. During the experiment, users' eye movement and other user-centered measures were collected. This section describes details of the system, tasks, experiment procedure and data analysis.

4.1 The System

Moodydb is an online MIR system that supports searching and browsing music by mood [13]. Based on spectrum features extracted from the music audio, this system classifies music into five mood categories, namely *passionate*, *cheerful*, *bittersweet*, *silly/quirky* and *aggressive* [10]. There were 750 songs in the system when the study was conducted, with 738 Western popular pieces (from the U.S. and the U.K.) and 12 Chinese popular songs. The pieces were unevenly distributed across the mood categories with *bittersweet* being the largest class (226 songs) and *silly/quirky* being the smallest (35 songs). The rest categories contained 141 – 184 songs. When a user input singer's name or song's title in query box, it will prompt a set of songs that match the textual query. After user chooses one of the songs as a seed song, the system will retrieve a group of songs with similar mood to the seed song and display them as recommended songs to the user.

The system interface is presented in two different ways; one is in a traditional list-based layout and the other is visualization of album covers based on nested figures as in the treemapping display method [28]. As shown in Figure 1, the List layout ranks the recommended songs from the top of the screen down according to how similar the moods of the recommended songs are to that of the seed song. The Visual layout shown in Figure 2 uses the size of the album covers to represent the degree of similarity: the larger size an album image is in, the more similar the song is to the seed song in term of music mood.

4.2 Tasks and Topics

There were two music information retrieval tasks designed for this study: searching and browsing. As for searching, the participant was given a seed song (e.g., Irreplaceable by *Beyoncé*) to start with, and then make use of the system as a search engine to find other songs whose moods are similar to that of seed song. As for browsing, the participant was required to browse music in a given (seed) mood (e.g., *bittersweet*) and find songs in the mood. For each task, there were two topics that had different seed songs (for the searching task) or different seed moods (for the browsing task), making a total of four topics. All participants conducted both tasks (searching and browsing) with both interface modes (List and Visual), making a 2 * 2 experiment design. All participants conducted the same topics (same seed songs /moods) but the task and interface mode combinations were ordered in a Graco-Latin square arrangement to counterbalance possible sequence effect.

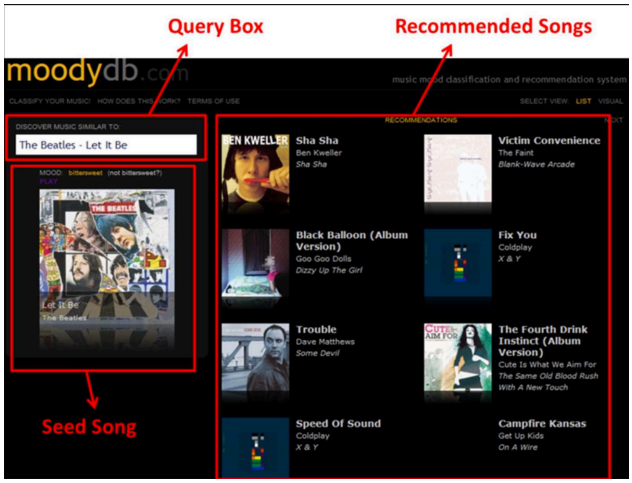


Figure 1. List-based interface mode in the system

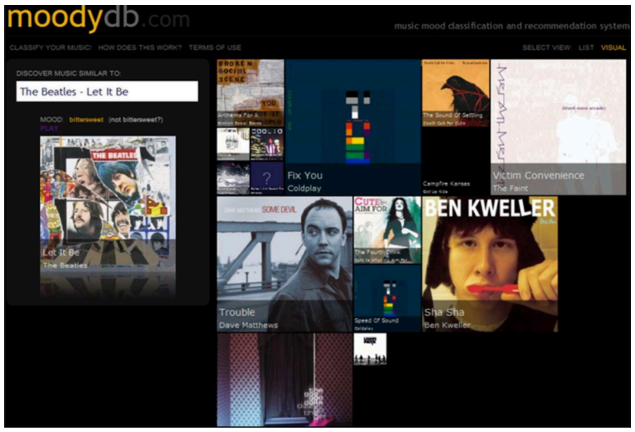


Figure 2. Visual-based interface mode in the system

4.3 Procedure

The experiment started with a pre-session questionnaire about demographic information, music knowledge, and general music information behaviors of the participants. After that, a research assistant demonstrated how the system could be used and the participants used the system for several minutes to familiarize with the system. An eye tracking sensor, Tobii T60, was used to collect data of participants' eye movements. Calibration was done with the eye tracker before a participant started working on each topic, with the assistance of a researcher. After completing all four topics (which covers both tasks and interface modes), participants were inquired about the feelings of the experiment through a post questionnaire. The experiment lasted for approximately one hour, and each participant was paid a nominal remuneration upon completing the experiment.

4.4 Measures

The following user-centered measures are used to answer the three research questions raised in this study.

User effectiveness was used to assess user performance in the experiment. It included four metrics: (a) Number of songs found to fulfill each topic; (b) Number of songs played during the process of working on each topic; (c) Completion time of each topic (measured in minute); (d) Number of mouse clicks during the process of working on each topic. The first metric came from answers submitted by the participants while others were recorded by Tobii 60.

User perception was used to understand users' feelings after they completed each topic. It included three metrics: (a) Task easiness; (b) Preference on the given seed song; (c) Satisfaction with songs found. Measured by a 7-point Likert scale, the perception degree ranges from the most negative to the most positive. Take task easiness as an example: point 1 means the user felt the topic not easy at all, while point 7 means he/she felt the topic very easy.

Eye movement was described statistically with five metrics: (a) Fixation Duration; (b) Fixation Count; (c) Visit Duration; (d) Total Visits Duration; (e) Visit Count. Metrics related to duration such as fixation duration and visit duration is measured by second. It is noteworthy that fixation and visit are two different concepts; the former indicates that participants' eyes fixate on one point, while the latter measures a process of saccade (i.e. moving the eye gaze from one point to another).

The eye movement metrics are calculated in defined Areas of Interest (AOI). In this study, the interface of system is divided into four AOIs based on their functions in the music retrieval process. They are (a) Search Box; (b) Seed Song; (c) Recommended Songs; (d) Player, which are shown in Figure 3. It is noteworthy that a Visit Duration in an AOI is defined as the interval of time between the first fixation inside the AOI and the first subsequent fixation outside the AOI. Besides, while the Visit Duration measures the duration of each individual visit within an AOI, Total Visit Duration measures the sum of duration of all the visits within an AOI. In addition to the metrics in each AOI, we also take the total across all AOIs into account. Statistics of the eye movement metrics in each AOI are generated by the Tobii Studio Analyzer.

4.5 Data Analysis

Hypotheses proposed under the research questions are tested using corresponding statistical tests. H1 and H2 compare metrics between two tasks and two interface modes respectively, for which pair wised t-tests are applied. H3 and H4 investigate relationships between metrics, and thus correlation analysis is used, including Pearson's correlation (for numerical metrics) and Spearman's correlation (for ordinal metrics). Linear regression is used to test H5 which concerns the predictive power of the metrics to user perception. Last but not least, H6 is tested through point-biserial correlation analysis (for binary metrics) and Spearman's correlation analysis. For cases where multiple tests are conducted, Bonferroni corrections are conducted to control type I errors [9].

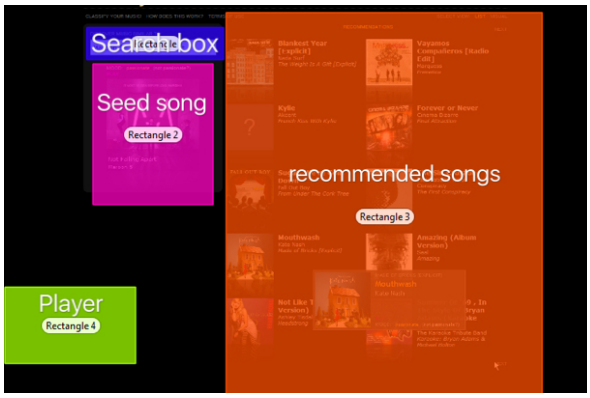


Figure 3. Four AOIs in this study

Besides, visualization as a data analysis method was used in this study as well. There are a number of visualization types in eye movement analysis, such as gaze plot and heat map. In this study, we used heat map to compare eye movement in two interface modes. Examples of heat map are shown in Figures 4 and 5 where aggregated time of fixations on the screen are illustrated by different colors. Red represents the most fixations or longest time period, while green is the least and shortest. The varying colors infer that the levels are in between.

5. RESULTS AND DISCUSSIONS

5.1 Participants

A combination of purposive and convenient sample methods was recruited in the user experiment, including 16 Japanese undergraduate and graduate students (8 female). Their ages ranged from 19 to 50, with a mean of 22.9 and a median of 21. The standard deviation of the age was 7.37. They were from a range of majors including Science, Medicine, Economics, Law and Humanities. A majority of them (11) were able to play an instrument while 5 of them could not. About half of them (7) searched for music fairly frequently, at least several times a week. While the rest of them searched music at least one a month. In term of how often they listened to music, 6 of them rated weekly, 7 daily, and 3 multiple times a day.

5.2 Influences of Task Types or Interface Modes on Eye Movement

This section aims to answer Q1 and test H1and H2. Table 1 presents means and standard deviation (in parenthesis) of eye movement metrics with significant difference in corresponding AOIs between the browsing and searching tasks, detected by paired t-test after Bonferroni correction [7]. As Table 1 shows, during the searching task, participants had longer fixation duration and more fixation count in areas of seed song as well as more visit count in total AOIs. Therefore, our hypothesis H1 is partially supported.

It is not surprising that there were more eye movement in the seed song area during searching task as users needed to compare the seed song and retrieval results, while there was no need for such comparison during the browsing task. The difference on visit count in total AOIs could possibly be attributable to the fact that comparing seed songs and retrieved songs would need a user to move eye gazes between the two AOIs which in turn generated more visits.

Measure	Browsing	Searching	*p* value
FixDur in SeedSong	16.22 (10.91)	27.66 (15.33)	.001*
FixCnt in Seedsong	69.56 (39.37)	109.72 (59.86)	.001*
VisCnt in Total	114.22 (48.98)	152.69 (69.50)	.000*

Table 1. Eye movement measures with significant differences between browsing tasks and searching tasks

The paired t-test on interface mode did not generate significant results after Bonferroni correction, and thus H2 is not supported. This can be explained by the fact (and limitation) that the difference between the two interface modes is actually within an AOI, the recommended song AOI (Figure 3). This calls for alternative methods to compare the two interface modes.

To compare eye movement in a qualitative manner, we present the heat maps of the two interface modes in Figures 4 and 5 respectively. As is shown in Figure 4, fixations from the list-based interface mode disperse along with the positions of the recommended songs. In contrast, in visual-based interface mode (Figure 5), fixations of participants nearly concentrate on the center of recommended songs area. In addition, although, by convention, the List mode ranks highly relevant results up in the list, Figures 4 shows that participants did not only focus on the top rows. For the Visual mode, while more relevant results (with bigger album images) could appear anywhere in the recommended song area, participants' attention seem to have focused on the middle only (Figure 5). These observations are somewhat anti-intuitive and worthy of future investigation.

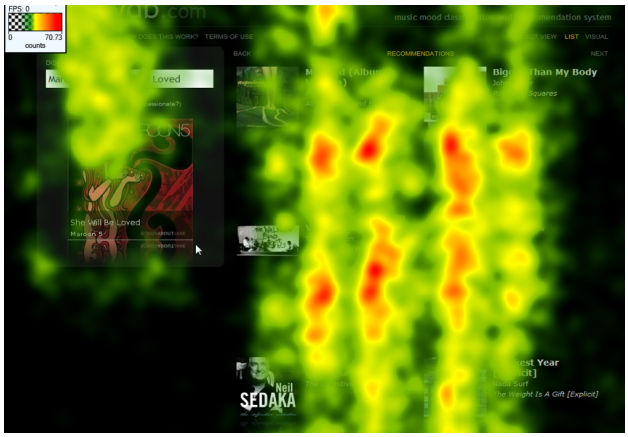


Figure 4. Eye fixation heat map of the List interface mode

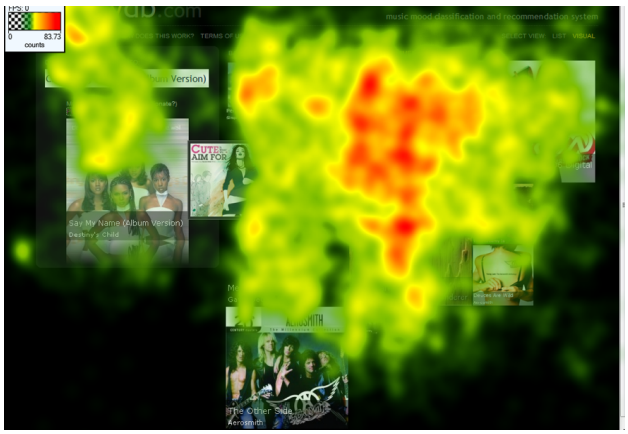


Figure 5. Eye fixation heat map of the Visual interface mode

5.3 Relationships among Eye Movement, User Effectiveness and User Perception Measures

This section aims to answer Q2 and tests H3 to H5. To test H3, Pearson's correlation coefficients were calculated for user effectiveness metrics on interval scales: number of songs found in each topic, number of songs played in each topic and completion time of each task. Significant results after Bonferroni corrections are shown in Table 2.

Effectiveness	Eye Movement	Coefficient	*p* value
Completion Time	TotVisDur in RecomSong	.416	.001*
	TotVisDur in Total	.476	.000*

Table 2. Correlation between eye movement and user effectiveness (Pearson)

As we can see in Table 2, completion time has significant correlations with total visit duration in recommended songs and in total AOIs, and the correlations were moderately positive (coefficient > 0.3). In other words, when users spent more time moving their eyes within the recommended song areas, the more time they would need to complete the tasks. In contrast, other two effectiveness metrics have no significant correlation with eye movement. Thus, our Hypothesis H3 is partially supported.

To test H4, Spearman's correlation coefficients were calculated for user perception metrics on ordinal scales: task easiness; satisfaction with songs found; preference on seed song. The significant results after Bonferroni correction are shown in Table 3.

Perception	Eye Movement	Coefficient	*p* value
Satisfaction with Songs Found	VisDur in Total	.396	.001*
	VisCnt in SeedSong	-.403	.001*

Table 3. Correlation between eye movement and user perception (Spearman)

As is shown in Table 3, satisfaction with songs found has significantly moderate correlation with visit duration in total AOIs and visit count in seed song (coefficient > 0.3). Interestingly, the correlations were positive and

negative respectively. In other words, the more time users spent in moving their eyes around the AOIs, the more satisfied they would be with the songs they found. However, the more times their eyes visited the seed song area, the less satisfied they would be with the found songs. One possible explanation of the former relationship could be from the perspective of enjoyment [18]: satisfied users presumably enjoyed the task and thus they spent more time looking around on the system interface. For the latter relationship, visiting the seed song area might have been an *effort* one had to pay in order to complete the task (e.g., comparing the seed song to a recommended song), and thus repeated visits to this area would entail more efforts and less satisfaction. As there is no significant correlation between the other two perception metrics (task easiness, preference on seed song) and eye movement, our hypotheses H4 is partially supported.

To test hypothesis H5, we ran a linear regression analysis on predicting user perception from eye movement and user effectiveness. To save space, only significant variables in the prediction models are reported (Table 4). As we can see in Table 4, several eye movement metrics have statistically significance in predicting participants' satisfaction with songs found (one user perception metric). In addition, it is noteworthy that the number of songs found, a user effectiveness measure, was significant in predicting all the three metrics of user perceptions. Thus, our hypothesis H5 is partially supported. This finding that users' eye movement and user effectiveness can contribute to predicting user perception measure has methodological implications in that eye movement and user effectiveness measures can be captured automatically in unobtrusive and objective manners. This can potentially provide novel and reliable methods for detecting user perceptions which have to rely on self-report to collect in traditional methods.

Perception	Metrics	Coefficient Beta	*t*	*p*
Task Easiness	# of songs found	.443	2.400	.021*
Satisfy with Songs Found	VisDur in Player	.533	2.308	.026
	VisDur in Total	.774	3.012	.004
	VisCnt in Search Box	-.984	-2.346	.024
	VisCnt in RecomSong	.718	2.127	.040
	# of songs found	.510	3.505	.001
Like Seed Song	# of songs found	.453	2.572	.014

*: significant at $p < 0.05$ level; R^2 = .334, .586, .394 for the three predictive models respectively

Table 4. Regression analysis on user perception

5.4 Relationship between Eye Movement Measure and User Characteristics

To answer Q3 and test H6, we calculated point-biserial correlation coefficients for user characteristics metrics on binary scales: gender (Male=1, Female=2), and being able to play music instrument (Y=1, N=2). We also ran Spearman's correlation on two ordinal variables, frequency of listening to music and frequency of searching for music. Results with significant correlation after Bonferroni correction are shown in Table 5.

Characteristics	Eye Movement	Coefficient	*p* value
Listening Freq	VisCnt in Total	.436	.001*
	VisCnt in Seed-Song	.485	.001*

Table 5. Correlation between eye movement and user characteristics

As shown in Table 5, frequency of listening to music (one user characteristic) has significant and moderately positive correlation with visit count in Total AOIs and that in the seed song area. The more often users listened to music, the more times their eyes visited the seed song area and all AOIs. As visit count of the eyes is related to attention [4], This result is possibly related to the enthusiasm that these users had in music. Nevertheless, other three user characteristics have no significant correlation with eye movement. Thus, our hypothesis H6 is partially supported.

6. CONCLUSION AND FUTURE WORK

This is an empirical study on using eye tracking method to analyze user interactions with MIR systems. We analyzed eye movement metrics in four AOIs and most of our hypotheses were partially supported by the results. Specifically, some eye movement metrics differed between different retrieval tasks. Furthermore, some eye movement metrics were related to user effective and user perception measures. It is potentially useful in research methodology that some metrics of eye movement and user effectiveness can be used to predict user perception. There was no significant difference on eye movement metrics on the different interface modes, which calls for alternative analysis methods beyond AOI-based ones. Also, through comparing visualized heat maps, we found that fixations of participants may not follow patterns found in text retrieval, which warrants further investigation.

This study aims to stimulate and encourage more work to utilize eye tracking in MIR research. There is much room for future work in investigating eye movement with different MIR tasks, use cases and user groups. It could be paradigm shifting if more findings in future work support deriving users' subjective perception (i.e. satisfaction, perceived difficulty) from unobtrusive and objective measures including eye movement.

7. ACKNOWLEDGEMENT

This study is supported by the JSPS KAKENHI Grant (No. JP16H01756), National Natural Science Foundation of China (No. 61703357), and a General Research Fund from the Research Grants Council of the Hong Kong S. A. R., China (No. HKU 176070).

8. REFERENCES

[1] Berzak, Y., Katz, B., & Levy, R. (2018). Assessing language proficiency from eye movements in reading. In Proceedings of NAACL-HLT (pp. 1986-1996).

[2] Bojko A. (2009). Informative or misleading? Heatmaps deconstructed. In Jacko J. A. (Ed.), Human-Computer Interaction, Part I (pp 30-39). Heidelberg, Berlin: Springer.

[3] Chandra, S., Sharma, G., Malhotra, S., Jha, D. & Mittal, A. P. (2016). Eye tracking based human computer interaction: Applications and their uses. International Conference on Man & Machine Interfacing.

[4] Cole, M. J., Gwizdka, J., Bierig, R., Belkin, N. J., Liu, J., Liu, C. & Zhang, X. (2010, August). Linking search tasks with low-level eye movement patterns. In Proceedings of the 28th Annual European Conference on Cognitive Ergonomics (pp. 109-116). ACM.

[5] Dias, R., Pinto, J. & Fonseca, M. J. (2014, September). Interactive Visualization for Music Rediscovery and Serendipity. In Proceedings of the 28th International BCS Human Computer Interaction Conference on HCI 2014-Sand, Sea and Sky-Holiday HCI (pp. 183-188). BCS.

[6] Farzan, R. & Brusilovsky, P. (2009). Social navigation support for information seeking: If you build it, will they come?. International Conference on User Modeling. Springer-Verlag.

[7] Goeman, J. J. & Solari, A. (2014). "Multiple Hypothesis Testing in Genomics". Statistics in Medicine. 33 (11): 1946–1978.

[8] Gwizdka, J. (2014). Characterizing relevance with eye-tracking measures. Information Interaction in Context Symposium. ACM.

[9] Holm, S. (1979). A simple sequentially rejective multiple test procedure. Scand. J. Statist. 6, 65-70.

[10] Hu, X. & Downie, J. S. (2007). Exploring mood metadata: Relationships with genre, artist and usage metadata. In Proceedings of the 8th Annual Conference of the International Society for Music Information Retrieval (ISMIR), Vienna, Austria (pp.67-72).

[11] Hu, X. & Kando, N. (2012). User-centered measures vs. system effectiveness in finding similar songs. In Proceedings of the 13th Annual Conference of the International Society for Music Information Retrieval (ISMIR), Porto, Portugal (pp. 331-336).

[12] Hu, X. & Kando, N. (2014). Evaluation of music search in casual-leisure situations. In Workshop on Searching for Fun, the Information Interaction in Context conference (IIiX), Regensburg, Germany.

[13] Hu, X., Kando, N. & Yuan, X. (2011). User evaluation of an interactive music information retrieval system. In Proceedings of the 5th Workshop on Human-Computer Interaction and Information Retrieval (HCIR), Mountain View, CA.

[14] Hu, X., Lee, J., Bainbridge, D., Choi, K., Organisciak, P. & Downie, J. S. (2017). The MIREX Grand Challenge: A framework of holistic user experience evaluation in music information retrieval, Journal of the Association for Information Science and Technology, 68(1), 97–112.

[15] Hu, X., Li, F. & Ng, T. D. J. (2018). On the Relationships between Music-induced emotion and physiological signals. In Proceedings of the 19th Annual Conference of the International Society for Music Information Retrieval (ISMIR), Paris, France (pp. 362-369).

[16] Kelly, D. (2009). Methods for evaluating interactive information retrieval systems with users. Foundations and Trends in Information Retrieval, 3(1-2): 1-224.

[17] Lange, E. B., Zweck, F., & Sinn, P. (2017). Microsaccade-rate indicates absorption by music listening. Consciousness and cognition, 55, 59-78.

[18] Laplante, A. & Downie, J. S. (2011). The utilitarian and hedonic outcomes of music information-seeking in everyday life. Library & Information Science Research, 33(3), 202-210.

[19] Lee, J. H. & Price, R. (2016). User experience with commercial music services: An empirical exploration. Journal of the Association for Information Science and Technology, 67(4), 800-811.

[20] Madell, J. & Hébert, Sylvie. (2008). Eye movements and music reading: where do we look next?. *Music Perception: An Interdisciplinary Journal, 26*(2), 157-170.

[21] Moro, R., Daraz, J. & Bielikova, M. (2014). Visualization of Gaze Tracking Data for UX Testing on the Web. CEUR Workshop Proceedings. 1210.

[22] Murakami, M., Sakamoto, T. & Kato, T. (2018, July). Music Retrieval and Recommendation Based on Musical Tempo. In International Conference on Applied Human Factors and Ergonomics (pp. 362-367). Springer, Cham.

[23] Papavlasopoulou, S., Sharma, K. & Giannakos, M. N. (2018). How do you feel about learning to code? Investigating the effect of children's attitudes towards coding using eye-tracking. International Journal of Child-Computer Interaction, 17, 50-60.

[24] Sevcech, J. & Bielikova, M. (2014). User's Interest Detection through Eye Tracking for Related Documents Retrieval. International Workshop on Semantic & Social Media Adaptation & Personalization. IEEE.

[25] Schedl, M., Flexer, A. & Urbano, J. (2013). The neglected user in music information retrieval research. Journal of Intelligent Information Systems, 41(3): 523-539.

[26] Schedl, M. (2016, June). The lfm-1b dataset for music retrieval and recommendation. In Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval (pp. 103-110). ACM.

[27] Sharma, K., Alavi, H. S., Jermann, P. & Dillenbourg, P. (2016, April). A gaze-based learning analytics model: in-video visual feedback to improve learner's attention in MOOCs. In Proceedings of the Sixth International Conference on Learning Analytics & Knowledge (pp. 417-421). ACM.

[28] Shneiderman, B. (1992). Tree visualization with tree-maps: A 2-d space filling approach. In ACM Transaction on graphics, 11(1): 92–99.

[29] Stober, S. & Nürnberger, A. (2010, June). MusicGalaxy: a multi-focus zoomable interface for multi-facet exploration of music collections. In International Symposium on Computer Music Modeling and Retrieval (pp. 273-302). Springer, Berlin, Heidelberg.

[30] The, B. & Mavrikis, M. (2016, April). A study on eye fixation patterns of students in higher education using an online learning system. In Proceedings of the Sixth International Conference on Learning Analytics & Knowledge (pp. 408-416). ACM.

[31] Vandemoortele, S., Feyaerts, K., Reybrouck, M., De Bievre, G., Brône, G. & De Baets, T. (2018). Gazing at the partner in musical trios: a mobile eye-tracking study. Journal of Eye Movement Research, 11(2), 6.

[32] Yang, Y. H. & Teng, Y. C. (2015). Quantitative study of music listening behavior in a smartphone context. ACM Transactions on Interactive Intelligent Systems (TiiS), 5(3), 14.

A CROSS-SCAPE PLOT REPRESENTATION FOR VISUALIZING SYMBOLIC MELODIC SIMILARITY

Saebyul Park **Taegyun Kwon** **Jongpil Lee**
Jeounghoon Kim **Juhan Nam**
Graduate School of Culture Technology, KAIST
{saebyul_park, ilcobo2, richter, miru, juhannam}@kaist.ac.kr

ABSTRACT

Symbolic melodic similarity is based on measuring a pairwise distance between two songs from diverse perspectives. The distance is usually summarized as a single value for song retrieval. This obscures observing the details of similarity patterns within the two songs. In this paper, we propose a cross-scape plot representation to visualize multi-scaled melody similarity between two symbolic music encodings. The cross-scape plot is computed by stacking up a minimum local distance between two segments from each of the two songs. As the layer goes up, the segment size increases and it computes incrementally more long-term distances. This hierarchical representation allows for capturing the location and length of similar segments between two songs in a visually intuitive manner. We show the effectiveness of the cross-scape plot by evaluating it on examples from folk music collections with similarity-based categories and plagiarism cases.

1. INTRODUCTION

Melodic similarity is a key concept in the field of ethnomusicology, music analysis, musicology, music psychology, copyright issues in music, and music information retrieval [13]. From music analysis to content-based retrieval, a great deal of effort and attention have been paid to quantitative measurement of melodic similarity using knowledge from various domains [5, 12, 13, 18–20, 22, 25]. The applications of melodic similarity include song retrieval and classification, music indexing, and music alignment systems [3].

When evaluating the similarity of two songs, it is essential to extract information about what parts and how long they are similar to each other. For example, considering a plagiarism case, a short three-second segment can be regarded as qualitatively significant if it can be easily recognized as a chorus or hook of a popular song [10]. If the similar parts of these two songs are meaningfully similar or *substantially similar* [9], even if the other parts of the two songs are different, people may effectively recognize it as a plagiarized song [10]. However, existing similarity measures often summarize the distance as a single value for song retrieval. This obscures the details of similar pattern within the two songs. This way, even if the two songs has almost identical parts locally, the overall similarity may be diluted by the calculation for the entire song.

In this paper, we tackle this issue by proposing a cross-scape plot representation that visualizes multi-scaled melody similarity between two symbolic music. The cross-scape plot is computed by stacking up a minimum local distance between two segments from each of the two songs. We segment songs from small to large units, and the local similarity is performed by a sequence-based similarity algorithm for all possible segments of the two songs. As the layer goes up, the segment size increases and it computes progressively more long-term distances. This results in a hierarchical visual representation with a triangular or trapezoidal shape.

The cross-scape plot provides rich information in an intuitive way, which a single value derived from most similarity measures cannot provide. Even with a simple glance, we can observe various characteristics of similarity, such as the location and length of similar parts by the pattern shape in the plot, and the overall similarity by the color. Even when the melody has the same similarity value in a single measurement, the visual representation can be very different. We show this aspect with examples from classified folk songs Meertens Tune Collections (MTC-ANN) [21] and music plagiarism cases. We also validate the multi-scaled melodic similarity by summarizing it into a similarity value and conduct a melody classification experiment on MTC-ANN.

2. MULTI-SCALE SIMILARITY ANALYSIS

This section describes three steps of multi-scaled similarity analysis to obtain the cross-scape plot.

2.1 Feature Extraction

Many choices of features have been extracted for melodic similarity analysis including pitch, pitch interval, duration, onset, duration weighted pitch sequence or metric weights [7]. In this study, we use pitch interval as a pitch feature and inter-onset interval ratio as a rhythmic feature. Pitch interval is computed by the pitch difference between two

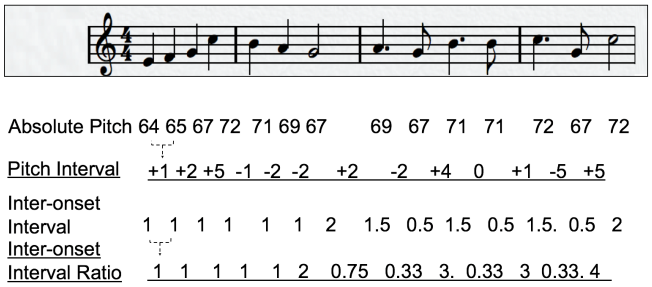


Figure 1: An example of pitch interval and inter-onset interval ratio of the rhythm extracted from melody.

successive notes. This is invariant to key of the songs. The pitch interval is limited to the range within 2 octaves in our setting. Inter-onset interval (IOI) ratio is computed from the relative IOI of three successive notes. This is invariant to tempo of the songs. The IOI is quantized to the unit of 0.5 and limited not to exceed 4 in our setting. Figure 1 shows that a melody is converted to a sequence of the pitch and rhythm features. We analyze similarity for both feature sequences independently.

2.2 Multi-segmentation

Multi-segmentation is the process of dividing the melody sequence into smaller sub-sequences of all sizes. This idea was inspired by Sapp's work [15, 16]. Sapp proposed a plotting method called *scape plot* that can display the results of an analysis of segments of varying lengths as a single image. The scape plot is a simple but effective method to understand similarity patterns that occur on every possible timescale. The scape plot was named in *landscape* because it shows small-scale features similar to the foreground of the picture, as well as large-scale features similar to the background. The original scape plot method was designed for structural analysis of harmony in musical scores [15] and has been applied in a variety of ways in different contexts, for example, tonality analysis [14, 27], musical performance analysis [17, 24] and audio thumbnailing [8].

Following the approach, we segment the sequence into multi-scale units from the smallest to the entire sequence. Figure 2 is an example of dividing the features sequences into the different sizes. The sequence $ABCDE$ is divided into 15 sub-sequences by sequentially grouping them into units of the smallest unit (that correspond to segment size of 1) to the maximum of 5, which is the length of the entire sequence (that correspond to segment size of 5). Formally speaking, given the sequence of melody features, $S = (x_1, x_2, ..., x_{|S|})$ where $|S|$ is the length of S, let s_k^n denote the kth sub-sequence with the length n of S so that $s_k^n = (x_k, x_{k+1}, ..., x_{k+n-1}), 1 <= k <= |S| - n + 1$. These sub sequences s^n are used to obtain the local distance between two songs. The following section describes how to calculate the local similarity using these multiple sub-sequences of melody.

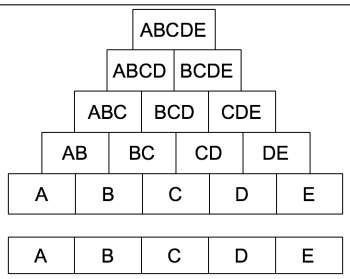


Figure 2: Example of multiple segments of melody. The sequence $ABCDE$ is divided into 15 sub-sequences.

2.3 Similarity Calculation

We use the multi-segmentation to compute multi-scale similarity between two songs. We first define the distance measure between two segments.

2.3.1 Distance Measure

In order to obtain the local distance of each segment, we adopt edit distance [11], the most commonly used string matching similarity calculation method in music research [5]. The edit distance, also known as the Levenshtein distance, is a metric that computes the minimum number of operations needed to transform one sequence into the other.

The operations between sequences include $deletion$, $insertion$, and $substitution$ of symbols. To find out the minimum path to obtain edit distance, we use dynamic programming algorithm known as Wagner—Fischer algorithm. For the compared melodic sequence $S_a = (a_1, a_2, ...a_n)$, and the $S_b = (b_1, b_2, ...b_m)$, where a_i and b_j are features of the melody. [1] We set 1 to the cost of $deletion$, $insertion$ and 2 to the cost of $substitution$. Let $d(i, j)$ denote the edit distance of sub-sequence $(a_1, a_2, ...a_i), (b_1, b_2, ...b_j)$. The calculation of edit distance $D(S_a, S_b) = d(n, m)$ is defined using a recursive algorithm as below [6]:

$$
\begin{aligned}
&d(i,0) = i, \\
&d(0,j) = j, \\
&d(i,j) = \begin{cases} d(i-1,j-1) & \text{if } a_i = b_j \\ \min \begin{cases} d(i-1,j)+1 \\ d(i,j-1)+1 \\ d(i-1,j-1)+2 \end{cases} & \text{if } a_i \neq b_j \end{cases} \\
&\qquad\qquad \text{for } 1 \leq i \leq n, 1 \leq j \leq m
\end{aligned} \tag{1}
$$

The final local distance between two sub-sequences, denoted by $\overline{D}$, is the normalized value of of $D(S_a, S_b)$ divided by the maximum possible distance (i.e., when the two sequences are completely different) as follows:

$$\overline{D}(S_a, S_b) = \frac{D(S_a, S_b)}{(|S_a| + |S_b|) \times 2} \tag{2}$$

[1] S can be either a whole melody, or a sub-sequence of it, and it can be pitch interval and the inter-onset interval ratio.

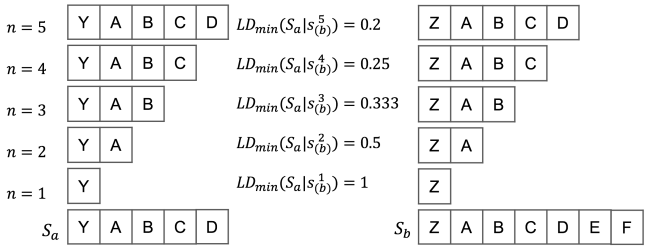


Figure 3: An example of minimum local distance calculation between S_a and S_b

We particularly define the minimum local distance between a melody S_a and $s^n_{k(b)}$, a sub-sequence of S_b at the kth position with length n:

$$LD_{min}(S_a|s^n_{k(b)}) = \min_{\forall s^n_{(a)} \in S_a} \overline{D}(s^n_{(a)}, s^n_{k(b)}) \quad (3)$$

Because $LD_{min}(S_a|s^n_{k(b)})$ is a minimum distance between a segment $s^n_{k(b)}$ and all possible sub-sequences of S_a with the same length n, it is not a commutative operation. However, we call it distance because it shows the distance between $s^n_{k(b)}$ and most-matched segment in S_a. For example, suppose that there are two melody sequences to be compared, S_b is a longer melodic sequence, and S_a is shorter (see Figure 3). S_a and S_b can be divided into sub-sequences. [2] The figure shows the LD_{min} value of the first segment for each n. For example, $LD_{min}(S_a|s^1_{1(b)})$ is 1 because in the first sub-sequence with $n = 1$ ("Z" in the example) does not match any sub-sequences of S_a. As the number of notes shared by S_a and S_b increases, $LD_{min}(S_a|s^n_{k(b)})$ becomes smaller, that is, the degree of similarity increases. Note that since the lengths of two melodies are different, the maximum size of the n is the length of S_a (the shorter).

2.3.2 Multi-scale Similarity Stack

After the operation for all segments is performed, it forms a multi-scale similarity stack for S_a and S_b. The multi-scale similarity stack is important for similarity analysis, and also for *cross-scape plot* visualization (see Figure 5) which will be discussed in Section 3. In this study, we obtain a multi-scale similarity stack based on segments of longer songs. [3] Thus, the maximum of the x-axis of the multi-scale similarity stack becomes the length of the longer song, while the maximum of the y-axis is the length of the shorter melody, respectively. For a given pair of melodic sequences S_a and S_b, with $|S_a| \leq |S_b|$ we can calculate $LD_{min}(S_a|s^n_{k(b)})$ for all sub-sequences with length n for all possible n in S_b against S_a. As a result, we create a two-dimensional multi-scale similarity stack (MSS) defined as below [4] :

$$MSS(S_a|S_b) = [a_{n,k}]_{|S_a|\times|S_b|},$$
$$a_{n,k} = 1 - LD_{min}(S_a|s^n_{k(b)}) \quad (4)$$

We can also summarize MSS and calculate the overall similarity as a single value. In this case, we discard a subset of elements in MSS which corresponds to similarity between segments less than three notes. To compensate low similarity values between longer sequence, we also define a weighted multi-scale similarity stack, $wMSS$, where the weight λ is a function of n. Finally, the overall similarity Sim_{S_a,S_b} of two songs is obtained by averaging $wMSS$. This overall similarity is derived in the same way for both pitch interval and inter-onset interval ratio of rhythm.

$$wMSS(S_a|S_b) = [\lambda(n) \times a_{n,k}]_{|S_a|\times|S_b|},$$
$$a_{n,k} = 1 - LD_{min}(S_a|s^n_{k(b)}), \quad (5)$$
$$\lambda(n) = 0.5 + 0.5 \times n/|S_a|$$

Finally, the overall similarity $Sim(S_a, S_b)$ of two songs is obtained by averaging the weighted multi-scale similarity stack:

$$Sim(S_a, S_b) = \text{mean}((wMSS(S_a|S_b))_{n,k})_{n\geq 3}) \quad (6)$$

This overall similarity can be computed in the same way for both pitch and rhythm features but using different weights. We implemented the algorithm using MATLAB and MIDI Toolbox [4]. The source code is available at the Github repository. [5]

3. CROSS-SCAPE PLOT

This section introduces the visualization method using the multi-scaled similarity of two melody sequences. As aforementioned, this method was inspired by [15, 16]. While previous studies using the original scape-plot method focused on analyzing a single song based on self-similarity, this study applies the idea to analyzing the similarity of two different songs.

3.1 Procedure

Figure 4 illustrates the procedure of drawing the cross-scape plot using the multi-scale similarity stack. We arrange the sub-sequences on top of each other to have the hierarchical patterns. Also, to increase readability, the multi-scale similarity stack is center-aligned. Unlike the original scape plot, the cross-scape plot has a trapezoidal shape because it is computed from two songs with different lengths. The more different the lengths of the two songs, the more likely the shape of the trapezoid. The x-axis indicates the position of the longer melody. The y-axis represents the segment size.

The cross-scape plot displays local similarity with a color. As the local similarity is higher, the color becomes darker, and vice versa. To distinguish the two features, we

[2] For readability, features are noted as letters.

[3] The opposite case, comparing segments of the song based on shorter songs, is also possible, but this is not covered in this study.

[4] Elements of the matrix only valid for $1 \leq n \leq |S_a|, 1 \leq k \leq |S_b| - n + 1$

[5] `https://github.com/saebyulpark/cross_scapeplot_visualization`

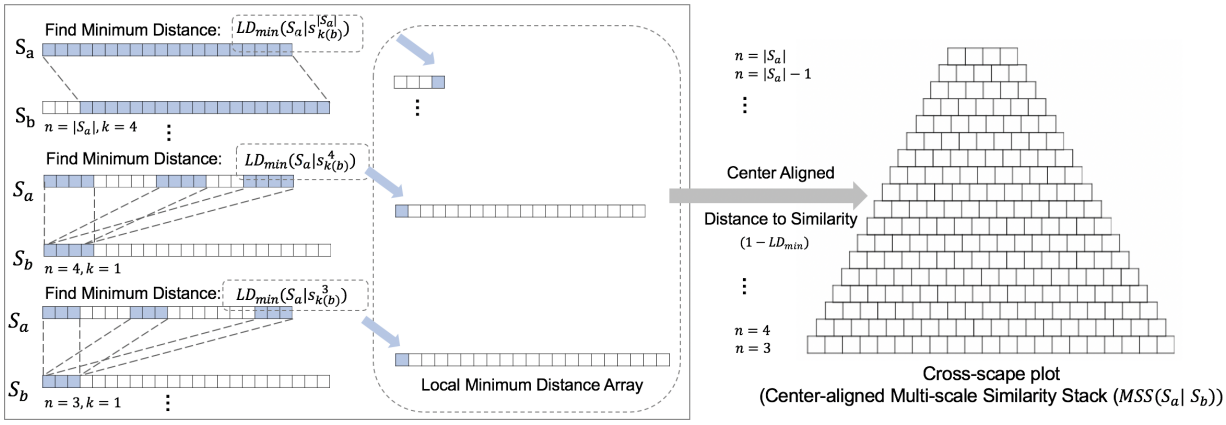


Figure 4: An illustration of steps for drawing a cross-scape plot.

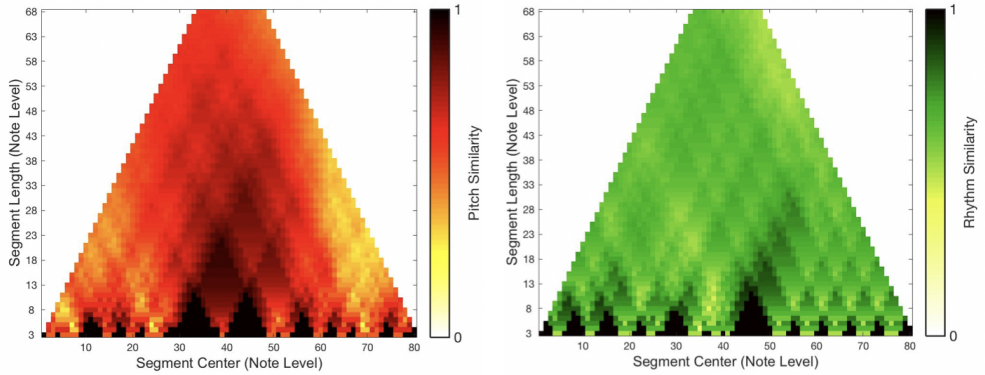


Figure 5: An example of a cross-scape plot by comparing similarities between two songs. The pixel value and the density of the color indicate the similarity score between the two sub-sequences.

set red to the cross-scape plot of the pitch feature and green to that of the rhythm feature. These different colors allows to easily recognize where and how the two songs are similar simply inspecting the plot. Figure 5 is an example of cross-scape plots where two songs are taken from the same tune family in the MTC-ANN datasets. The left part is the plot for the pitch feature and the right is for the rhythm feature. Through the cross-scape plot on the left, we can see that these two pieces are very similar in the middle parts. in the middle parts of the pitch feature, and the uppermost color suggests that the overall similarity is between 0.6 and 0.7. On the right plot, the rhythm feature shows a similarity with a certain periodicity in the latter part. Also, the slightly right part of the center represents a high similarity (black or 1) in the same way as the pitch feature, so we can assume that the melody of a particular part is nearly identical. Overall, a type of tendency is found where a large segment has a larger distance and a smaller segment has a smaller local distance. This is because the smaller the segment size is less distinct, the higher the probability that the segment is present in both songs.

3.2 Case Study: Toy Example

Figure 6 is a toy example of the cross-scape plot showing the similarity between the melodies generated such that they have the same edit distance between pitch interval sequences. One pair has exactly matching at a specific part of the melody (Song 1 and 2, left part of Figure 6). In the other pair, the specific parts are not exactly the same, but they are slightly similar overall (Song 1 and 3, right part of Figure 6). In this example, the edit distance alone yields the same distance between the melodies (in both cases $D = 12$ for pitch interval sequences). However, we can see that the first half of the song is exactly the same for both pitch and rhythm in the case of songs 1 and 2. On the other hand, the pitch features of Songs 1 and 3 are generally similar at random, but the rhythm is apparently similar to the latter half of the song. In this way, we can see various perspectives on the similarity of melody features even though they have the same similarity value.

3.3 Case Study: Plagiarism Cases

The benefits of cross-scape plots become more important for issues that require a certain level of qualitative judgment, such as originality or substantial similarity of plagiarism. Figure 7 shows cross-scape plots of the cases in our ongoing plagiarism research project. These two examples are cross-scape plots of songs with court rulings on copyright infringement. In both cases (*Mood Music v. De Wolfe* [6] : left side of figure and *MCA Music v. Earl Wilson* [7] : right side of figure), copyright infringement is ruled by the court or settled between the parties, where plagia-

[6] *Mood Music v. De Wolfe*, 1. Ch. 119 (1976).
[7] *MCA Music v. Earl Wilson* 425 F. Supp. 443 (S.D.N.Y. 1976)

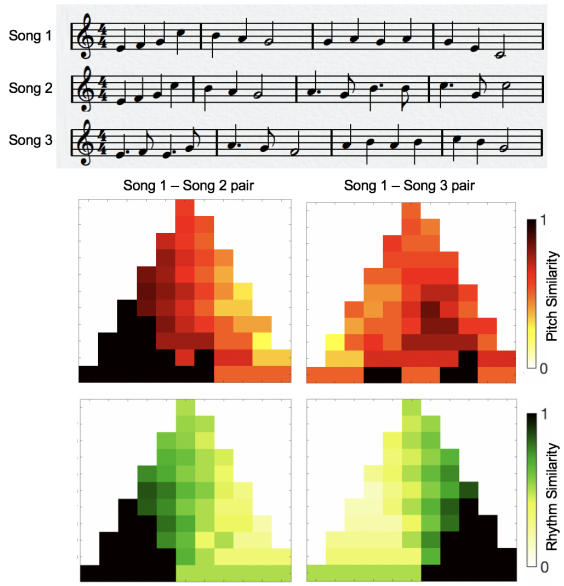


Figure 6: Toy example of an exact match pair and an overall similar pair of cross-scape plots

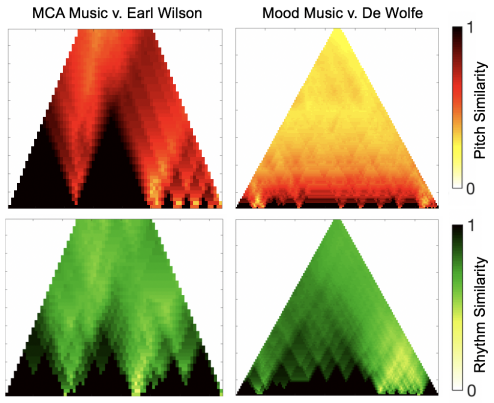


Figure 7: Cross-scape plots of plagiarism cases (*Mood Music v. De Wolfe*: top of the figure and *MCA Music v. Earl Wilson*: bottom of the figure).

rism is acknowledged. We show these cases because these are either the most confusing (case 1 where 93.5% of the participants were confused) or less confused (case 2 with 0% confusion rate) cases in a trial experiment in which the implicit memory task is performed (High confusion means two songs are very similar to each other). [8]

Although both are cases of copyright infringement, the ways in which similarities appear are very different. In *Music v. De Wolfe*, most parts of the song are almost identical on all sides of the pitch and rhythm features. On the other hand, in *MCA Music v. Earl Wilson*, the rhythm features are generally similar, but the pitch features are different. Thereby, for the case *MCA Music v. Earl Wilson*, it can be assumed that the plagiarism judgment is based on other factors such as rhythm, harmony, arrangement or lyrics rather than the tonal characteristics of the melody. [9]

[8] This experiment is an ongoing project; a similar experiment and result can be found in [28]

[9] Indeed, in spite of the defendant's arguments contending fair use of parody, infringement was considered based on identical bass line, a general harmonic similarity and certain specialized rhythmic patterns (https://blogs.law.gwu.edu/mcir/case/mca-music-v-earl-wilson/).

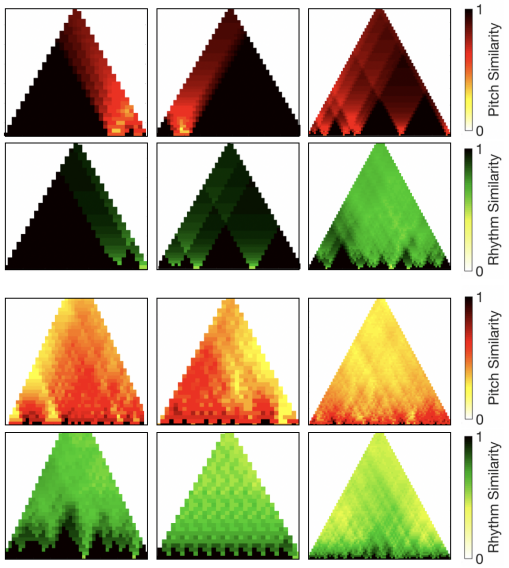


Figure 8: Cross-scape plots on examples of MTC-ANN. The three pairs on the top belong to the same tune family whereas the three on the bottom belong to different tune families.

In this way, gaining rich information about similarities can be of great help in making this kind of intuitive and sophisticated decision regarding of similarity.

3.4 Case Study: MTC-ANN

Figure 8 shows more examples of cross-scape plots showing how different pairs of MTC-ANN represent similarities of different characteristics. The three pairs on the top are similarities between songs in the same tune family, the three on the bottom are for those included in other tune families. The similarity of the two songs decreases from left to right. Looking at these pairs, we can observe that similarities in pitch, rhythm, position, and lengths of similar parts appear in a variety of ways, even in pairs in the same group.

4. MELODY CLASSIFICATION

We can utilize the outcome of cross-scape plots for the symbolic melodic similarity task. In this section, we conduct the task with a classification experiment using summarized similarity derived from the multi-scale similarity stack.

4.1 The dataset

We used the annotated corpus of the Meertens Tune Collection (MTC-ANN), version 2.0.1 [21] for the evaluation. It contains 360 melodies divided into 26 tune families annotated by musicological experts. The MTC-ANN dataset has been used in a variety of music studies. This allowed us to compare our model with recent studies of measuring melodic similarity.

	CSR	AUC	MAP
Pitch Interval	0.95	0.89	0.68
Inter-onset Interval Ratio	0.76	0.82	0.49
Combined	0.96	0.91	0.71

Table 1: Results for the proposed model

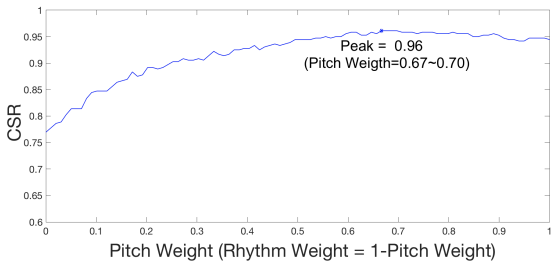


Figure 9: Accuracy that varies with the weighting value of the pitch feature and the rhythm feature.

4.2 Similarity Measures

We performed the classification of MTC-ANN songs into the same tune family with each of pitch and rhythm features or their combinations. We compared the performance to those from previous work that conducted the same task over the last three years [1, 2, 26].

4.3 Evaluation Metrics

We used the three evaluation metrics following the previous works.

Classification Success Rate (CSR) represents a correctly classified rate of the total when the melody is indexed into the same tune family using k-Nearest Neighbors (k-NN) classifier to which the melody with the highest similarity belongs. This evaluation method was used in all reference papers to be compared.

Area Under the Curve (AUC) is calculated by adaptively modifying the decision threshold of the similarity score. In this case, the songs within the same tune family is assigned to a ground truth. The ranking scores are then calculated and averaged.

Mean Average Precision (MAP): has the same ground truth as the one used in the AUC calculation. The songs are sorted by the similarity score and the songs that have the same tune family with the query song are treated as a correct one. We repeat this procedure by adjusting the number of correct answers and averaged the whole scores.

4.4 Evaluation Results

Table 1 shows the results of the proposed model to classify 360 songs of MTC-ANN into 26 family tunes, using pitch features, rhythm features and both of them. In addition, Figure 9 shows the plot of CSR when the relative weight in computing the overall similarity (λ in Equation 6) between two features changes from 0 to 1. the highest score is obtained when the weight (pitch weight) ranges between 0.67 and 0.7.

	CSR	AUC	MAP
Boot [1]	0.92	-	-
Bountouridis [2]	0.94	-	0.70
Walshaw [26]	0.93	0.89	-
Proposed	0.96	0.91	0.71

Table 2: Comparison with previous studies using the MTC-ANN dataset

As a result of the classification evaluation, the melody of a family group is sufficiently classified by a single pitch feature, whereas a rhythm feature has a limitation in performing a classification task alone. In addition, classification with pitch and rhythmic features combined with appropriate weights has achieved the highest performance, which indicates that pitch and rhythm together complement the similarity of the melody, although the pitch feature contributes to it more.

4.5 Comparison with Recent Studies

Table 2 provides a comparison with recent studies that performed classification of MTC-ANN using the entire melody. The results show that our approach not only provides the visualization but also comparable performance to them.

5. DISCUSSION AND CONCLUSION

We proposed a cross-scape plot which visualizes symbolic melodic similarity between two songs based on multi-segmentation analysis. With the examples of folk songs and plagiarism cases, we showed that the cross-scape plot can reveal the melodic similarity in various ways. We also performed a classification task of the MTC-ANN dataset based on the summarized similarity derived from the method.

The proposed method used edit distance but this can be replaced with with other sequence-based distances. On the other hand, however, it has already been shown that many studies yield high classification results when sequence alignment techniques are used as a measure of similarity between folk songs [23]. Thus, the high classification performance in this study may reflect the advantages of the alignment method used as the main distance. Therefore, the performance evaluation of the present method as a measure of similarity requires further study to experiment with more data sets and distance measures. Besides, since only segments with the same length are compared and are performed based only on shorter melodies, there is a limitation that some loss of information can occur. There is also a disadvantage that this method is time-consuming because it repeatedly performs calculations among all the segments. Despite several limitations, we believe that this multi-scaled approach provides a wealth of insight that will help us to understand the properties of similarity.

6. REFERENCES

[1] Peter Boot, Anja Volk, and W Bas de Haas. Evaluating the role of repeated patterns in folk song classification and compression. *Journal of New Music Research*, 45(3):223–238, 2016.

[2] Dimitrios Bountouridis, Daniel Brown, Frans Wiering, and Remco Veltkamp. Melodic similarity and applications using biologically-inspired techniques. *Applied Sciences*, 7(12):1242, 2017.

[3] J Stephen Downie. Music information retrieval. *Annual Review of Information Science and Technology*, 37(1):295–340, 2003.

[4] Tuomas Eerola and Petri Toiviainen. *MIDI Toolbox: MATLAB Tools for Music Research*. University of Jyväskylä, Jyväskylä, Finland, 2004.

[5] Kuldeep Gurjar and Yang-Sae Moon. Comparative analysis of music similarity measures in music information retrieval systems. *Journal of Information Processing Systems*, 14(1), 2018.

[6] Dan Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge university press, 1997.

[7] Berit Janssen, Peter van Kranenburg, and Anja Volk. A comparison of symbolic similarity measures for finding occurrences of melodic segments. In *Proc. International Conference on Music Information Retrieval (ISMIR)*, pages 659–665, 2015.

[8] Nanzhu Jiang and Meinard Müller. Towards efficient audio thumbnailing. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5192–5196, 2014.

[9] Stephanie J Jones. Music copyright in theory and practice: An improved approach for determining substantial similarity. *Duquesne Law Review*, 31:277, 1992.

[10] Jeffrey F Kersting. Singing a different tune: Was the sixth circuit justified in changing the protection of sound recordings in bridgeport music, inc. v. dimension films. *University of Cincinnati Law Review*, 74:663, 2005.

[11] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[12] Daniel Müllensiefen and Marc Pendzich. Court decisions on music plagiarism and the predictive value of similarity algorithms. *Musicae Scientiae*, 13(1_suppl):257–295, 2009.

[13] Daniel Müllensiefen and Klaus Frieler. Cognitive adequacy in the measurement of melodic similarity: Algorithmic vs. human judgments. *Computing in Musicology*, 13(2003):147–176, 2004.

[14] Meinard Müller and Nanzhu Jiang. A scape plot representation for visualizing repetitive structures of music recordings. In *Proc. International Conference on Music Information Retrieval (ISMIR)*, pages 97–102, 2012.

[15] Craig Stuart Sapp. Harmonic visualizations of tonal music. In *International Computer Music Conference (ICMC)*, volume 1, pages 419–422, 2001.

[16] Craig Stuart Sapp. Visual hierarchical key analysis. *Computers in Entertainment (CIE)*, 3(4):1–19, 2005.

[17] Craig Stuart Sapp. Comparative analysis of multiple musical performances. In *Proc. International Conference on Music Information Retrieval (ISMIR)*, pages 497–500, 2007.

[18] Patrick E Savage and Quentin D Atkinson. Automatic tune family identification by musical sequence alignment. In *Proc. International Conference on Music Information Retrieval (ISMIR)*, pages 162–168, 2015.

[19] Patrick E Savage, Charles Cronin, and Daniel Müllensiefen Quentin D Atkinson. Quantitative evaluation of music copyright infringement. In *Proceedings of the 8th International Workshop on Folk Music Analysis (FMA2018)*, pages 61–66, 2018.

[20] Rainer Typke. Music retrieval based on melodic similarity. *Ph.D thesis*, 2007.

[21] Peter van Kranenburg, Berit Janssen, and Anja Volk. The meertens tune collections: The annotated corpus (MTC-ANN) versions 1.1 and 2.0.1, 2016.

[22] Peter van Kranenburg, Anja Volk, and Frans Wiering. A comparison between global and local features for computational classification of folk song melodies. *Journal of New Music Research*, 42(1):1–18, 2013.

[23] Peter van Kranenburg, Anja Volk, Frans Wiering, and Remco C Veltkamp. Musical models for folk-song melody alignment. In *Proc. International Conference on Music Information Retrieval (ISMIR)*, pages 507–512, 2009.

[24] Carlos Vaquero. A quantitative study of seven historically informed performances of Bach's BWV1007 prelude. *Early Music*, 43(4):611–622, 2015.

[25] Valerio Velardo, Mauro Vallati, and Steven Jan. Symbolic melodic similarity: State of the art and future challenges. *Computer Music Journal*, 40(2):70–83, 2016.

[26] C. Walshaw. Tune classification using multilevel recursive local alignment algorithms. In *Proc. 7th Intl Workshop on Folk Music Analysis (FMA)*, pages 80–87, 2017.

[27] Christof Weiß and Meinard Müller. Quantifying and visualizing tonal complexity. In *Conference on Interdisciplinary Musicology (CIM)*, 2014.

[28] Anna Wolf and Daniel Müllensiefen. The perception of similarity in court cases of melodic plagiarism and a review of measures of melodic similarity. In *Int. Conf. of Students of Systematic Musicology (SysMus)*, 2011.

JOSQUINTAB: A DATASET FOR CONTENT-BASED COMPUTATIONAL ANALYSIS OF MUSIC IN LUTE TABLATURE

Reinier de Valk[1] **Ryaan Ahmed**[2] **Tim Crawford**[1]
[1] Department of Computing, Goldsmiths, University of London, [2] Digital Humanities, MIT
r.f.de.valk@gold.ac.uk, rahmed@mit.edu, t.crawford@gold.ac.uk

ABSTRACT

An enormous corpus of music for the lute, spanning some two and half centuries, survives today. Unlike other musical corpora from the same period, this corpus has undergone only limited musicological study. The main reason for this is that it is written down exclusively in lute tablature, a prescriptive form of notation that is difficult to understand for non-specialists as it reveals little structural information. In this paper we present JOSQUINTAB, a dataset of automatically created enriched diplomatic transcriptions in MIDI and MEI format of 64 sixteenth-century lute intabulations, instrumental arrangements of vocal compositions. Such a dataset enables large-scale content-based computational analysis of music in lute tablature hitherto impossible. We describe the dataset, the mapping algorithm used to create it, as well as a method to quantitatively evaluate the degree of arrangement (goodness of fit) of an intabulation. Furthermore, we present two use cases, demonstrating the usefulness of the dataset for both music information retrieval and musicological research. We make the dataset, the source code, and an implementation of the mapping algorithm, runnable as a command line tool, publicly available.

1. INTRODUCTION

An enormous corpus of music for the lute, roughly spanning the sixteenth, seventeenth, and first half of the eighteenth century, and containing an estimated 60,000 individual pieces in circa 860 sources, printed and manuscript, survives today [27]. Although the lute was one of the most widely used instruments during these two and a half centuries, and this corpus is thus extremely rich in information about daily musical practice, up until the present day it has not undergone the same critical musicological study as other musical corpora from the same period. The main reason for this is that the notation used to write down lute music, lute tablature, is notoriously difficult to understand for non-specialists [18, 28]. This is because lute tablature is a purely *prescriptive* form of notation: like all forms of instrumental tablature [12], it merely provides the actions a player must take—in this case, where to place the fingers on the fretboard and which strings to pluck—rather than the sounds and musical edifice these actions produce, which a *descriptive* form of notation does [20, 29]. In practice, this means that lute tablature, as opposed to mensural forms of notation, conveys virtually no information about the structure, polyphonic or other, of the music it encodes, making it hard to comprehend *a prima vista*.

In this paper we propose to address the problem of the marginal position of music in lute tablature within musicological research by providing JOSQUINTAB, a dataset of 64 automatically created *enriched diplomatic transcriptions*—literal transcriptions annotated with voice, key, and mensuration information—of a selection of sixteenth-century lute and vihuela (the lute's Spanish counterpart) *intabulations*, instrumental arrangements of vocal compositions, as well as a mapping algorithm for creating such a dataset, and an implementation thereof, runnable as a command line tool. The proposed dataset enables large-scale content-based computational analysis of music in lute tablature, which, in the absence of some notion of the music's polyphonic structure, has hitherto not been possible. The command-line tool allows researchers to extend the dataset by creating their own transcriptions.

In summary, the main contributions of this paper are:

- a dataset of 64 automatically created enriched diplomatic transcriptions of lute intabulations, each in the original and in an unornamented version;
- a mapping algorithm for creating such a dataset;
- an implementation of the mapping algorithm, runnable as a command line tool.

Furthermore, we provide:

- a method to quantitatively evaluate the degree of arrangement (goodness of fit) of an intabulation, which is incorporated in the mapping algorithm;
- two use cases, demonstrating the usefulness of the dataset for both music information retrieval (MIR) and musicological research.

In the spirit of open science and reproducible research, we aim to make the resources we contribute FAIR (findable, accessible, interoperable, and reusable) [32]. We do so by using *recommended formats*—MIDI and MEI—for the dataset, [1] and by making the dataset, the source code for the mapping algorithm, and the command line tool publicly available.

[1] https://www.loc.gov/preservation/resources/rfs/

The remainder of the paper is structured as follows. In Section 2, we sketch the scholarly and scientific background against which this research is carried out. In Section 3 we describe the contents of JOSQUINTAB and the preprocessing that the data we build on requires, and in Section 4 the mapping algorithm by means of which we create JOSQUINTAB, the method to evaluate the dataset quantitatively, and the command line tool. Section 5 is dedicated to the use cases, and in Section 6, conclusions and directions for future work are presented.

2. BACKGROUND IN MUSICOLOGY AND MIR

2.1 Lute tablature: A brief introduction

When in the late fifteenth century lutenists began playing polyphonic music on their instrument, this entailed a need for an accommodating, score-like, notation. Several approaches were developed simultaneously, crystallising into three principal systems, whose modern names refer to their regions of origin: French, Italian, and German lute tablature [1]. [2] From these three, the French system would outlive the other two, from the seventeenth century on becoming the primary system. The former two systems use letters (French) or numbers (Italian), indicating where on the fretboard the fingers must be placed, on a six-line 'staff', each line of which represents a *course* (string pair) to be plucked. The German system is more abstract—it does not use a staff—and consists of a collection of glyphs (letters, numbers, and special characters, often varying per author) representing unique course-fret combinations.

2.2 Transcription practices

Figure 1, which shows an example of the Italian system, perfectly demonstrates the problem with lute tablature. Below a deceivingly simply notational surface, structurally rather complex music may lie hidden—and it requires a significant amount of expertise to infer the information not notated. The traditional musicological solution to 'unlock' this corpus for research is the preparation of scholarly editions, commonly in paper format, in modern music notation. Some of these are 'literal' or *diplomatic* transcriptions, recording the information given in the source exactly as it appears [17] (i.e., providing only pitch and minimal duration information), thus involving a minimum of interpretation but in turn revealing little about the polyphonic structure of the music, while others are full-fledged polyphonic transcriptions, involving a maximum of interpretation, and therefore often accompanied by a critical apparatus motivating the choices made.

Preparing scholarly editions is time-consuming and specialist work—all the more so in the latter case. This is one of the reasons why recent times have seen attempts at automatic transcription of lute tablature. Computers are ideally suited for fast (and large-scale) literal transcription—in fact, this can be achieved fairly easily with many of the music notation software packages, proprietary or open-source, available today. Automatic *polyphonic* transcription of lute tablature, however, has proven to be quite challenging [8, 13, 14], and in MIR research is still considered an open problem. Thus, despite ongoing efforts in musicology and MIR—both concerning content-based [22, 23] and bibliographical research [11]—only a fraction of the lute repertory has been explored up until the present day, and much work remains to be done.

2.3 Sixteenth-century lute music and intabulations

Sixteenth-century lute music is generally considered to fall into three categories [3]: (i) original compositions, i.e., idiomatic, abstract works in varying degrees of counterpoint, such as fantasias, ricercares, or preludes; (ii) settings of dance tunes; and (iii) intabulations, i.e., instrumental arrangements of vocal compositions. It is the latter category—by far the largest of the three, by itself taking up at least approximately half of the music [3]—we focus on in this paper. Up until today, intabulations have largely escaped in-depth musicological research—mostly on the grounds that as non-original works, they would be inferior compositions, which, especially when heavily ornamented, served no other goal than to demonstrate the player's virtuosity. While it is doubtlessly true that certain arrangements are less exciting than others, a fact that should not be overlooked is that intabulations are highly informative about contemporary compositional and performance practices [6,18,19,26], giving insights into, e.g., ornamentation techniques [4] or the application of *musica ficta*, i.e., chromatic alterations not notated in vocal music [5].

Intabulations generally remain close to their vocal models: contemporary treatises on the subject of intabulating prescribe that the intabulator take over as much of the vocal model as is technically possible on the instrument [24]; several existing in-depth studies of selected intabulations [2, 4, 5, 25, 30] support this. It is exactly the resulting close relation between model and intabulation that facilitates the mapping approach presented in this paper.

3. JOSQUINTAB

As its name reveals, the dataset exclusively contains intabulations of vocal compositions by Josquin des Prez (c. 1450-1521), an earlier-generation composer highly popular among sixteenth-century intabulators [3, 21]. According to the *New Josquin Edition*, no less than 18 chansons, 17 motets (most of them in multiple parts), various parts of 17 mass sections, and even eight complete masses have been intabulated [16]—adding up to at least 60 works, depending on how one counts.

In creating the dataset, which we do by mapping machine-readable tablature encodings (in TabCode format [10, 28]) onto MIDI renderings of editions of the vocal models (as explained in detail in Section 4), [3] we build on two existing, high-quality data repositories. The tablature encodings were retrieved for a previous study [22]

[2] The vihuela players generally adopted the Italian system; vihuela tablature, especially the upside-down variant that equals modern guitar tablature, is sometimes referred to as *Spanish tablature*.

[3] See also `http://www.ecolm.org/`.

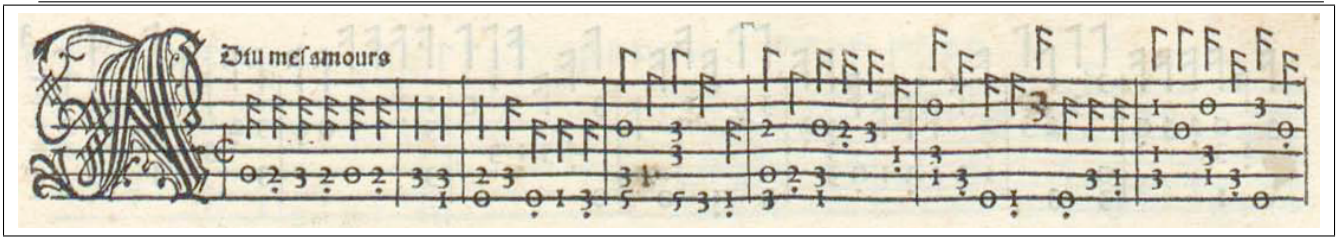


Figure 1. Francesco Spinacino, *Intabulatura de lauto, Libro primo* (Venice, 1507), *Adiu mes amours*, first system.

from Sarge Gerbode's Lute Page, [4] a curated repository of encodings made by amateur enthusiasts; the MIDI renderings of editions of the vocal models (as well as scores in PDF format of the same editions, used for reference) are retrieved from the GitHub repository for the Josquin Research Project (Center for Computer Assisted Research in the Humanities, Stanford University), [5] which aims to provide ways to search and analyse scholarly edited scores of polyphonic early music. [6]

Our eventual goal is for the dataset to contain transcriptions of *all* intabulations of Josquin compositions, including all dubious attributions (but excluding the spurious ones). This inclusive approach is taken as scholarly opinion on the matter of the authenticity of a significant number of pieces tends to change constantly [31]. [7] However, while the Josquin Research Project repository provides editions of all vocal models, the Lute Page repository only contains intabulations of nine of the chansons, 10 of the motets (in 21 parts in total), and 12 of the mass sections parts, where approximately half of the chansons and motet parts have been intabulated more than once. (For practical reasons, for now we ignore the eight complete masses, all of which appear in a single print and are therefore somewhat of an exception; moreover, we only look at printed sources, facsimiles of which, needed for error checking, are generally easier to come by.) As Table 1 shows, this yields a dataset of 64 pieces (60658 data points). Each piece is represented by (i) a machine-readable encoding of the tablature in TabCode format; (ii) a MIDI rendering of the vocal model; (iii) a tuple (see below) of MIDI renderings of the created transcription; (iv) a tuple of MEI renderings (containing score and tablature) of the created transcription; and (v) a CSV file with the mapping details. [8]

In many of the intabulations, a 'net' of ornamentation has been added to the original notes, like a superstructure imposed on the music [5]. We provide each transcription both in the ornamented, original version and in a version stripped of the ornamental net (the procedure by which this is done is explained in more detail in Section 4). The latter is expected to be more useful for analysis tasks that are complicated by large amounts of ornamentation, such as the one in Use Case 2 (see Section 5.2). Furthermore, it must be noted that tablature only provides a 'minimum' duration for each note, applying to all notes in a chord (see Figure 1). We have made no attempt at reconstructing the notes' actual duration, i.e., at determining their offset—which is not a trivial problem as it depends on multiple contextual factors (e.g., counterpoint, phrasing, etc.). At this point, it is therefore important to stress that the transcriptions created are not full-fledged editions in the musicological sense of the word, and are not intended as such—rather, they serve to facilitate large-scale content-based computational analysis.

3.1 Preprocessing: Alignment issues

The mapping algorithm takes as input an encoding of the tablature in TabCode format and a MIDI rendering of the vocal model. In order for it to function optimally, these files must be aligned both in terms of pitch (i.e., the appropriate lute tuning must be used) and time (i.e., they must span the same number of bars). [9] In the ideal case, only the tuning needs to be provided—but in practice, most of the tablature encodings require a number of small corrections, and in some cases even additions, all of them necessary to ensure temporal alignment. Currently, these need to be carried out in a manual preprocessing step.

There are four types of *corrections*, three of which apply to the encodings. First, almost all encodings require the mensuration signs used to be adapted in order for the nominal beat level to correspond to that in the MIDI file (in most cases this results in augmentation—e.g., a change of a $\frac{2}{2}$ mensuration to a $\frac{2}{1}$ mensuration—or, conversely, a reduction; but in some cases they are actually incorrect); moreover, a small number of encodings require lacking mensuration signs to be added. Second, a small number of encodings require unspecified triplets to be made specific. Third, a small number of encodings require the correction of syntactic errors that render them non-parsable (technically, this type of correction does not affect any alignment issues). [10] The fourth and last type of correction applies to the MIDI renderings of the vocal models, which all require the key signature meta message to be set.

Furthermore, there is one type of *addition*, which again applies to the encodings only, and which results in lengthening them. A substantial number of encodings—29 in

[4] See `http://www.gerbode.net/`. The files were converted from the proprietary Fronimo format into TabCode.

[5] `https://github.com/josquin-research-project/`

[6] `http://josquin.stanford.edu/`

[7] See `https://www.cmme.org/database/projects/14/` for a project born because of this exact reason.

[8] `https://github.com/reinierdevalk/data/`

[9] The standard tuning used for the six-course sixteenth-century lute is one where the courses are tuned in perfect fourths with a major third in the middle. The lowest-sounding course was generally tuned to nominal G or A, but several other tunings occured as well.

[10] We make no attempt at correcting any incorrectly encoded notes.

Vocal model	Voices	I	Vocal model	Voices	I
Absalon fili mi	4	1	*Missa Hercules Dux Ferrarie*,	4, 2, 4	1, 1, 1
Benedicta es, pt. 1–3	6, 2, 6	4, 3, 3	Sanctus–Pleni sunt–Osanna		
Fecit potentiam	2	1	*Missa Pange lingua*, Benedictus	2	1
In exitu Israel de Egypto, pt. 1–3	4	1, 1, 1	*Missa Sine nomine*, Cum sancto	4	1
Inviolata, pt. 1–3	5	1, 1, 1	*Qui belles amours*	4	2
Memor esto verbi tui, pt. 1–2	4	1, 1	*Adieu mes amours*	4	3
Pater noster, pt. 1–2	6	2, 2	*Comment peult avoir joye*	4	1
Preter rerum seriem, pt. 1–2	6	3, 2	*Faulte d'argent*	5	1
Qui habitat in adjutorio, pt. 1–2	4	2, 2	*Je ne me puis tenir*	5	3
Stabat mater, pt. 1–2	5	2, 1	*La plus des plus*	3	1
Missa De beata virgine, Cum	4, 5, 5	2, 1, 1	*Mille regretz*	4	2
sancto–Credo–Crucifixus			*Plus nulz regretz*	4	1
Missa Faysant regretz, Qui tollis–	4, 4, 3, 4	1, 1, 1, 1	*Si j'ay perdu*	4	1
Sanctus–Pleni sunt–Osanna					

Table 1. JOSQUINTAB: 36 motet part intabulations (ranging in size from 234–2238 notes; median 1257.5 notes; total 43557 notes); 13 mass section part intabulations (260–1141 notes; median 343 notes; total 5942 notes); 15 chanson intabulations (483–1250 notes; median 686 notes; total 11159 notes). I = number of intabulations.

total—contain fewer bars than their vocal models. In most cases, this is obviously due to an error on behalf of the intabulator, i.e., accidental omission. Given a sequence of bars $\{b_1, ..., b_{n-1}, b_n\}$, where bar b_1 has the same ending as bar b_{n-1}, all bars between b_1 and b_n may easily be skipped unwittingly (a scribal error known as *homoeoteleuton* in manuscript studies [9]). In exceptional cases, however, omission is intentional. Omissions are countered by inserting rests into the intabulation at the appropriate position.

Apart from an encoding and a MIDI rendering, the mapping algorithm must also be given the appropriate tuning, as well as the appropriate *reduction*: the factor by which the durations in the tablature must be multiplied (or divided) in order for its nominal beat level to correspond to that of the vocal model. The reduction can be a positive integer greater than 1, in which case it amounts to an augmentation; a negative integer, in which case it amounts to a reduction; or 1, in which case the durations do not change. Although determining the tuning and reduction is strictly speaking not a preprocessing step, this information must be given as input to the mapping algorithm, and must thus be established—currently, manually—prior to running it.

4. MAPPING ALGORITHM

The mapping algorithm itself is fairly straightforward and traverses the music chord by chord (where a chord is any event that has one or more notes in it), from left to right. Given a tablature encoding and a MIDI representation of the vocal model, first, an ordered list T is made, containing the union of the chord onset times in both tablature and MIDI. Then, two two-dimensional matrices with $|T|$ rows are constructed: the *grid* G, containing, for each onset time T_i, onset, pitch, voice, and duration information for the chord at T_i in the MIDI (or null values when there is no chord); and the *mask* M, containing, for each onset time T_i, onset, pitch, duration and note index information for the chord at T_i in the tablature (or null values when there is no chord). The grid and mask are traversed jointly, row by row, and in the process a list of *voice labels* V is created. A voice label is a binary vector encoding all voices onto which a note is mapped. Let mask row $M_{i,*}$ represent a tablature chord and grid row $G_{i,*}$ a MIDI chord at onset T_i. If $M_{i,*}$ is non-null, i.e., there is a chord in the tablature at T_i, the following steps are taken.

1. If $M_{i,*}$ contains a single note whose duration is less or equal than the *ornamentation threshold* (a pre-set parameter) and $G_{i,*}$ is null, [11] i.e., there is no chord in the MIDI at T_i (see Figure 2, horizontal brackets), the note is flagged as ornamental. Null is added to V, the note is added to a running list O, and Steps 2-4 below are skipped.
2. Each note in $M_{i,*}$ is mapped to a voice by finding the voice that goes with its counterpart in $G_{i,*}$. A note can be mapped to multiple voices.
   (a) If a match is found, a voice label is added to V.
   (b) If no match is found (see Figure 2, single brackets and non-bracketed sharps), null is added to V, and the note is added to U, the list of unmapped notes in $M_{i,*}$.
3. If U is not empty, the notes in it are mapped—in one go—to the available voices (any voices not yet occupied by a note in $M_{i,*}$). To this end, for each available voice in the MIDI, the last note with an onset that is less than T_i in that voice is added as a pitch-voice tuple to a list A. Then, the cheapest mapping of U onto A is calculated, where the cost is based on pitch distance. All corresponding null voice labels in V are replaced, and all mapped notes removed from U. (If, after this process, there are still unmapped notes—which can happen if $M_{i,*}$ contains more notes than there are voices in the model—this step is repeated, now considering *all* voices as avail-

[11] As ornamentation threshold we use the value two levels below beat level (e.g., in $\frac{2}{1}$ or $\frac{3}{1}$ mensuration, a quarter note).

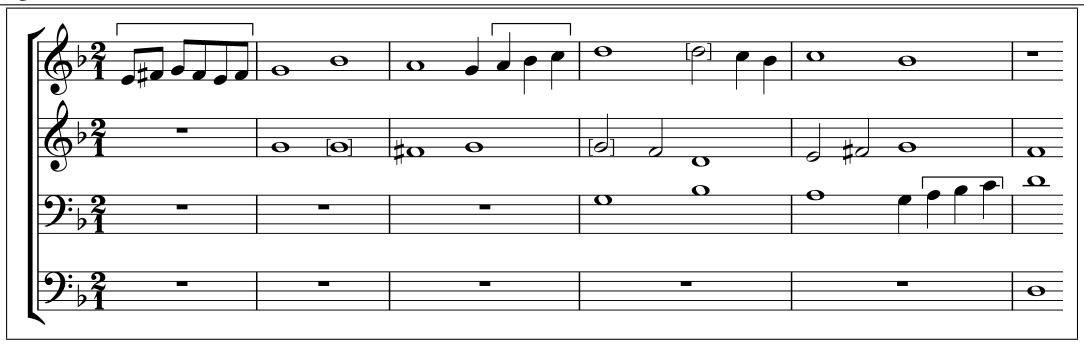


Figure 2. Francesco Spinacino, *Adiu mes amours*, opening bars (see Figure 1). Mapping algorithm output. The tablature contains all notes from the vocal model; bracketed notes and sharps are additions. Full note durations are inferred.

able. As a result, voices can have more than one note mapped onto them.)

4. (Optional.) If $M_{i-1,*}$ is ornamental, the cheapest connection of the last note in O to $M_{i,*}$ is calculated (where the cost is again based on pitch distance), and all notes in O are mapped onto the voice onto which the note in $M_{i,*}$ that yields the cheapest connection has been mapped. In case of a tie, the cheapest connection of the first note in O to $M_{i-1-|O|,*}$ (i.e., the chord immediately before that first note) is decisive. All corresponding null voice labels in V are replaced, and O is cleared.

When the grid and mask have been traversed, the algorithm combines the created list of voice labels V with an internal representation of the tablature created from the encoding, and, retaining the key and mensuration information from the MIDI, returns this as a track-separated MIDI file and an MEI rendering thereof. If Step 4 is included, these contain all ornamentation that is in the tablature; if it is skipped, they remain unornamented.

4.1 Evaluation method

There are four categories of mismatches. If in Step 1 a note is flagged, it is considered an *ornamentation*. Else, if in Step 2(b) no match is found for a note, and in Step 3 it is mapped at cost 0), it is considered a *repetition* (see Figure 2, single brackets); else, if there is a semitone discrepancy between it and a note in A, but they are the same pitch class, it is considered *musica ficta* (see Figure 2, non-bracketed sharps); else, it is considered an *alteration*. These categories are hierarchical and mutually exclusive.

This classification enables us to quantitatively evaluate the degree of arrangement (goodness of fit) of an intabulation in a granular manner. We measure the degree of arrangement by means of the *mapping ratio* $m = \frac{|N|-|M|}{|N|}$, where N is the set of all notes, and M the set of all mismatches. At the highest level of granularity, which results in the lowest m, M is equal to the total of all ornamentations, repetitions, instances of musica ficta, and alterations—i.e., to the total of mismatches across all four categories. It can be argued, however, that not all categories carry the same weight. Repetitions, for example, are idiomatic for lute music because of the instrument's short sustain, and have only little harmonic and melodic impact; and instances of musica ficta, which have a stronger harmonic and melodic impact (but no rhythmic impact), are nothing but inflections, different 'flavours' of the same pitch class. Thus, it makes sense not to include mismatches falling into these categories in M, or at least to weight them differently. We therefore redefine m as

$$m = \frac{|N| - (o|M_o| + r|M_r| + f|M_f| + |M_a|)}{|N|}, \quad (1)$$

where M_o, M_r, and M_f are the sets of all ornamentations, repetitions, and instances of musica ficta; o, r, and f their respective weighting parameters; and M_a is the set of all alterations (N, as above, is the set of all notes).

Genre	Ornamented			Unornamented	
	o, r, f			r, f	
	$1, 1, 1$	$1, 0, 0$	$0, 0, 0$	$1, 1$	$0, 0$
Motets	0.52	0.66	0.92	0.70	0.89
Mass sections	0.70	0.83	0.96	0.80	0.95
Chansons	0.57	0.69	0.94	0.75	0.92
All	0.54	0.68	0.93	0.72	0.90

Table 2. Mapping ratio m at different levels of granularity. Values are weighted averages, where each per-piece ratio is weighted by the number of notes in the piece.

Table 2 shows the mapping ratio, averaged per genre, at the highest level of granularity (i.e., the most strict case, with all parameters set to 1), at an intermediate level (with only o set to 1), and at the lowest level (i.e., the most lenient case, with all parameters set to 0) for the ornamented transcriptions, as well as at the highest and lowest level for the unornamented transcriptions, where the intermediate level does not apply. The table shows that, on average, at least half of the notes ($m = 0.54$) in the ornamented transcriptions, and approximately three quarters ($m = 0.72$) in the unornamented transcriptions, have an exact match in the vocal models. The ratios increase considerably—to 0.68 and 0.90, respectively— if we consider repetitions and musica ficta to be matches as well. The table also shows that, generally, the intabulations are fairly heavily ornamented, as the ratio at the lowest level of granularity ($m = 0.93$) is much higher than that at the other levels.

4.2 Command line tool

The Java implementation of the mapping algorithm, `TabMapper`, [12] can be run as a command line tool. A single transcription is created with the command

```
$ tabmapper tab.tc model.mid <t> <r> -n
```

where `tab.tc` is the TabCode encoding of the intabulation, `model.mid` the MIDI rendering of the vocal model, `<t>` and `<r>` are the tuning and reduction (see Section 3.1), and `-n` is an optional argument indicating that the transcription created should be unornamented. A *set* of transcriptions is created with the command

```
$ tabmapper list.csv -n
```

where `list.csv` is a CSV file specifying, for each piece, the name of the TabCode file, that of the MIDI file, the tuning, and the reduction, and `-n` is again optional.

5. USE CASES

Two use cases demonstrate the usefulness of the dataset for both MIR and musicological research.

5.1 Use Case 1: Voice separation

Use Case 1 shows how the dataset can serve to train machine learning models for voice separation in lute tablature [13, 14], which can be used for automatic polyphonic transcription of music in lute tablature. *Voice separation* can be defined as "the task of separating a musical work consisting of multi-note sonorities into independent constituent voices" [7]. From an MIR perspective, this task is considered an open problem. As with many supervised machine learning tasks in MIR, a lack of labelled data to train and evaluate models on hinders progress. This is a particularly pressing issue in the case of tasks related to music in lute tablature. Apart from our own manually created dataset containing 15 pieces (11641 data points) [13, 14], to our knowledge there currently exist no other datasets of voice-annotated tablature.

Automatic data creation can be a solution. However, data created by means of a rigid algorithm is less consistent in terms of transcription quality than data created manually, where each individual situation can be addressed on its own terms—and this may result in a model having more difficulty learning the task, and consequently, generalising worse. An experiment confirms this. In [13], we train and evaluate neural networks on nine four-voice pieces (8892 data points); our best-performing model achieves a generalisation accuracy of 79.63%. A model with the same architecture [13] trained and evaluated on the 28 four-voice pieces (26215 data points) in JOSQUINTAB yields a generalisation accuracy of only 64.47%. [14] We hypothesise this to be partly due to inconsistencies in the data; however, an analysis of the results reveals that an error propagation issue particular to imitative music, as discussed in [15], also plays a substantial role. As the dataset grows and inconsistencies are ironed out statistically, performance improvement is expected. Despite its current subpar performance, this model can already be reliably used for automatic polyphonic transcription.

5.2 Use Case 2: Cross-corpus melodic matching

Use Case 2 shows how the dataset can be applied for melodic matching across heterogeneous—in this case, instrumental versus vocal—corpora. A major goal of automatic transcription of tablature is the ability to search freely for *musical quotations* occurring between corpora in tablature and in standard notation. Our dataset provides an important stepping stone in this direction. Using the unornamented transcriptions created, we are able to do a further processing step in order to create search strings for Early Music Online Search, [15] a tool that uses OMR to enable full-text searches into sixteenth-century printed music.

Our initial results are promising. Eight pieces exist in both corpora, of which we are able to successfully search and match four. Not only are we able to match the individual pieces, but also the individual voice parts within each piece are correctly identified. In three additional cases, our search matches not with the correct piece, but rather with another piece by the same composer—showing that our search method enables the identification of common stylistic features. This shows the potential of transcribing a larger tablature corpus in order to identify hitherto unknown vocal models.

6. CONCLUSIONS AND FUTURE WORK

In this paper we present JOSQUINTAB, a dataset of automatically created enriched diplomatic transcriptions of 64 lute intabulations. We describe the contents of the dataset, the mapping algorithm used to create it, and we show how we can quantitatively evaluate it. Two use cases illustrate how the dataset can be used for content-based computational analysis within and across corpora, demonstrating its usefulness for MIR and musicological research. We make the dataset, the source code, and an implementation of the mapping algorithm publicly available.

There are many ways in which this work can be extended. One is to further automate the preprocessing (alignment, determination of tuning and reduction) that the data currently requires. Another is to parameterise certain functionality of the mapping algorithm (e.g., the value of the ornamentation threshold) in order to allow more flexibility in the transcription creation. A third is to extend the dataset so that it includes all intabulations of Josquin compositions, and those of other composers as well. It is clear that the work presented opens many new research avenues, which we plan to explore in future work.

[12] `https://github.com/reinierdevalk/tabmapper/`

[13] We now use the backward processing mode [13], and train using a validation set, doubling the number of training iterations and decreasing the amount of regularisation applied by a factor of 10.

[14] Values are cross-validation averages, where the dataset is partitioned along its individual pieces.

[15] `http://www.doc.gold.ac.uk/usr/265/`

7. REFERENCES

[1] W. Apel. *The notation of polyphonic music: 900-1600*. The Mediaeval Academy of America, Cambridge, MA, 5th edition, 1953.

[2] C. M. S. Bocchinfuso. Intabulations of music by Josquin des Prez in lute books published by Pierre Phalèse, 1547–1574, volumes 1–2. Master's thesis, University of Canterbury, 2009.

[3] H. M. Brown. *Instrumental music printed before 1600: A bibliography*. Harvard University Press, Cambridge, MA, 1965.

[4] H. M. Brown. Embellishment in early sixteenth-century Italian intabulations. *Proceedings of the Royal Musical Association*, 100:49–83, 1973–1974.

[5] H. M. Brown. Accidentals and ornamentation in sixteenth-century intabulations of Josquin's motets. In E. E. Lowinsky and B. J. Blackburn, editors, *Josquin des Prez: Proceedings of the International Josquin Festival-Conference*, pages 475–522. Oxford University Press, London, 1976.

[6] H. M Brown. The importance of sixteenth-century intabulations. In L. P. Grijp and W. Mook, editors, *Proceedings of the International Lute Symposium Utrecht 1986*, pages 1–29. STIMU Foundation for Historical Performance Practice, Utrecht, 1988.

[7] E. Cambouropoulos. Voice and stream: Perceptual and computational modeling of voice separation. *Music Perception*, 26(1):75–94, 2008.

[8] H. Charnassé and B. Stepien. Automatic transcription of german lute tablatures: An artificial intelligence application. In A. Marsden and A. Pople, editors, *Computer representations and models in music*, pages 143–170. Academic Press, London, 1992.

[9] R. Clemens and T. Graham. *Introduction to manuscript studies*. Cornell University Press, New York, 2007.

[10] T. Crawford. TabCode for lute repertories. *Computing in Musicology*, 7:57–59, 1991.

[11] T. Crawford, B. Fields, D. Lewis, and K. Page. Explorations in Linked Data practice for early music corpora. In *Proc. of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries, London, UK*, pages 309–312, 2014.

[12] T. Dart, J. Morehen, and R. Rastall. Tablature. In S. Sadie, editor, *The new Grove dictionary of music and musicians*, volume 24, pages 905–914. Macmillan, London, 2nd edition, 2001.

[13] R. de Valk. *Structuring lute tablature and MIDI data: Machine learning models for voice separation in symbolic music representations.* PhD thesis, City University, London, 2015.

[14] R. de Valk and T. Weyde. Bringing 'Musicque into the tableture': Machine-learning models for polyphonic transcription of 16th-century lute tablature. *Early Music*, 43(4):563–576, 2015.

[15] R. de Valk and T. Weyde. Deep neural networks with voice entry estimation heuristics for voice separation in symbolic music representations. In *Proc. of the 19th International Society for Music Information Retrieval Conference, Paris, France*, pages 281–288, 2018.

[16] W. Elders and E. Jas, editors. *Josquin des Prez: The sources*. The Collected Works of Josquin des Prez, 1. Koninklijke Vereniging voor Nederlandse Muziekgeschiedenis, Utrecht, 2013.

[17] J. Grier. *The critical editing of music: History, method, and practice*. Cambridge University Press, Cambridge, 1996.

[18] J. Griffiths. The lute and the polyphonist. *Studi Musicali*, 31(1):89–108, 2002.

[19] J. Griffiths. Songs without words: The motet as solo instrumental music after Trent. In E. Rodríguez-García and D. V. Filippi, editors, *Mapping the motet in the post-Tridentine era*, pages 206–227. Routledge, Abingdon, 2019.

[20] M. Kanno. Prescriptive notation: Limits and challenges. *Contemporary Music Review*, 26(2):231–254, 2007.

[21] H. Y. Kwee. Sixteenth-century printed instrumental arrangements of works by Josquin des Prez: An inventory. *Tijdschrift van de Vereniging voor Nederlandse Muziekgeschiedenis*, 22(1):43–66, 1971.

[22] D. Lewis, T. Crawford, and D. Müllensiefen. Instrumental idiom in the 16th century: Embellishment patterns in arrangements of vocal music. In *Proc. of the 17th International Society for Music Information Retrieval Conference, New York, NY, USA*, pages 524–530, 2016.

[23] R. J. Lewis, T. Crawford, and D. Lewis. Exploring information retrieval, semantic technologies and workflows for music scholarship: The Transforming Musicology project. *Early Music*, 43(4):635–647, 2015.

[24] H. Minamino. *Sixteenth-century lute treatises with emphasis on process and techniques of intabulation*. PhD thesis, University of Chicago, 1988.

[25] W. H. Mook. De bewerkingen voor luit en vihuela da mano van het motet *Benedicta es caelorum regina* van Josquin des Prez, volumes 1–2. Master's thesis, Rijksuniversiteit Utrecht, 1993.

[26] A. Moths. Mit oder ohne Text? Was Diminutionen und Intabulierungen zur Analyse vokaler Musik des 16. und 17. Jahrhunderts beitragen können. *Dutch Journal of Music Theory*, 16(1):39–46, 2011.

[27] A. J. Ness and C. A. Kolczynski. Sources of lute music. In S. Sadie, editor, *The new Grove dictionary of music and musicians*, volume 24, pages 39–63. Macmillan, London, 2nd edition, 2001.

[28] C. Rhodes and D. Lewis. An editor for lute tablature. In R. Kronland-Martinet, T. Voinier, and S. Ystad, editors, *Computer Music Modeling and Retrieval: Third international symposium, CMMR 2005*, pages 259–264. Springer, Berlin, 2006.

[29] C. Seeger. Prescriptive and descriptive music-writing. *The Musical Quarterly*, 44(2):184–195, 1958.

[30] G. Thibault. Instrumental transcriptions of Josquin's French chansons. In E. E. Lowinsky and B. J. Blackburn, editors, *Josquin des Prez: Proceedings of the International Josquin Festival-Conference*, pages 455–474. Oxford University Press, London, 1976.

[31] R. C. Wegman. The other Josquin. *Tijdschrift van de Koninklijke Vereniging voor Nederlandse Muziekgeschiedenis*, 58(1):33–68, 2008.

[32] M. D. Wilkinson et al. A design framework and exemplar metrics for FAIRness. *Scientific Data*, 5(180118), 2018.

A DATASET OF RHYTHMIC PATTERN REPRODUCTIONS AND BASELINE AUTOMATIC ASSESSMENT SYSTEM

Felipe Falcão[1,2] **Baris Bozkurt**[2,3] **Xavier Serra**[2]
Nazareno Andrade[1] **Ozan Baysal**[4]
[1] Universidade Federal de Campina Grande, Brazil
[2] Music Technology Group, Universitat Pompeu Fabra, Barcelona
[3] Izmir Demokrasi University, Turkey
[4] Istanbul Technical University, Turkey

ABSTRACT

This work presents a novel dataset comprised of audio and jury evaluations for rhythmic pattern reproduction performances by students applying for a conservatory. Data was collected *in-loco* during entrance exams where students were asked to imitate a set of rhythmic patterns played by teachers. In addition to the *pass* or *fail* grades provided by the members of the jury during the exam sessions, a subset of the data was also evaluated by external annotators on a 4-level scale. A baseline automatic assessment system is presented to demonstrate the usefulness of the dataset. Preliminary results deliver an accuracy of 76% for a simple pass/fail logistic regression classifier and a mean average error of 0.59 for a linear regression grade estimator. The implementation is also made publicly available to serve as baseline for alternative assessments systems that may leverage the dataset.

1. INTRODUCTION

Automatic assessment of music performances is an important audio signal processing application drawing increased attention over the past few years. The *Massive Open Online Course* (MOOC) methodology has recently contributed to the growth of online music courses, attracting a large number of students. In this scenario, automatic assessment methodologies have the potential to largely reduce the instructor load of assessing student submissions. Moreover, due to its subjective nature, the task of evaluating students performances can be very difficult [4, 14, 24], sometimes even preventing different evaluators from reaching an agreement while assessing the same performance [17, 22]. Automatic evaluations may circumvent this obstacle by defining clear and objective goals that must be achieved in order to succeed in a musical performance.

While some authors leverage their evaluation tools using linear measures to quantify similarity between pairs of reference and performance [20], most of the recently proposed assessment systems rely on machine learning based models for this task. Earlier models were trained with labeled data and hand-crafted audio features targeting at the prediction of grades for performances. Such methodology is applied in Nakano et al. [13] for improving the state-of-art for singing voice assessment. The authors gathered data from the AIST-HDB dataset [9] to train a model able to predict *good/poor* classifications for singing performances. Bozkurt et al. [4] have also suggested a supervised learning method for singing voice assessment, but this time conducting their own data collection procedures inside a music conservatory. The collected data containing jury evaluations was fed into a machine learning model whose accuracy was reported as 74% for binary *pass/fail* predictions. Singing assessments following similar methodologies and including classification systems were previously proposed by Schram et al. [16] and Molina et al. [12]. Both authors aimed at the automatic evaluation of voice performances by training machine learning models with audio features and targeted scores.

Recent advances in unsupervised learning led researchers to also rely on learned metrics to support their assessment systems. Unlike the aforementioned supervised procedures, these techniques delegate to the model itself the task of figuring data patterns directly from raw audio data, clustering similar observations into equivalent groups and using such information to predict assessments. Examples of such methodology include the results discussed by Wu and Lerch in [24], where authors modeled a feature learning approach specially designed for assessing percussive performances recorded during band auditions. This very same data source leveraged Pati et al. [14] in their similar study also tackling the problem of modeling music assessment by means of learned features. They have proposed the application of deep neural networks capable of capturing non-linear aspects of performances that would better correlate with reference's features. These recent studies encourage the use of unsupervised feature learning rather than supervised methods, reporting that the former

outperforms the latter in most of the cases.

Audio corpora play a critical role in every assessment system, as the decision functions used for predicting evaluations are trained on the corpora. In their work, Li et al [10] delivered an important contribution by reviewing several commonly-used music datasets made publicly available for MIR tasks. This systematic review also describes important information regarding the nature of each dataset (e.g. available content, total audio durations, types of annotations) and points out to a lack of datasets with annotations related to rhythmic assessment, we argue. Specifically regarding the rhythmic dimension, the dataset provided by the Florida Bandmasters Association (FBA) has been commonly adopted [14,21,23,24] and is, to the extent of our knowledge, the only dataset that currently includes rhythmic performances that are annotated with grades. It is comprised of audio performances from band auditions recorded in the Florida state between 2013 and 2015, including jury assessments for different music aspects (musicality, note accuracy, rhythmic accuracy, tone quality, etc.).

This present work is an effort to address the shortage of music datasets designed for rhythmic assessment. The presented data was collected during the rhythmic sessions of entrance exams based in a music conservatory, where student performances were recorded and evaluated by members of a jury. Data curation procedures were applied over this raw dataset (including 1040 student performance recordings) in order to extract a subset of (80) performances which were submitted to an extra evaluation, this time in a 4-level scale (i.e. grades ranging from 1 to 4). The resulting subset featured with annotated data was fed to a simple machine learning rhythmic assessment system in order to demonstrate the use of the dataset in this scenario. A binary *pass/fail* logistic classifier and a linear regression grade estimator are implemented, the former delivering a maximum accuracy of 76% while the latter pointing to a minimum mean average error of 0.59 when tested over a 5-fold cross validation. Both the complete rhythmic dataset and the re-annotated subset are made publicly available. The implementation of the proposed rhythmic assessment system is also openly shared to serve as baseline for comparisons with similar approaches.

2. THE MAST RHYTMIC DATASET

The Musical Aptitude Standard Test (MAST) Rhythmic Dataset is a collection of rhythmic performances and references recorded in the Istanbul Technical University (ITU) Turkish Music Conservatory during entrance exams. In Turkey, these assessments are commonly applied to support the evaluation of the musical aptitude of applicants, determining whether or not they should be accepted to study in the institution. Categorizations are preferably achieved during jury-assisted exams when students are individually auditioned and evaluated according to multiple musical aspects (e.g. chord recognition, melodic singing, rhythm playing). For the rhythmic session students are asked to imitate reference performances by usually clapping hands or tapping a hard surface.

The rhythmic patterns included in the dataset are taken from the jury based qualification exams of the years 2015 and 2016. The rhythmic assessment portion of the entrance exams in these years was composed of two types of rhythmic pattern reproduction questions; rhythm one in a simple meter (4/4) and rhythm two in a compound meter (7/8, 9/8, 10/8 or a 5/4). In order to ensure the confidentiality of the questions asked in the exam and minimize the chance of a memorization and the leakage of these pattern outside of the examination areas, the applicants chose randomly from 10 different question packages, each package having a different version of the two types of rhythmic patterns stated. Thus, there are 20 different rhythms for each year making up the total of 40 distinct rhythmic patterns in our dataset.

Besides taking into consideration the two types of rhythms, the exam preparation committee designed the questions such that each pattern should be employing similar rhythmic values (quarter-note, eight-note, sixteenth-note and a triplet) and similar number of notes. The applicants were expected to perform above a threshold of success regardless of the package the selected. Later, it was observed that, while there may be differences in terms of difficulty for different packages which might affect an applicant's test score, there wasn't any significant relationship observed between the selected packages and the success of the candidates [1].

The jury committee consists of three members and the candidates are expected to reproduce the rhythmic pattern after it has been played two times by a member of the committee. The jury gives a full grade (10pts) and moves on to the next question if the candidate's performance is an exact reproduction (or nearly). If there are flaws in the reproduction, then the candidate is exercised by performing the rhythmic pattern divided to two halves separately, after which the rhythmic pattern is played for reproduction one last time. The evaluation at this stage may have three outcomes, either the participant receives a partial grade (8pts) if it is an exact reproduction (since he/she couldn't perform an exact reproduction at the beginning), if it still has 1-2 errors the jury gives a minor grade (4 pts), and no points are given if the performance has more than 2 errors. The evaluations of the jury member can show variances at this stage due to individual preferences (e.g. for some a consistent tactus may be more important than missing an attack, for some it is the accentuation and the phrasing). Due to these, in our dataset we have only selected those performances in which there was an unanimously consensus among jury members that it was an exact (or nearly exact) reproduction or a failure (all giving 0 pts.).

In general, to allow reinspection of the execution of these exams, each applicants performance is video recorded by the ITU conservatory directorate. For our purposes – and to ensure anonymity - these recordings were converted to wave files and then cropped so that each recording consisted of the candidates performances only.

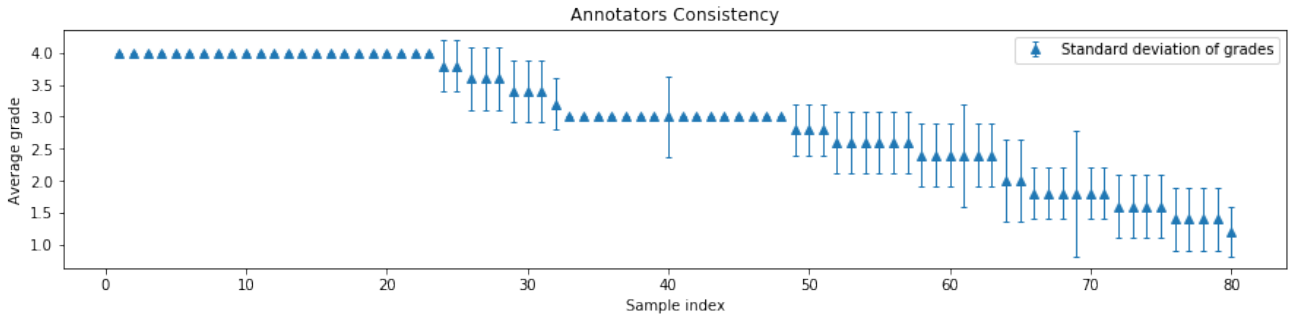


Figure 1. Distribution of grades assigned by the distinct annotators. The x-values indicate the performance index while the y-values stand for the average grade assigned. Error bars describe the standard deviation for all evaluations

3. USER ANNOTATIONS

In order to contribute to a more complete validation over the collected data, the original dataset - which originally only contained *pass/fail* classifications provided by members of a jury - was also annotated with a higher resolution (4-level) grading. Since the re-assessment of all the 1040 student performances comprised by the full dataset would require high human resources, the original data was sampled. This convenience sampling initially filtered 20 references with low rhythmic complexity, followed by the selection of four noise-free student performances for each reference - two from each *pass/fail* class, totaling 80 sampled student performances and 20 jury member performances (that serve as the reference/target rhythmic pattern for grading a student performance via comparison).

This subset of performances (from now on addressed as re-annotated subset) was submitted to evaluation sessions completed by seven annotators (male: six, female: one) and aided by a custom evaluation tool (depicted in Figure 2) developed for a similar data collection task. During these sessions, the annotators could hear in sequence the rhythmic reference and student performance (with a one second silence in-between) as many time as desired until feeling comfortable to choose between one of the available grades: *1 - Completely off, 2 - Major errors, 3 - Minor errors, 4 - Perfect*. Although no advanced music skills were considered mandatory to support the assessment of quite simple rhythmic patterns, authors tried to select annotators with some relevant music background. Besides, the graders were provided with a custom rubric documenting the musical aspects that should be taken into consideration when assigning grades. They were asked to assess the similarity of the student performance to the reference in terms of the beat and duration patterns, discarding tempo differences.

Evaluation sessions could be interrupted and resumed by annotators at any moment using the session control feature provided by the tool. Figure 1 presents the distribution of averaged grades assigned by annotators to all the sampled performances. Nearly half (38) of evaluated performances had unanimous assessments while the rest of them presented some deviations (mean: 0.49, max: 0.97).

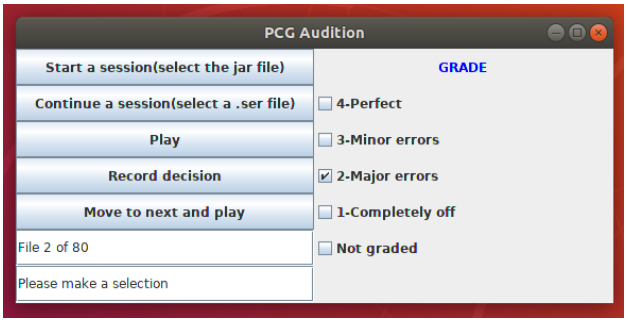


Figure 2. Annotation tool used during the custom evaluation sessions

4. DATA PREPARATION

The detection of rhythmic events is an issue recurrently addressed via extraction of onset times from raw audio [6–8]. Onset features are primarily encoded as onset vectors containing the moments when signal-disturbing events (e.g. chord attack, drum kick) happen. Multiple similarity measures have already been proposed for the comparison of onset vectors, including distance between vectors [18] and error measures [15]. In this present work we propose a hybrid model that benefits from both aforementioned similarity measures to predict rhythmic assessments.

For the onset detection, our audio processing module relies on the *OnsetDetection* algorithm implemented by the Essentia library [3] [1]. All references and performances comprised by the re-annotated subset were sampled at a rate of 44.1kHz and the resulting frames were provided to the onset detection algorithm to allow for feature extraction. The onset extractor is parameterized with default values (window size: 1024 samples, hop size: 512 samples) and three methods for onset extraction are tested and compared: High Frequency Content detection (HFC) [11], Spectral Flux detection (FLUX) [19] and Complex-Domain spectral difference (COMPLEX) [2].

The recordings in the dataset are not aligned in time, nor cropped with a fixed offset before/after the first/last onset. Hence the first and the last onset are considered as boundaries of each performance. For the use of vector distance measures applied to same-sized vectors, binary vectors are

[1] https://essentia.upf.edu

computed applying a fixed-numbered (60) grid on the time axis (i.e. for each recording, the duration between the first and last onset is divided into 60 time bins and a binary value (onset/non-onset) is stored in the vector for each bin). We opted for 60 bins for each rhythmic performance, since this number is divisible by two, three, four, five, and six, which are the common multipliers for most rhythmic patterns.

Figure 3 presents an example of a visual representation for onsets times before and after the quantizing procedures that we have just described. The relative positions for the original onsets are plotted in dashed lines and circle stems, while the quantized information is drawn in solid lines and triangle stems. All the quantized, unquantized onsets and scaling information are also included into the re-annotated dataset for further use.

5. BASELINE ASSESSMENT SYSTEM

This work introduces a baseline automatic assessment system for rhythmic performances of students. Regression models are trained with vector similarities aiming at the modeling of the rhythmic evaluations detailed in the re-annotated subset.

The feature set for both models described below includes eight different similarity measures: two strictly related to the rhythmic domain (beat difference and Percival's similarity [15]), four text-distance approaches (Levenshtein, Damerou-Levenshtein, Jaro and Jaro-Winkler) and two vector-distance measures (Hamming and Yule). This last subset of features was selected according to their reported benefits in comparing vector with boolean data [5]. Since the nature of these measures resulted in different distance ranges, all features were normalized in order to range within a common scale.

Two different types of evaluations are targeted by the proposed assessment systems. A grade estimation is implemented through a linear regression model while categorizations between *pass/fail* classes are predicted by a logistic regression classifier. The overall 'true' grade of a performance is calculated via removing the highest and lowest grades and averaging the rest, which is an approach similar to the one usually applied in music conservatories. Binary categorization is modeled by judging as accepted (*pass*) all performances whose average grades are greater than or equal to three, rejected (*fail*) otherwise.

All machine learning implementations are written in Python 3.6.7 using scikit-learn [2] modules.

6. RESULTS

Both mentioned datasets are now made publicly available for further investigation. The complete rhytmic dataset (MAST rhythm dataset) [3] is a collection of 3721 audio files cropped from recordings of conservatory entrance examinations in Turkey (summer 2015 and summer 2016). 1040 of the recordings are student performances graded by a jury of 3 instructors as pass or fail. The rest of the recordings are jury performances of the same rhythmic patterns in various sessions. The re-annotated subset (MAST rhythm re-annotated subset) [4] is a balanced sample (in terms of pass-fail graded samples) extracted from the complete dataset, assessed by seven annotators in a 4-level grid and onset information for 80 performances, accounting for 20 distinct rhythmic patterns (references). All the code supporting the implementation of our automatic assessment systems is also shared as Jupyter notebooks at Github [5].

Classifier	Ac.	Pr.	Rc.
Logistic Regression (FLUX)	63%	66%	83%
Logistic Regression (COMPLEX)	72%	75%	83%
Logistic Regression (HFC)	**76%**	**79%**	**78%**

Table 1. Performance measures (accuracy, precision and recall) for classifiers trained with different onset features

The designed models are evaluated according to how well they predict assessment for unseen test data. Our evaluation relied on a 5-fold cross validation (test size: 20%) for both models. The performance results for the logistic classifier are summarized in the learning curves shown in Figure 4 and states a maximum accuracy of 76% when trained with HFC data. The complete comparison stating accuracy, precision and recall results from various onset features can also be examined in Table 1.

For the linear grade estimator, the performance analysis consisted of measuring the errors observed between the predicted grades and the expected values. Our estimator was compared with baseline naive versions whose projected evaluations are modeled using uniform and random distributions. Results are summarized in Table 3 and indicate that our approach provides better predictions (yet with a small margin) than the naive estimators regardless of the trained feature, with HFC once again delivering the best performance. As for the influence of specific features over the decision function, the coefficients suggested by the linear regression (Table 2) encourage us to infer that Jaro, Jaro-Winkler and Hamming distances are the features whose influences are higher over grading predictions.

Our final evaluation is carried out by crossing data from the two proposed datasets. Models trained with sampled data from the re-annotated dataset were provided with all performances coming from the complete dataset in order to verify how well the trained assessment systems would behave when asked to predict evaluations for new unseen data. All the 1040 performances comprised by the full dataset were submitted to the same pre-processing steps described in Section 4 and the resulting similarity measures were tested against both the logistic classifier and linear estimator trained with HFC features (since it's the extraction method that delivers the best performance). Here accuracy is calculated according to the number of predic-

[2] https://scikit-learn.org
[3] https://zenodo.org/record/2620357
[4] https://zenodo.org/record/2619499
[5] https://github.com/MTG/mast-rhythm-analysis

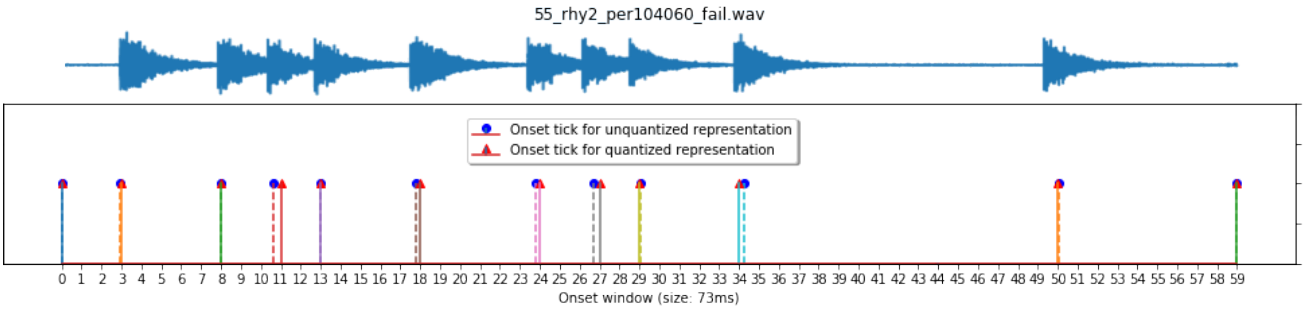


Figure 3. Example of waveform and onset information (detection method: HFC) before and after quantizing procedures. The stems in both ends relate with the fact that audios were prior cropped to range from first to last onsets

Feature	Coefficient	Intercept
Beat Difference	-44.31	
Rhythmic Similarity	72.71	
Levenshtein	-8.94	
Damerau-Levenshtein	11.17	-54.85
Jaro	**-8229.71**	
Jaro-Winkler	**6894.59**	
Hamming	**-988.81**	
Yule Similarity	188.32	

Table 2. Feature coefficients and intercept for the linear regression model trained with HFC onsets. Features with higher influence over predictions are highlighted

Estimator	MAE	MSE	R^2
Fixed grading to 2	1.12	1.71	-1.52
Fixed grading to 3	0.71	0.76	-0.04
Random grading	0.99	1.52	-1.53
Linear Regression (FLUX)	0.69	0.69	0.01
Linear Regression (COMPLEX)	0.59	0.57	0.23
Linear Regression (HFC)	**0.59**	**0.50**	**0.21**

Table 3. Comparison between error measures (Mean Average Error, Mean Squared Error and R-squared) observed in predictions for naive estimators and proposed model trained with different features

tions matching the jury evaluations (*pass/fail*). Final results report a matching rate of 65% for the binary predictions while the grade estimator guessed the right class for 70% of the unseen data.

7. CONCLUSIONS

The present study is an attempt to address the lack of data sources designed for automatic rhythmic assessment. Student performances for a set of rhythmic patterns were recorded and evaluated by a jury during entrance exams conducted in a music conservatory. An additional data annotation step was also carried out with seven annotators, this time grading a subset of these performances with grades ranging from one to four. The re-annotated subset trained an automatic assessment system able to predict

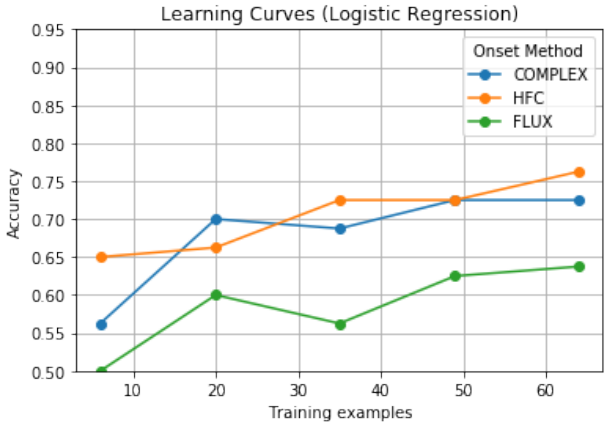


Figure 4. Learning curve for the *pass/fail* logistic classifier

students evaluations for rhythmic tasks. Models delivered a maximum accuracy of 76% for a binary (*pass/fail*) classifier and presented a minimum mean average error of 0.59 when predicting grades in a linear fashion, also pointing to Jaro, Jaro-Winkler and Hamming distances as the best model predictors. When compared with the data remained from the re-annotation process, the baseline assessment system predictions matched the jury-labeled data in about 70% of the cases.

All the aforementioned artifacts are now made public for further investigation. Both the full dataset and the re-annotated subset can be freely accessed and used to support assessment systems that builds on more sophisticated techniques to predict student grades for rhythmic lessons. Besides, the proposed implementation is also made available as Jupyter notebooks that can be examined and used as baseline in comparative studies.

8. ACKNOWLEDGEMENTS

This work was conducted during a visiting scholar period at Universitat Pompeu Fabra, sponsored by the Capes Foundation within the Ministry of Education, Brazil (grant n. 88881.189929/2018-01). The dataset was collected during a project funded by the Scientific and Technological Research Council of Turkey, TUBITAK, Grant [215K017].

9. REFERENCES

[1] Ozan Baysal, Baris Bozkurt, Turan Sager, and Nilgun Dogrusoz. Towards a hybrid assessment model for music conservatory entrance exams. In *Proceedings of Education Studies '18 - II.International Conference on Education and Learning Conference, Istanbul*, pages 81–96. DAKAM, 2018.

[2] Juan Pablo Bello, Chris Duxbury, Mike Davies, and Mark Sandler. On the use of phase and energy for musical onset detection in the complex domain. *IEEE Signal Processing Letters*, 11(6):553–556, 2004.

[3] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez Gutiérrez, Sankalp Gulati, Perfecto Herrera Boyer, Oscar Mayor, Gerard Roma Trepat, Justin Salamon, José Ricardo Zapata González, and Xavier Serra. Essentia: An audio analysis library for music information retrieval. In *Britto A, Gouyon F, Dixon S, editors. 14th Conference of the International Society for Music Information Retrieval (ISMIR); 2013 Nov 4-8; Curitiba, Brazil.[place unknown]: ISMIR; 2013. p. 493-8.* International Society for Music Information Retrieval (ISMIR), 2013.

[4] Baris Bozkurt, Ozan Baysal, and D Yuret. A dataset and baseline system for singing voice assessment. In *Proceedings of the International Symposium on Computer Music Multidisciplinary Research (CMMR), Matosinhos, Portugal*, pages 25–28, 2017.

[5] Seung-Seok Choi, Sung-Hyuk Cha, and Charles C Tappert. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, 8(1):43–48, 2010.

[6] Norberto Degara, Antonio Pena, Matthew EP Davies, and Mark D Plumbley. Note onset detection using rhythmic structure. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5526–5529. IEEE, 2010.

[7] Simon Dixon, Fabien Gouyon, Gerhard Widmer, et al. Towards characterisation of music via rhythmic patterns. In *ISMIR*. Citeseer, 2004.

[8] Jonathan Foote and Shingo Uchihashi. The beat spectrum: A new approach to rhythm analysis. page 224. IEEE, 2001.

[9] M Goto and T Nishimura. Aist humming database: Music database for singing research. *IPSJ SIG Notes (Technical Report)(Japanese edition)*, 2005(82):7–12, 2005.

[10] Bochen Li, Xinzhao Liu, Karthik Dinesh, Zhiyao Duan, and Gaurav Sharma. Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications. *IEEE Transactions on Multimedia*, 21(2):522–535, 2019.

[11] Paul Masri and Andrew Bateman. Improved modelling of attack transients in music analysis-resynthesis. In *ICMC*, 1996.

[12] Emilio Molina, Isabel Barbancho, Emilia Gómez, Ana Maria Barbancho, and Lorenzo J Tardón. Fundamental frequency alignment vs. note-based melodic similarity for singing voice assessment. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 744–748. IEEE, 2013.

[13] Tomoyasu Nakano, Masataka Goto, and Yuzuru Hiraga. An automatic singing skill evaluation method for unknown melodies using pitch interval accuracy and vibrato features. In *Ninth International Conference on Spoken Language Processing*, 2006.

[14] Kumar Pati, Siddharth Gururani, and Alexander Lerch. Assessment of student music performances using deep neural networks. *Applied Sciences*, 8(4):507, 2018.

[15] Graham Keith Percival. *Computer-assisted musical instrument tutoring with targeted exercises*. PhD thesis, 2008.

[16] Rodrigo Schramm, Helena de Souza Nunes, and Cláudio Rosito Jung. Automatic solfège assessment. In *ISMIR*, pages 183–189, 2015.

[17] Sam Thompson and Aaron Williamon. Evaluating evaluation: Musical performance assessment as a research tool. *Music Perception: An Interdisciplinary Journal*, 21(1):21–41, 2003.

[18] Godfried T Toussaint. A comparison of rhythmic similarity measures. In *ISMIR*, 2004.

[19] George Tzanetakis and Perry Cook. Multifeature audio segmentation for browsing and annotation. In *Proceedings of the 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. WASPAA'99 (Cat. No. 99TH8452)*, pages 103–106. IEEE, 1999.

[20] Burak Uyar and Baris Bozkurt. An interactive rhythm training tool for usuls of turkish makam music. In *5th Int. Workshop on Folk Music Analysis (FMA)*, Paris, France, 2015.

[21] Amruta Vidwans, Siddharth Gururani, Chih-Wei Wu, Vinod Subramanian, Rupak Vignesh Swaminathan, and Alexander Lerch. Objective descriptors for the assessment of student music performances. In *Audio Engineering Society Conference: 2017 AES International Conference on Semantic Audio*. Audio Engineering Society, 2017.

[22] Brian C Wesolowski, Stefanie A Wind, and George Engelhard. Examining rater precision in music performance assessment: An analysis of rating scale structure using the multifaceted rasch partial credit model. *Music Perception: An Interdisciplinary Journal*, 33(5):662–678, 2016.

[23] Chih-Wei Wu, Siddharth Gururani, Christopher Laguna, Ashis Pati, Amruta Vidwans, and Alexander Lerch. Towards the objective assessment of music performances. In *Proceedings of International Conference on Music Perception and Cognition (ICMPC)*, pages 99–102, 2016.

[24] Chih-Wei Wu and Alexander Lerch. Learned features for the assessment of percussive music performances. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, pages 93–99. IEEE, 2018.

LEARNING SEMANTIC SIMILARITY IN MUSIC VIA SELF-SUPERVISION

Mason Bretan
Samsung Research America
mason.bretan@samsung.com

Larry Heck
Samsung Research America
larry.h@samsung.com

ABSTRACT

Neural networks have been used to learn a latent "musical space" or "embedding" to encode meaningful features and provide a method of measuring semantic similarity between two musical passages. An ideal embedding is one that both captures features useful for downstream tasks and conforms to a distribution suitable for sampling and meaningful interpolation. We present two new methods for learning musical embeddings that leverage context while simultaneously imposing a shape on the feature space distribution via backpropagation using an adversarial component. We focus on the symbolic domain and target short polyphonic musical units consisting of 40 note sequences. The goal is to project these units into a continuous low dimensional space that has semantic relevance. We evaluate relevance based on the learned features' abilities to complete various musical tasks and show improvement over baseline models including variational autoencoders, adversarial autoencoders, and deep structured semantic models. We use a dataset consisting of classical piano and demonstrate the robustness of our methods across multiple input representations.

1. INTRODUCTION

Music is inherently complex. A single motif can be described along a multitude of dimensions. Some of these dimensions may describe the motif in broad terms and capture properties that offer a more aggregate representation including tonality, note density, complexity, and instrumentation. Others may consider the sequential nature and temporal facets of music such as syncopation, harmonic progression, pitch contour, and repetition. While these features may describe specific attributes about the music, they are intrinsically related and when combined can be used to predict or classify higher level musical descriptors such as genre, style, or even mood and emotion.

In recent years, neural networks have been used to learn a low dimensional latent "musical space" or "embedding" to encapsulate such features and provide a method of measuring semantic similarity between two musical passages [26]. Ideally, the embedding $f(x) \in \mathbb{R}^d$ for a passage x is learned such that the Euclidean distance $D_{i,j} = ||f(x_i) - f(x_j)||^2$ between two passages describes their semantic relationship. For the task of autonomous music generation learning this space effectively is important in order to influence the generator such that its outputs conform to human expectations. This is particularly true in interactive applications where a machine's response is typically conditioned on a human performer. Thus, an effective embedding is one that is capable of interpreting music in a manner which correlates with human perception.

Previously, musical embeddings have been learned using restricted boltzman machines (RBMs) [9, 22, 27], autoencoders (with various denoising techniques) [2, 3], siamese network models [4,13,23], word2vec models [11], and sequence prediction models [1, 6, 18]. Variational autoencoders (VAEs) have also demonstrated some success with monophonic inputs [25]. VAEs learn a normally distributed latent space which has shown to make sampling and embedding manipulation more effective [15]. Though VAEs are useful for constraining the statistical properties of the learned space, it has also been shown that, like word embeddings in language, improved features can be learned when networks are trained to reconstruct the context. The resulting features perform well on prediction and composer classification tasks [2].

An ideal embedding is one that both captures useful features and conforms to a distribution suitable for sampling and meaningful interpolation. In this work, we present two methods for learning musical embeddings. The methods leverage context and simultaneously impose a shape on the feature space distribution via backpropagation using an adversarial component. We focus on the symbolic domain and target short musical units such as a three second clip or a sequence consisting of a small of number notes. The goal is to project these units into a continuous low dimensional space that has semantic relevance. We evaluate relevance based on the learned features' abilities to complete several music-related tasks. We use a dataset consisting of classical piano and demonstrate that it is possible impose a prior distribution on the embeddings while maintaining the quality of the features.

2. RELATED WORK

One possible scenario for learning a space which encodes semantic similarity is to explicitly label pairs of musical units as being similar and train a model in a supervised fashion, thus, pulling units labeled as similar closer together and pushing non-related units further apart in the latent space. This type of metric learning can be effective, however, the requirement for constructing such a dataset makes it challenging. In this work we focus on self-supervised methods that don't require explicitly labeled data.

RBMs, autoencoders, and prediction are all examples of self-supervised learning paradigms. They rely on what is readily available in the data to serve as a proxy to human labeled data, thus, autonomously constructing the supervising signal. Many of the methods have been inspired from the natural language processing community including skip-gram and language models [20]. For example, both [11] and [18] used skip-gram inspired techniques to embed chords. The learned tonal space had similarities to the circle of fifths. Context is also leveraged in [4] where a Deep Structured Semantic Model (DSSM) is used learn one, two, and four bar embeddings [12]. Such siamese network techniques have proven useful for learning semantic similarity in language. In [21] labeled pairs were used to train siamese networks to effectively learn sentence similarity. A pairwise ranking loss was similarly used for the task of hit song prediction [30].

RBM and autoencoder methods don't leverage context, but have still demonstrated the ability to learn good features compared to manually designed features. This is particularly true in the audio domain [9, 16, 29]. In [7] autoencoders were used to learn a latent space encoding timbre. Autoencoders are particularly useful because they inherently support contain a generative component and if learned effectively the embeddings can be manually manipulated for interactive applications [3]. VAEs, in particular, have shown promise and utility for music because the latent space is regularized in a manner that makes sampling and manipulation more convenient and meaningful [15, 24, 25].

The methods in this work are inspired by adversarial autoencoders [19]. Like VAEs, the goal of this type of autoencoder is to constrain the latent codes to some arbitrary prior distribution. However, instead of using a KL-divergence penalty, the autoencoder incorporates an adversarial method to train the distribution of the latent codes to match that of the prior distribution.

While both VAEs and adversarial autoencoders are useful for generation, it has been shown that autoencoder features are not as effective for downstream tasks compared to methods that include context or prediction [2]. In this work we propose a solution that computes a pairwise loss based on context and includes an adversarial component to regularize the latent space.

3. METHODS

We present two methods. At a high level the embeddings of semantically similar units should be geometrically closer in the latent space than units that are dissimilar. For each method the objective is to learn a space that achieves this by leveraging context while adhering to a predefined distribution. In lieu of explicitly labeled data we train the networks using the assumption that two adjacent units (i.e. two adjacent measures in a composition) are related. In other words the distance between two adjacent units should be smaller than two random units in the database.

3.1 Adversarial DSSM

Our first method is a modified implementation of the DSSM. If $q(z)$ represents the aggregated posterior distribution of all the embeddings of length d generated by the DSSM $f(x)$ for $x \in X$ then the goal here to is match $q(z)$ to a prior distribution $p(z)$ we define as $z_i \sim \mathcal{N}_d(\mu, \sigma^2)$ where $\mu = 0$ and $\sigma^2 = 1$. This is achieved by connecting a discriminator to the last layer of the DSSM as shown in Figure 3. This discriminator is trained adversarially in coalescence with the generator which also happens to be the DSSM itself.

The standard DSSM training procedure is well-suited for metric learning as it explicitly trains parameters to produce embeddings that are closer together (according to a distance metric) for related items while pushing non-related items away. However, the number of negative examples and the ratio of easy to hard examples is usually greatly biased towards the easy-end. This often produces poor performance since many examples can satisfy the constraint with a very small loss that provides no real meaningful update during backpropagation [5]. This typically leads to high inter-class and low intra-class variance making fine grained categorization or meaningful similarity measures (important for music) challenging or impossible.

To address this problem a bootstrapping method was used in [5] in which particularly difficult examples were manually mined from the dataset and used during different stages of training. In this work, the adversarial component naturally helps to mitigate this problem by enforcing the prior distribution. The network parameters must find a way to achieve the desired similarity metric, but adhere to a distribution that does not allow for a learned space in which most examples can easily satisfy the similarity constraint.

The adversarial DSSM is trained in two stages: 1) Using the standard DSSM technique compute a softmax loss with negative examples and 2) Using the adversarial network train the generator and discriminator so that the generator is trained to produce embeddings that look as if they have been sampled from the predefined distribution $p(z)$. Thus, the parameters are being optimized according to two different losses with one learning the similarity metric and the other learning to describe the data such that the aggregated posterior distribution of the embeddings are Gaussian and continuous.

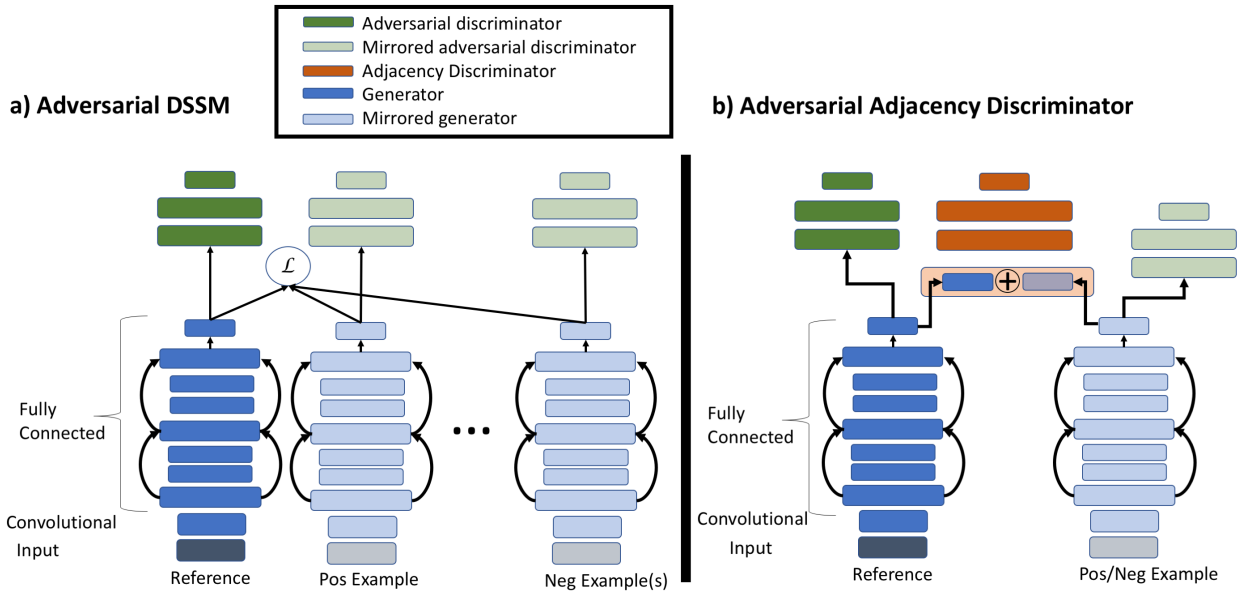


Figure 1. Architectures for a) Adversarial DSSM and b) Adversarial Adjacency Model

For the first part the network is trained using Euclidean similarity.

$$sim(\tilde{X}, \tilde{Y}) = \frac{1}{1 + D(\tilde{X}, \tilde{Y})} \tag{1}$$

Negative examples are included in a softmax function to compute $P(\tilde{R}|\tilde{Q})$ where $\vec{R}$ is the reconstructed vector and $\vec{Q}$ is the input vector.

$$P(\tilde{R}|\tilde{Q}) = \frac{\exp(sim(\tilde{Q}, \tilde{R}))}{\sum_{\tilde{d} \epsilon D} \exp(sim(\tilde{Q}, \tilde{d}))} \tag{2}$$

The network learns the parameters by minimizing the following loss function using gradient descent:

$$\mathcal{L} = -log \prod_{(Q,R)} P(\tilde{R}|\tilde{Q}). \tag{3}$$

For the second part generative adversarial network (GAN) training procedures are used [8]. First, the adversarial discriminator is trained to distinguish between the generated embeddings and vectors sampled from $q(z)$. Second, the generator (also the DSSM or $f(x)$) is trained to fool the discriminator. We use a deterministic version of the GAN where stochasticity comes solely from the data distribution. In other words no additional randomness is incorporated.

Training alternates between the DSSM and GAN procedures until Eqn. 3 converges. We found that a higher learning rate for the GAN procedures (particularly for updating the generator) relative to the DSSM loss was necessary in order to get the desired results. Otherwise, the GAN based updates had very little to no effect resulting in a model with a very similar behavior to the vanilla DSSM without the adversarial component.

3.2 Adversarial Adjacency Model

The second method we propose is also inspired by siamese network paradigms. However, unlike the DSSM, the embeddings are not directly optimized for the desired metric. Instead, a classifier is trained to determine whether two units are related or not. Though, because we use adjacency as the self-supervising surrogate signal in lieu of manually designed similarity labels the classifier is really being trained to determine whether two units would be contiguous or not in a composition.

A simple version of this classifier would concatenate *both* units into a single input and be trained to produce a binary classification from this concatenated vector. Our goal, however, is to be able to embed a *single* unit and having a network which requires two units as input would prevent this. Therefore, we use tied weights in which the lower layers of the network are identical and the embeddings are not concatenated until several layers deep into the network (see Figure 3). In other words the two inputs are embedded independently, but use the same parameters to do so.

A much smaller classifier is attached to the top of the concatenated embeddings to discriminate between related and non-related inputs. By using only a couple layers to perform the actual discrimination most of the good features for classification will need to be learned by the embedding portion. This enforces the network to embed an input in a manner that not only efficiently encodes itself, but also can effectively distinguish itself from unrelated inputs. Our thinking was that hopefully the network would achieve this by embedding related units closer together. (Note, for ease of comparison the architecture of the embedding network here is identical to the DSSM network in the previous section).

Like the previous method we attach an adversarial component to the end of the embedding portion of the network. The goal is the same in that the aggregated posterior distribution of the embeddings should conform to a predefined distribution. The model is trained in two stages: 1) Train the classifier to discriminate between related and non-related inputs and 2) Train the embeddings to fit a

prior distribution using the GAN scheme. Therefore, the embedding portion of the model is updated during both stages.

For the first part the classifier is trained using cross entropy with two classes (related and non-related).

$$-\sum_{c=1}^{M} y'_c \log(y_c) \tag{4}$$

where $M = 2$, y' is the predicted probability and y is the ground truth. The GAN portion is trained similarly to the previous method. We found that this method was inherently more stable than the previous method and much less tuning of the learning rates between the two losses were necessary.

4. EXPERIMENTS

In order to test the various networks and training procedures we used a collection of piano compositions from 27 artists. The distribution of compositions among artists is depicted in Table 4. At least two songs from each artist were held out for testing. We augmented the data by transposing each piece into all keys. This also prevented the networks from simply learning the bias any composers might have had for specific key signatures. We also augmented the data by altering the tempo randomly within a range of .95 to 1.05 of the original.

Composer	Num. Train Songs	Num. Test Songs
Albeniz	15	2
J.S. Bach	8	2
Bartok	21	4
Beethoven	30	4
Borodin	8	2
Brahms	31	5
Burgmueller	10	2
Byrd	34	4
Chopin	49	6
Clementi	17	2
Couperin	10	2
Debussy	10	2
Galuppi	6	2
Grieg	17	2
Handel	20	3
Haydn	20	3
Scott Joplin	57	4
Liszt	17	2
Mendelssohn	16	2
Mozart	22	3
Mussorgsky	9	2
Rachmaninov	10	2
Ravel	5	2
Scarlatti	6	2
Schubert	30	4
Schumann	25	3
Tschaikovsky	13	2

Table 1. Piano Music Dataset.

4.1 Baseline Models

We compare our methods against four baseline models for a total of six methods:

1. Variational autoencoder (VAE)
2. Adversarial autoencoder (AAE)
3. Deep structured semantic model (DSSM)
4. Adjacency Discriminator (AdjD)
5. Adversarial deep structured semantic model (A-DSSM)
6. Adversarial Adjacency Discriminator (A-AdjD)

4.2 Input Representation

Often performance of a music-based model is ultimately determined by the input representation of the data. Therefore, we test our system using two different input representations.

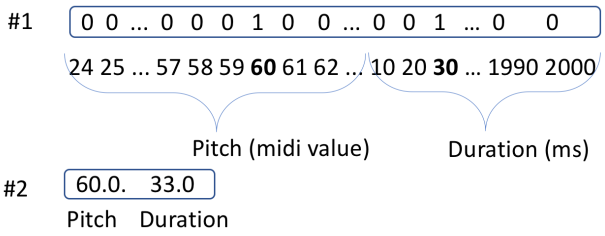


Figure 2. Two representations for encoding middle 'C' (midi note 60) and a 33ms interval to the next pitch. Input representation #1 uses a two-hot encoding that discretizes time in 10ms chunks. Input representation #2 uses a single floating point value for both pitch and time.

The first representation is inspired by Google Magenta's event based method [28] in which each event is one-hot encoded. Because much of the data does not include relevant volume information we do not include it in our representation. We also use only the onset time and use a fixed duration for each note, therefore, it is not necessary to include note off events. This was primarily to simplify the input space and attempt to better interpret the results. Pitch is represented in a one-hot manner on a vector representing midi values 24 to 96. We encode intervals between pitch events discretely in 10ms intervals from 0 to 2000ms, thus, the time portion of the vector consists of 200 values. The single vector for all possible events including pitch and interval has a length of 272. A two-hot encoding method is used so both the pitch and interval before the next pitch is encoded using the 272 values (see Figure 4.2).

The second representation is much less conventional. We represent both pitch and time intervals continuously. The pitch of a note is represented by a single floating point value determined by its midi value and the interval between notes is also represented by a single floating point value determined by the number of milliseconds. Therefore a vector for a single event only has a length of two (4.2).

On the surface, this continuous pitch representation does not make a lot of sense as the Euclidean distance in this space does not really translate to pitch distances

that are particularly meaningful in Western music or piano. An autoencoder trained to reconstruct this input using a Euclidean-based loss is unlikely to learn many relevant musical features. However, the two methods proposed in this work do not compute the loss in the original space. The adversarial DSSM optimizes the latent space by computing the loss directly on the embeddings and the adversarial adjacency model optimizes the latent space through a classification task using a cross entropy loss. Therefore, we hypothesize that our methods should be more robust against varying input representations compared to models that compute a loss in the original space (e.g. autoencoders). Additionally, a method that is robust when using this representation can be useful for styles of music or particular instruments in which continuous pitch representations are more appropriate.

In this continuous representation the pitches and intervals are standardized to a mean of zero and standard deviation of one. In our experiments we did not find a difference in our final results when compared to using non-standardized input vectors, however, the learning was faster using standardized vectors.

We focus on short musical units consisting of exactly 40 notes. This means that the input vector to the network using the first representation has a length of 10880 (40 * 272). The input vector using the second representation has a length of only 80 (40 pitches and 40 intervals). Forty notes was chosen because it provides enough content to capture local structure, yet, the vector length using the first representation is not overwhelmingly large.

4.3 Architectures

The first layer of the network is convolutional using a filter with an input length equivalent to a vector containing one note and one interval (i.e. 272 for the first representation and 2 for the second). The filter is convolved over the entire vector using an equivalent stride length (272 or 2), thus, the filter learns to encode a single pitch and interval. After this initial convolutional layer all remaining layers are fully connected.

For each method the portion of the network which performs the embedding is the same aside from marginal differences in the number of parameters for the first convolultional layer between the two representations. The network encodes the input into a 32-dimensional vector. Each network has eight layers between the final embedding and input vector and uses residual connections [10] as depicted in Figure 3.

It is plausible that higher capacity networks (both wider and deeper) may improve results further, however, the primary objective in this work is to compare various training methods and not architectures. We designed this architecture because it allowed us to leverage the utility of deep learning and specific techniques (e.g. convolution, residual connections, etc.), yet, it is not too large that training time would become problematic during experimentation. Each layer uses LeakyReLU and the parameters are updated using Adam optimization [14, 17].

4.4 Experiments

We perform five different experiments based on music-related tasks. Performance on these tasks will be used to determine the efficacy of the learned latent space and features for the various models.

Ranking The primary measure for evaluation is based on a ranking task. Given a reference unit and a group of 100 units consisting of 99 random units from the database and one unit that is adjacent to the reference (either before or after) the task is to rank all 100 units according to their Euclidean distance to the reference in the latent space. This is repeated for each unit in the test set and a mean rank is reported where the a lower rank indicates a higher similarity. The assumption is that adjacent units should be geometrically closer in the latent space relative to non-adjacent units.

Composer Classification We evaluate how useful the learned features are for classifying the units according to their composers. Using the embeddings as inputs we train a simple two layer network to perform classification. In the test set there are 27 possible composers. Though works by these composers are seen during the training phase, the test set consists of unique compositions that were not available during training.

Pitch Chroma Prediction We evaluate whether the embedding retains enough information about the input units to reconstruct a unit's chromagram representation. We train a two layer network to minimize a softmax function (Eqn. 3) over cosine similarity. Thus, in Eqn. 2 we replace the Euclidean similarity (Eqn. 1) with $sim(\tilde{X}, \tilde{Y}) = \frac{\tilde{X}^T \cdot \tilde{Y}}{|\tilde{X}||\tilde{Y}|}$. The positive example is the true chroma extracted from the unit computed directly from the original input vector. Negative examples come from chroma extracted from random units in the data.

Note Density Regression We evaluate whether the embedding retains enough information about the input units to describe their note density. Because we use a fixed number of notes per unit the network is trained to predict the unit's duration in seconds. We use root mean square error (RMSE) to measure performance.

Forward Prediction We evaluate whether a sequential model can be trained to predict the embedding of the next unit in a composition given a sequence of the previous seven units. This task is related to the first Ranking task, but focused on generation and prediction rather than general distances in the learned manifold. We train two stacked LSTM cells, each with 200 units, to predict the next step of a sequence. Specifically, at each time step, the input to the network is a the 32-dimensional embedding vector of a 40 note unit x_i. We train this network to predict the 8^{th} unit, x_{i+7} given the previous 7 units $x_i, z_{i+1}, \ldots x_{i+6}$. This means that 280 notes of context are provided before the prediction is made.

For each given target x_{i+7} as described above, we create a set of 32 embedding vectors: one is $f(x_{i+7})$, the true embedding for the target. The other 99 vectors are embeddings of randomly selected units in the data set. The Euclidean distance is measured between the output of the

LSTM and the encodings of each unit. The distances are then sorted and ranked similarly to the first Ranking experiment.

5. RESULTS

The results for each experiment are reported in Tables 2-6.

Method	Representation #1	Representation #2
VAE	10.8	12.3
AAE	9.9	12.3
DSSM	11.2	14.8
AdjD	5.8	6.3
A-DSSM	8.8	9.2
A-AdjD	5.1	5.6

Table 2. **Ranking Results.** Geometric means are reported for the adjacency ranking task. A lower score indicates a better result.

Method	Representation #1	Representation #2
VAE	.11	.087
AAE	.11	.091
DSSM	.18	.10
AdjD	.26	.26
A-DSSM	.23	.19
A-AdjD	.27	.28

Table 3. **Composer Classification** Macro-F1 scores are reported for the composer classification task. A higher score indicates a better result.

Method	Representation #1	Representation #2
VAE	.56	.43
AAE	.57	.44
DSSM	.34	.27
AdjD	.71	.68
A-DSSM	.72	.67
A-AdjD	.83	.78

Table 4. **Chroma Predication** Geometric means of cosine similarities between predicted and ground truth chroma. A higher score indicates a better result.

Method	Representation #1	Representation #2
VAE	.33	.28
AAE	.33	.29
DSSM	.20	.18
AdjD	.26	.20
A-DSSM	.21	.19
A-AdjD	.21	.19

Table 5. **Note Density Regression** RMSE values are reported for note density regression. The deviations were measured in seconds. A lower score indicates a better result.

Method	Representation #1	Representation #2
VAE	5.9	8.7
AA	5.9	8.6
DSSM	10.7	11.5
Adj	2.7	3.0
A-DSSM	4.8	5.2
A-Adj	2.6	2.7

Table 6. **Forward Prediction** Geometric means are reported for the adjacency ranking task. A lower score indicates a better result.

5.1 Discussion

The vanilla DSSM performed relatively poorly on all tasks except for note density regression. Without the adversarial component it learns the most significant feature (in this case note density in time), yet, fails to learn much more beyond this. By fitting the latent space to a prior distribution, the adversarial component seems to do what it was designed for – preventing the model from satisfying the similarity constraint without actually learning too many meaningful features.

Our adjacency discriminator model worked reasonably well even without the adversarial training. We found that without the adversarial component the latent embeddings were naturally fairly close to a Gaussian distribution, thus, adding an adversarial discriminator had much less of an effect than when used with the DSSM. Though the additional fine tuning did improve performance across the tasks albeit marginally.

Finally, the differences in performance for the two input representations were much less pronounced for our methods. This suggests our proposed methods (the Adversarial Adjacency Discriminator in particular) may be useful for non-Western music and instruments capable of continuous pitch spaces.

6. CONCLUSION

In this work we described new approaches to self-supervised metric learning. The learned features showed improved results on downstream tasks over various baseline methods. Most importantly, the quality of the features were either improved or maintained when imposing a prior distribution on the embeddings. The next steps for this work are to: 1) develop decoders from the latent spaces learned from our methods and 2) measure the perceptual significance of the learned space.

7. REFERENCES

[1] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

[2] Mason Bretan, Sageev Oore, Doug Eck, and Larry Heck. Learning and evaluating musical features with

deep autoencoders. In *KDD Workshop on Machine Learning for Creativity*, 2017.

[3] Mason Bretan, Sageev Oore, Jesse Engel, Douglas Eck, and Larry Heck. Deep music: towards musical dialogue. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[4] Mason Bretan, Gil Weinberg, and Larry Heck. A unit selection methodology for music generation using deep neural networks. In *Proceedings of the 8th International Conference on Computational Creativity, Atlanta*, 2017.

[5] Yin Cui, Feng Zhou, Yuanqing Lin, and Serge Belongie. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1153–1162, 2016.

[6] Douglas Eck and Juergen Schmidhuber. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Proceedings of the 12th IEEE workshop on neural networks for signal processing*, pages 747–756. IEEE, 2002.

[7] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. Neural audio synthesis of musical notes with wavenet autoencoders. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1068–1077. JMLR. org, 2017.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[9] Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks. In *ISMIR*, volume 10, pages 339–344. Utrecht, The Netherlands, 2010.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Cheng-Zhi Anna Huang, David Duvenaud, and Krzysztof Z Gajos. Chordripple: Recommending chords to help novice composers go beyond the ordinary. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, pages 241–250. ACM, 2016.

[12] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338. ACM, 2013.

[13] Yu-Siang Huang, Szu-Yu Chou, and Yi-Hsuan Yang. Similarity embedding network for unsupervised sequential pattern learning by playing music puzzle games. *arXiv preprint arXiv:1709.04384*, 2017.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[16] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009.

[17] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[18] Sephora Madjiheurem, Lizhen Qu, and Christian Walder. Chord2vec: Learning musical chord embeddings. In *Proceedings of the constructive machine learning workshop at 30th conference on neural information processing systems (NIPS'2016), Barcelona, Spain*, pages 1–5, 2016.

[19] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[21] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[22] Juhan Nam, Jorge Herrera, Malcolm Slaney, Julius O Smith, et al. Learning sparse feature representations for music annotation and retrieval. In *ISMIR*, pages 565–570, 2012.

[23] Xiaoyu Qi, Deshun Yang, and Xiaoou Chen. Triplet convolutional network for music version identification. In *International Conference on Multimedia Modeling*, pages 544–555. Springer, 2018.

[24] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[25] Adam Roberts, Jesse Engel, and Douglas Eck. Hierarchical variational autoencoders for music. In *NIPS Workshop on Machine Learning for Creativity and Design*, 2017.

[26] Fanny Roche, Thomas Hueber, Samuel Limier, and Laurent Girin. Autoencoders for music sound synthesis: a comparison of linear, shallow, deep and variational models. *arXiv preprint arXiv:1806.04096*, 2018.

[27] Jan Schluter and Christian Osendorfer. Music similarity estimation with the mean-covariance restricted boltzmann machine. In *2011 10th International Conference on Machine Learning and Applications and Workshops*, volume 2, pages 118–123. IEEE, 2011.

[28] Ian Simon and Sageev Oore. Performance rnn: Generating music with expressive timing and dynamics. *Magenta Blog: https://magenta. tensorflow. org/performancernn*, 2017.

[29] Jan Wülfing and Martin A Riedmiller. Unsupervised learning of local features for music classification. In *ISMIR*, pages 139–144, 2012.

[30] Lang-Chi Yu, Yi-Hsuan Yang, Yun-Ning Hung, and Yi-An Chen. Hit song prediction for pop music by siamese cnn with ranking loss. *arXiv preprint arXiv:1710.10814*, 2017.

BLENDING ACOUSTIC AND LANGUAGE MODEL PREDICTIONS FOR AUTOMATIC MUSIC TRANSCRIPTION

Adrien Ycart[1*] **Andrew McLeod**[2*] **Emmanouil Benetos**[1] **Kazuyoshi Yoshii**[2]
[1] Queen Mary University of London, UK [2] Kyoto University, Japan
a.ycart@qmul.ac.uk, mcleod@sap.ist.i.kyoto-u.ac.jp

ABSTRACT

In this paper, we introduce a method for converting an input probabilistic piano roll (the output of a typical multi-pitch detection model) into a binary piano roll. The task is an important step for many automatic music transcription systems with the goal of converting an audio recording into some symbolic format. Our model has two components: an LSTM-based music language model (MLM) which can be trained on any MIDI data, not just that aligned with audio; and a blending model used to combine the probabilities of the MLM with those of the input probabilistic piano roll given by an acoustic multi-pitch detection model, which must be trained on (a comparably small amount of) aligned data. We use scheduled sampling to make the MLM robust to noisy sequences during testing. We analyze the performance of our model on the MAPS dataset using two different timesteps (40ms and 16th-note), comparing it against a strong baseline hidden Markov model with a training method not used before for the task to our knowledge. We report a statistically significant improvement over HMM decoding in terms of notewise F-measure with both timesteps, with 16th note timesteps improving further compared to 40ms timesteps.

1. INTRODUCTION

The ultimate goal of the task of Automatic Music Transcription (AMT) is to convert an audio signal into some form of human- or machine-readable music notation [2]. This process is divided into two main steps. First, an acoustic model performs multi-pitch detection by converting an input acoustic signal into a *posteriogram*: a pseudo-piano roll matrix which contains the probability of each pitch being present at each timestep according to the acoustic model. Next, a music language model (MLM) is used to enforce some musicality on the results, converting the posteriogram into a human-readable format.

While it is desirable to run these two models jointly—and some systems have been designed in such a way with success, either with relatively simple MLMs (e.g., [19,20]), or for performing a simpler task than AMT (e.g., chord detection [15])—the search space and resulting computation quickly becomes too large to be feasible for more complex probabilistic MLMs which explicitly model musical structure (e.g., [22]). Furthermore, such MLMs have typically been designed to take as input MIDI (or MIDI-like) data which consists of lists of musical notes, rather than the typical posteriogram output of acoustic models.

However, the conversion of a posteriogram into MIDI is not a trivial task, and has not been the focus of much research until recently. A naive approach is simple thresholding of the posteriogram, and a simple two-state (on/off) HMM has also been proposed [17]. Some more sophisticated models have attempted to use neural networks as implicit MLMs to incorporate some prior musical knowledge into their systems (e.g., [21,24]), but often, they bring only modest improvement [21] or the MLM is only used in rare occasions [24] (see Section 2 for a more complete discussion on these and other related systems). The main issues that we identify with these previous attempts to incorporate MLMs are (1) the MLM fails to capture musical features because of an inappropriately short timestep which overemphasizes self-transitions [25], and (2) the MLM is not robust to noise during decoding.

Our main contributions are to:

1. compare the use of a musically-relevant timestep for MLM decoding—specifically a 16th note, as recommended in [25]—to the more standard 40ms.
2. train the MLM with scheduled sampling [3], making it more robust to noise at test time.
3. propose a novel "blending" model which dynamically merges probabilities from the acoustic model and the MLM rather than using a simpler method such as a linear combination.
4. describe a new training method for a previously-proposed post-processing HMM [17], leading to a significant improvement in F-measure over the standard maximum likelihood approach.

For contribution (1), note that in a realistic setting, using a 16th note timestep would require a beat-tracking algorithm. However, in this proof-of-concept experiment, we consider 16th note locations as given, and leave the integration of a noisy 16th note timestep for future work. It should be noted that a 16th note timestep was already investigated in [27] for polyphonic sequence transduction, which concluded that although an improvement is observed when using 16th note timesteps, it is attributed only to the fact

*Authors 1 and 2 contributed equally to this work.

that the resulting outputs are quantized to the ground truth metrical grid. By contrast, we show here that a 16th-note MLM brings improvement for language model decoding beyond quantisation of the output.

We conduct our experiments using a state-of-the-art piano-specific acoustic model [13] (although our system could be applied to a variety of musical instruments and styles). Overall, we show that our full system with the 16th note timestep, scheduled sampling, and the blending model, leads to a significant improvement in F-measure over both the baseline and the proposed HMM, with identical timesteps.

2. RELATED WORK

The most basic strategy for binarizing time-pitch posteriograms is simple thresholding, where outputs below some value are set to 0 while all others are set to 1 (e.g., in [9, 11, 13]). Slightly more sophisticated is to use a two-state (on/off) HMM for each pitch (proposed in [17], used in, e.g., [6, 7]), where results tend to be cleaner with fewer spurious notes using this method. Still, the musicality of such a model is limited, as it considers each pitch independently, and does not consider more than one previous frame for each transition.

Recently, deep learning methods have also been used for this task, typically using some form of RNN. They can be broadly grouped into performing one of two tasks: transduction or language model decoding.

Transduction methods aim to convert one sequence of symbols into another (here, the output of an acoustic model into a binary piano roll). Examples include [5], which uses an architecture combining an RNN with a Restricted Boltzmann Machine (RBM), and [27], which investigated the performance of an LSTM-based model. One drawback of transduction methods is that they require aligned MIDI and audio recordings for training. Similarly, they are trained on one specific acoustic model's outputs, and do not necessarily generalize to other acoustic models.

With *language model decoding* methods, an MLM is trained to assess the likelihood of an output sequence. Importantly, such a model is trained on symbolic data, independent of any acoustic model. For example, [21] uses an RNN-RBM as a language model, combined with various neural acoustic models. Similarly, [24] uses an RNN-RBM language model, but instead of using a fixed framerate, it operates on frames corresponding to detected inter-onset-intervals. Our method belongs to this second category of language model decoders, with the caveat that one component of it, the blending model, requires training on aligned pairs of input and output, though much less than would be needed to train a neural transduction model.

As mentioned, using an MLM has brought only limited improvement to the performance of AMT systems in the above studies. One reason for this lack of substantial improvement in [21] might be the use of an inappropriate timestep for language modelling: the MLM operates on 32ms timesteps, a duration much shorter than the typical duration of a note, and unrelated to the tempo of the piece being analyzed. Indeed, [25] hints at the fact that for polyphonic music sequence prediction, using a small time-step only results in a smoothing effect due to the predominance of self-transitions, and using a musically-relevant time step such as a 16th note allows the network to learn more interesting musical properties. To that end, [24] describes an MLM which uses note-based timesteps. However, the MLM was only used in the rare case that a note onset was detected without a corresponding pitch. Using the MLM over the whole note sequence resulted in decreased performance over simple thresholding, possibly due to the discrepancy between training using perfect inputs and decoding noisy sequences (this issue was also noted in [21]).

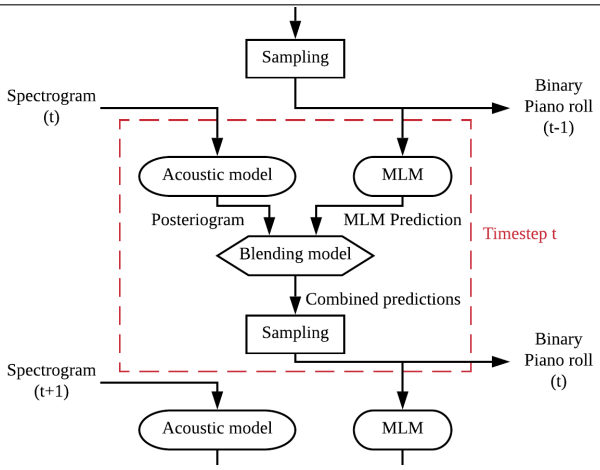


Figure 1. The proposed system.

3. PROPOSED SYSTEM

Our system takes as input a probabilistic piano roll, specifically the output of the acoustic model from [13]. That model is a CNN which takes as input a spectrogram with logarithmically-spaced frequency bins and log-magnitude with a timestep of 40ms, and is a benchmark acoustic model for piano transcription.

Our system's inputs are in the form of matrix $I \in \mathbb{R}^{N_p \times T}$, where T is the length of the input in frames, $N_p = 88$ (one row per key on a piano keyboard), and each element $I_{p,t}$ contains the probability of a pitch p being present at frame t. Our output is the binary matrix $O \in \{0,1\}^{N_p \times T}$, where $O_{p,t}$ is 1 if pitch p is present at frame t, and 0 otherwise.

Our system flow is shown in Fig. 1. It consists of two main components: an LSTM-based language model (see Section 3.1), which predicts the presence of each pitch at a frame given the previous frames; and a blending model (see Section 3.2), which combines the input acoustic prior with the LSTM's priors at each frame, and outputs a final combined probability distribution over pitches at each frame. The search process for finding the most probable output according to our system is detailed in Section 3.3.

3.1 Language Model

The language model has the same architecture as described in [25]. It is a single-layer LSTM, with a hidden layer of

size 256 and sigmoid outputs of size N_p. It is trained to predict which pitches might be present in the next frame of a binary piano roll, given all the previous (binary) frames, using cross-entropy between the output of the network and the actual next frame as training loss. Two different MLMs were trained, one operating on 40ms timesteps, and one on 16th note timesteps. We use $L_{p,t}$ to denote the MLM's output corresponding to pitch p at frame t.

During a typical training procedure, the input sequences from which the network learns are taken directly from the ground truth. Such an MLM learns to make prediction based only on perfect musical sequences. However, during inference, the MLM must make predictions based on potentially noisy sequences, as input frames are obtained from previous predictions and noisy acoustic multi-pitch detections. This discrepancy hinders performance of MLMs, as noted in [21] and [24]. To solve this problem, we use scheduled sampling [3]: during training, at each timestep, instead of always using the ground-truth frame, we randomly choose either the ground-truth frame (with probability p_{GT}), or a frame sampled from predictions made by the MLM at the previous timestep. Training starts with $p_{GT} = 1$, and p_{GT} is decreased as training progresses, allowing the MLM to progressively become more robust to noisy inputs and recover from previous mistakes.

One limitation is that the noise the MLM adapts to is not the same the MLM sees at test time, since samples are drawn using the acoustic model's outputs as well at test time. We could sample from a distribution that does the same during training, but we choose not to, because (1) [3] mentions that even adding uniform noise helps performance, so exactly matching noise distributions is less important, and (2) this would require paired audio and MIDI data for MLM training (as acoustic model predictions must be made from audio), which is available in smaller quantities than MIDI data alone.

3.2 Blending Model

The intuition behind the blending model is that the MLM and the acoustic model might each perform better or worse in certain situations, so combining their probabilities with a constant weight may achieve poor results. The blending model's job is to learn the situations in which each model performs well, and output the combined prior for a pitch at a frame based on both the probabilities from the acoustic model and the MLM, as well as some surrounding context.

It is a feed-forward neural network with l [1] hidden layers with 5 nodes each followed by an output layer with a single sigmoid. For each pitch p at time t, it takes as input: (1) the acoustic and language priors at that pitch and frame ($I_{p,t}$ and $L_{p,t}$), (2) the sample history at that pitch for the previous $hist$ [1] frames ($O_{p,t'}$ for $max(0, t - hist) \leq t' < t$), and (3) nine additional hand-crafted features, described in Table 1, resulting in an input vector of length $11 + hist$.

The sample history allows the model to learn if there are certain situations in which the LSTM performs particularly well or poorly. Features 1–4 model how peaked the output distribution from each model is (and thus how certain it might be) but with different nonlinear properties. Features 5–6 model the expected polyphony, features 7–8 model how fast-changing each model's predictions are, and feature 9 allows the blending model to learn if either model performs better or worse for high or low pitches.

Feature	Description	Equation
1–2	Uncertainty	Eqn. (1)
3–4	Entropy	Eqn. (2)
5–6	Mean	$\sum_{p'<N_p} \frac{I_{p',t}}{N_p}$
7–8	Flux	$I_{p,t} - I_{p,t-1}$
9	Pitch	$\frac{p}{N_p}$

Table 1. The features used for the blending model. For equations written using I, the second feature is calculated identically with L.

$$\text{Uncertainty} = \sum_{p'<N_p} \begin{cases} (1 - I_{p',t})^2 & I_{p',t} > 0.5 \\ (I_{p',t})^2 & I_{p',t} \leq 0.5 \end{cases} \quad (1)$$

$$\text{Entropy} = \frac{1}{log_2(N_p)} \sum_{p'<N_p \wedge I_{p',t} \neq 0} -I_{p',t} log_2(I_{p',t}) \quad (2)$$

We create two versions of the blending model. First, a weight model (WM) which outputs a weight $w_{p,t}$, which is used to calculate a blended prior $P_{p,t}$ as a weighted sum: $P_{p,t} = w_{p,t} I_{p,t} + (1 - w_{p,t}) L_{p,t}$. Second, a prior model (PM) which outputs $P_{p,t}$ directly. The main difference between the two models is that WM can only ever result in a $P_{p,t}$ that lies somewhere between $I_{p,t}$ and $L_{p,t}$, while PM can always output any $P_{p,t}$ between 0 and 1.

3.3 Search Process

Since our model's search space has a branching factor of 2^{N_p} at each frame, we cannot perform a global search. Therefore, we use Viterbi decoding [23] with beam search using a beam of size b and a branching factor k. Specifically, at each frame, we save only the b most probable histories to that point. Then, for each of those at frame t, using the blending model's output distribution $P_{p,t}$, we sample the k most probable samples using Algorithm 2 from [5] (again saving only the top b from the $b * k$ resulting hypotheses). The sample at frame t is denoted S_t, and is a set containing the pitches active at that frame. The probability of a sample S_t, given the blended priors $P_{p,t}$ is:

$$P(S_t) = \prod_{p' \in S_t} P_{p',t} \prod_{p'<N_p \wedge p' \notin S_t} 1 - P_{p',t} \quad (3)$$

Beam search has the drawback that the beam can easily become saturated with only slight variations of the most probable hypothesis. Therefore, similar to [21] and [15], we use a hashed beam search. We consider any two hypotheses which are identical for the past h frames to be duplicates of each other for our purposes, and only save the most probable of them.

The final output O of our system is constructed using the sample history of the most probable state in the beam, such that $O_{p,t}$ is 1 if $p \in S_t$ and 0 otherwise.

[1] See Section 4.5 for details on the training of l and $hist$.

4. EXPERIMENTS

4.1 Data

For our experiments, we use the MAPS dataset [10], which contains MIDI-aligned recordings of various classical music pieces, some as played by an upright Disklavier, and some synthesized using high-quality piano samples. We create the exact same test set as was used in [13] (which was created in the same way as *Configuration II* from [21], with the additional constraint that only the Disklavier recordings were used). We create our training and validation sets slightly differently because the blending model requires a reasonably-sized validation set on which to train. From the remaining synthesized pieces, we choose 20 to become the validation set (counting multiple synthesized recordings of a single piece as only 1), and use the remaining pieces for training. This results in final split sizes of 59 pieces for test, 105 for training, and 32 for validation. The decrease in training set size compared to [13] does not seem to affect the performance of the acoustic model. We train the acoustic model on the whole pieces, but our evaluation is performed on the first 30 seconds of each recording, as is usually done, e.g. in [13].

To train our MLM, we use MIDI files taken from the Piano-midi.de [2] dataset. This dataset currently holds 324 pieces of classical piano music from various composers, with both quantised note durations and expressive timings. Every piece in MAPS can be found in the Piano-midi.de dataset, as these files were used to create MAPS. To avoid training the MLM with pieces later used for testing, we split the dataset using the same pieces as in the MAPS splits: all the pieces in the MAPS test set are used for testing (52 pieces), all the pieces in the MAPS validation set are used for validation (20 pieces), and all the remaining pieces are used for training (252 pieces).

We use two different timesteps in our experiments: 40ms (the resolution of [13]); and 16th-note, in which the input is divided into 16th-note frames based on the metrical grid. For the 16th-note timesteps, we use the metrical annotations from the A-MAPS dataset [26]. To downsample the acoustic prior for the 16th-note timesteps, we take the average of its original 40ms frames for the duration of each new frame. Before evaluation, we upsample our outputs back to 40ms timesteps, assigning each resulting frame the value of the corresponding output frame.

4.2 Metrics

We report both framewise and two versions of notewise precision (P), recall (R), and F-measure (F_1), each of which is averaged across all recordings in the test set. The frame-based metrics are standard as used in the MIREX Multiple-F0 Estimation task [1], comparing the output piano roll to the ground truth piano roll, using 40ms frames (after upsampling when using 16th-note timesteps). Since our model does not output onsets and offsets explicitly, we treat any output 1 not preceded by a 1 as an onset, and any 0 not preceded by a 0 as an offset. We treat the ground truth the same after first converting it into a piano roll. Thus, our "notewise" metrics do not correspond with notes exactly, but rather as close as our output format can get. We leave an analysis using proper notes for future work.

We also perform two post-processing steps for the notewise metrics, for all methods: (1) minimum duration pruning, where we remove any notes shorter than 50ms; and (2) gap filling, where we fill rests shorter than 50ms. Additionally, we report both onset-only (On) and onset-offset (OnOff) notewise results. For On, a note is considered correct if its pitch is correct and its onset time is within 50ms of the ground truth, and for OnOff, we add the constraint that the offset is such that the note duration is within 20% of ground truth (or 50ms, whichever is biggest). Both are as described for note-tracking in [1], and we use `mir_eval` [18] to perform all calculations.

As argued in [12], the most relevant metrics are the notewise metrics. Indeed, a poor transcription system could still score high in terms of framewise F_1 if its only errors correspond to short spurious notes and fragmentation of held notes. When discussing our results, we thus concentrate mainly on the notewise metrics. Furthermore, the onset-only metrics are the most commonly-used ones for the task, and onsets are much more perceptually important (and salient) than offsets [4, 8]. Thus, our main evaluation concentrates on onset metrics, and we discuss the OnOff metrics only in Section 5.4.

4.3 Configurations

Besides the two versions of our blending model described in Section 3.2 (WM and PM), we use a baseline blending model: a constant weight (CW) model, which always calculates $P_{p,t}$ similarly to WM, but using the constant $w_{p,t} = 0.8$ for all p and t (a value set in an ad hoc fashion on the validation set). CW should indicate whether the adaptability of the blending model is important for performance. For each blending model, we train a version both with and without scheduled sampling (using +S to denote its use) for each timestep.

There is a risk with PM that the blending model might choose to dismiss the MLM input completely. With WM, even if the MLM is not used to choose the weight, it will still have an influence on the resulting probabilities, unless the blending model's output is exactly 1. To see whether our improvement comes from the MLM or simply the use of the blending model, we also train a blending model which is identical to configuration PM+S, except that any of its inputs which come from the MLM's predictions are set to 0 at both train time and test time (this includes the MLM prediction itself as well as various features which use the MLM's output). We call this configuration PM-A, and present a brief discussion of its results in Section 5.2.

4.4 Baselines

We compare our models against that of [13], retrained with our training and validation sets. We threshold its output at a 0.5, setting all values ≥ 0.5 to 1 and all others to 0.

We also compare our model against a common HMM baseline [17] where each pitch is represented by a simple 2-state (on/off) HMM, run independently. Typically, the

[2] `http://piano-midi.de/`

Method	40ms timesteps						16th note timesteps					
	Framewise			On-Notewise			Framewise			On-Notewise		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
[13]	73.0	**65.5**	**68.3**	54.2	65.7	58.1	75.2	65.5	69.2	70.9	63.6	65.9
HMM	74.6	63.9	68.0	64.0	62.2	61.9	73.7	**69.4**	**70.7**	74.4	66.4	69.1
CW	73.7	64.9	68.2	61.2	63.3	61.1	76.6	61.6	67.3	76.1	55.7	63.0
CW+S	73.9	64.8	68.2	61.3	63.3	61.3	76.6	61.5	67.3	74.8	55.9	62.6
WM	73.0	64.8	67.9	63.0	60.8	61.0	77.2	62.6	68.0	75.6	61.4	66.7
WM+S	75.1	62.3	67.3	**67.6**	60.2	**62.8**	**77.8**	61.2	67.5	**79.4**	58.5	66.0
PM	**81.8**	50.8	60.8	57.0	**66.5**	59.9	77.2	63.9	69.1	72.4	66.4	68.3
PM+S	79.7	57.4	65.6	61.4	65.6	62.6	77.6	64.2	69.3	76.8	**68.7**	**71.7**

Table 2. Results of all experiments, with all timesteps, with the best values in bold. CW uses our constant weight model, WM uses the weight model, and PM uses the prior model. +S denotes the use of scheduled sampling in training.

HMM raises precision and lowers recall, removing short spurious notes from the output. In [17], one HMM is trained per pitch class. For transposition invariance, we instead train a single HMM, and use it for all pitches. In previous work, it has been standard to use maximum likelihood estimation (MLE) to set the transition probabilities (by counting transitions in some training set), and to treat the input probabilistic piano roll directly as the observation probabilities. We instead learn the transition probabilities with Bayesian Optimization (BO) [16] to maximize the notewise F_1 on the validation set. There are only two probability values to search for (since $P(\text{off}|S) = 1 - P(\text{on}|S)$). We use the validation set instead of the larger training set so that the HMM has noisier observations during training (for both MLE and BO), and we set the initial state probabilities to a uniform distribution.

The resulting HMM is one which is much more likely to change states: for 40ms timesteps, $P(\text{on}|\text{off})$ is 0.004 with MLE and 0.493 with BO, and $P(\text{off}|\text{on})$ is 0.167 with MLE and 0.196 with BO. 16th-note timesteps see a similar change. Specifically, the probability for transitioning from off to on is much greater, likely because the observed data is much more accurate at note onsets, and thus the model can safely trust those values in most cases. The BO-trained HMM leads to a significant increase in both framewise and notewise F_1 for both timesteps compared to the MLE-trained HMM (MLE results omitted).

4.5 Training

The MLM is trained using the Adam optimizer [14] with a learning rate of 0.001. Piano rolls are cut into smaller sequences of 750 frames for 40ms timesteps and 300 frames for the 16th-note timesteps. We augment the data by transposing each sequence by a number of semitones randomly chosen between -5 and 7 at each epoch, so that each tonality is equally represented without shifting the note range too much. We use early stopping, such that if the cross-entropy evaluated on the validation dataset does not decrease for 200 epochs, training is stopped, and the best model so far is kept. For scheduled sampling, we decrease p_{GT} linearly from 1 to 0.7 over 500 epochs. Validation is done using a fixed value of $p_{GT} = 0.7$, and we use early stopping once the schedule is finished (after 500 epochs).

The blending model is trained on the validation set to maximize On-notewise F_1. Training data is generated by running our MLM on the first 30 seconds of each piece in the validation set with a fixed weight of 0.8 and a beam size of 10. We save a data point—containing the priors, a sample history of length $hist$, and features—for each (frame, pitch, hypothesis) triple for which the acoustic prior differs from the language prior by at least Δ_{min}. Bayesian Optimization for 200 iterations is used to search for the values of Δ_{min} and $hist$ (up to 10 for 16th-note timesteps and up to 50 for 40ms timesteps), and how many hidden layers l to use (1–4 of size 5).

The parameters for the beam search are set in an ad hoc fashion on the validation set. The beam size b and the branching factor k have small effects, where larger values lead to better results, but slower computation, and we use $b = 50$ and $k = 5$ for evaluation. The hash length h has an effect where smaller values force the model to perform a more global search, but with less ability to make decisions based on frames further in the past. We use a value of $h = 12$ for evaluation.

5. RESULTS

Full framewise and On-notewise results are in Table 2. Overall, we can see that for 40ms timesteps, PM+S and WM+S outperform all other models in the On-notewise F_1 ($p < 10^{-3}$ with a paired t-test, which we use for all significance tests), but WM+S does not significantly outperform PM+S. For 16th note timesteps, PM+S is significantly better than all other models for On-notewise F_1 ($p < 10^{-6}$). The baselines achieve the best framewise results—HMM for 16th-note ($p = 0.021$ over PM+S) and [13] for 40ms (not significant). This makes sense, as our blending models are optimised for On-notewise F_1. Moreover, our MLM is designed to take advantage of knowledge of musical structure, which is more clear at the note level.

In the following sections, we investigate the impact of each component of our system: the timestep (5.1), scheduled sampling (5.3), and the blending model (5.2). The OnOff-notewise metrics are presented in Section 5.4.

5.1 Timestep

It is clear that using 16th-note timesteps improves the results (framewise for the baselines and On-notewise for all

methods). Similar results were seen in [27], which showed that the improvement was mainly due to quantisation of the output. Here, the increase in performance of [13] is due entirely to this quantisation. However, when performing the same quantisation procedure to the output of the 40ms-timestep PM+S, we see framewise and On-notewise F_1 of only 63.6 and 62.9 respectively, significantly worse than PM+S with a 16th-note timestep ($p < 10^{-6}$ for both). PM also sees a significant improvement when using a 16th-note timestep directly ($p < 10^{-8}$ for both), as well as WM+S, but for On-notewise F_1 only ($p = 0.01$). Weaker or insignificant effects are seen for our other models, both framewise and On-notewise. This shows that for our MLM, much of its improvement with 16th-note timesteps is due to its ability to learn more musical patterns at that scale, particularly when the full system has the ability to take advantage of that knowledge. In the following sections, therefore, we concentrate on the results with the 16th-note timestep.

5.2 Blending model

The adaptability of WM and PM clearly allow them to outperform CW, and the wider output range of PM leads to improvement over WM ($p = 0.018$ for WM over CW without scheduled sampling, $p < 10^{-6}$ for all other pairs), although WM's precision is greater. For the framewise metrics, a similar pattern is seen, though the effect is weaker ($p < 0.001$ for PM over WM, $p = 0.048$ from WM over CW, and not significant for WM+S over CW+S).

To see whether our improvement comes from the MLM or simply from the use of the blending model, we evaluate the PM-A configuration. This model achieves an On-notewise F_1 of 65.1 with a 16th-note timestep, significantly worse than PM+S ($p < 10^{-9}$), which shows that both the MLM and the blending model play an important part in our system's performance. In the following sections, we concentrate on PM results.

5.3 Scheduled sampling

We can see that PM+S performs significantly better overall than PM for On-notewise ($p < 10^{-7}$), but not framewise F_1 ($p = 0.50$). Looking at the results in a piecewise fashion leads to an interesting conclusion about where that improvement comes from. In Figure 2, we plot the notewise F_1 of [13] (x-axis, a proxy for the noisiness of the input) against the increase in On-notewise F_1 for PM+S over PM (y-axis) for each piece in our test set. Here, it can be seen that PM+S outperforms PM by a greater margin in exactly those cases that we expect scheduled sampling to help: when the input is noisier. This correlation is significant ($p = 0.02$), but with high variance ($R^2 = 0.09$). Overall, we can conclude that scheduled sampling does indeed lead to improved performance with noisy inputs.

5.4 Onset-offset Evaluation

Table 3 presents the OnOff-notewise results for the two baselines and our overall best performing model (PM+S), where it can be seen that our model significantly outperforms the other systems ($p < 10^{-5}$ for both timesteps).

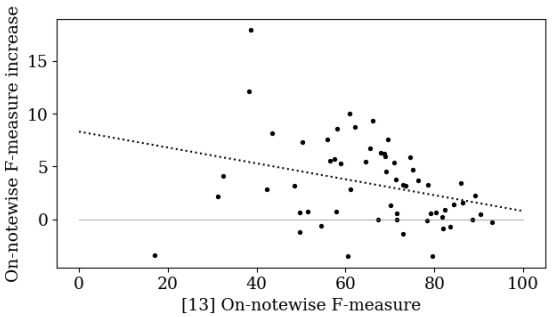


Figure 2. Increase of On-notewise F_1 of PM+S over PM plotted against On-notewise F_1 of [13] for each piece. Dotted line shows linear correlation ($p = 0.02$, $R^2 = 0.09$).

Method	40ms timestep			16th-note timestep		
	P	R	F_1	P	R	F_1
[13]	31.8	37.7	33.8	41.0	37.3	38.5
HMM	37.8	36.5	36.5	41.3	37.5	38.8
PM+S	**41.5**	**43.2**	**41.8**	**47.7**	**43.3**	**45.0**

Table 3. Results using the OnOff-notewise metrics. PM+S uses our prior model with scheduled sampling.

This is promising, and it seems that the MLM might have learned some rhythmic components of musical structure.

6. CONCLUSION

In this paper, we have presented a system for converting a posteriogram output of an acoustic multi-pitch detection system into a binary piano roll. Our system consists of an LSTM-based MLM and a feed-forward neural blending model to combine the MLM outputs with those from the acoustic model. We have shown that our system performs significantly better than thresholding the posteriogram, as well as post-processing it with a new strong baseline HMM. We have further shown that (1) scheduled sampling helps the MLM perform better in the case of noisy inputs, and (2) the use of a 16th-note timestep allows the MLM to learn musical structures better than with a 40ms timestep.

To that end, in future work, we intend to investigate the use of noisy 16th-note labels from a beat-tracking system, rather than the ground truth labels that we have used here. Furthermore, we will perform a systematic ablation study for the input features of the blending model. We also intend to analyze our results in a more qualitative fashion in future work with listening tests. Our own subjective conclusions are that our system does often produce more musical results than the baselines, but a proper listening test would show more objectively whether that is the case throughout the test set, and what aspects of the resulting piano rolls become more musical in which cases. In the meantime, we provide some examples of our model's performance as supplementary material [3], along with the code to reproduce our experiments [4].

[3] c4dm.eecs.qmul.ac.uk/ycart/ismir19.html
[4] github.com/adrienycart/MLM_decoding

7. ACKNOWLEDGEMENTS

Thanks to Dr. Eita Nakamura for his help on this project. A. Ycart is supported by a QMUL EECS Research Studentship. A. McLeod and K. Yoshii are supported in part by JST ACCEL No. JPMJAC1602, JSPS KAKENHI No. 16H01744 and No. 19H04137, and the Kyoto University Foundation. E. Benetos is supported by UK RAEng Research Fellowship RF/128.

8. REFERENCES

[1] Mert Bay, Andreas F. Ehmann, and J. Stephen Downie. Evaluation of Multiple-F0 Estimation and Tracking Systems. In *10th International Society for Music Information Retrieval Conference (ISMIR)*, pages 315–320, 2009.

[2] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[3] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.

[4] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 255–261, 2016.

[5] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. High-dimensional sequence transduction. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3178–3182, 2013.

[6] F.J. Cañadas Quesada, N. Ruiz Reyes, P. Vera Candeas, J.J. Carabias, and S. Maldonado. A multiple-f0 estimation approach based on gaussian spectral modelling for polyphonic music transcription. *Journal of New Music Research*, 39(1):93–107, 2010.

[7] Dorian Cazau, Yuancheng Wang, Olivier Adam, Qiao Wang, and Grégory Nuel. Improving note segmentation in automatic piano music transcription systems with a two-state pitch-wise hmm method. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 523–530, 2017.

[8] Adrien Daniel, Valentin Emiya, and Bertrand David. Perceptually-based evaluation of the errors usually made when automatically transcribing music. In *9th International Society for Music Information Retrieval Conference (ISMIR)*, 2008.

[9] Arnaud Dessein, Arshia Cont, and Guillaume Lemaitre. Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence. In *11th International Society for Music Information Retrieval Conference (ISMIR)*, pages 489–494, 2010.

[10] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010.

[11] Graham Grindlay and Daniel P. W. Ellis. Multi-voice polyphonic music transcription using eigeninstruments. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 53–56, 2009.

[12] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. *19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[13] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. *17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 475–481, 2016.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[15] Filip Korzeniowski and Gerhard Widmer. Improved chord recognition by combining duration and harmonic language models. *19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[16] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2, 1978.

[17] Graham E. Poliner and Daniel P. W. Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, 5(1):154–162, Oct 2006.

[18] Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Dan P. W. Ellis. mir_eval: A transparent implementation of common MIR metrics. In *15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[19] Matti P. Ryynanen and Anssi Klapuri. Polyphonic music transcription using note event modeling. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 319–322. IEEE, 2005.

[20] Rodrigo Schramm, Andrew McLeod, Mark Steedman, and Emmanouil Benetos. Multi-pitch detection and voice assignment for a cappella recordings of multiple singers. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 552–559, 2017.

[21] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):927–939, 2016.

[22] David Temperley. A unified probabilistic model for polyphonic music analysis. *Journal of New Music Research*, 38(1):3–18, March 2009.

[23] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[24] Qi Wang, Ruohua Zhou, and Yonghong Yan. Polyphonic piano transcription with a note-based music language model. *Applied Sciences*, 8(3), 2018.

[25] Adrien Ycart and Emmanouil Benetos. A study on LSTM networks for polyphonic music sequence modelling. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 421–427, 2017.

[26] Adrien Ycart and Emmanouil Benetos. A-MAPS: Augmented MAPS dataset with rhythm and key annotations. In *ISMIR Late Breaking and Demos Papers*, 2018.

[27] Adrien Ycart and Emmanouil Benetos. Polyphonic music sequence transduction with meter-constrained LSTM networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 386–390, 2018.

MODELLING THE SYNTAX OF NORTH INDIAN MELODIES WITH A GENERALIZED GRAPH GRAMMAR

Christoph Finkensiep
EPFL*
christoph.finkensiep@epfl.ch

Richard Widdess
SOAS University of London
rw4@soas.ac.uk

Martin Rohrmeier
EPFL*
martin.rohrmeier@epfl.ch

ABSTRACT

Hierarchical models of music allow explanation of highly complex musical structure based on the general principle of recursive elaboration and a small set of orthogonal operations. Recent approaches to melodic elaboration have converged to a representation based on intervals, which allows the elaboration of pairs of notes. However, two problems remain: First, an interval-first representation obscures one-sided operations like neighbor notes. Second, while models of Western melody styles largely agree on stepwise operations such as neighbors and passing notes, larger intervals are either attributed to latent harmonic properties or left unexplained. This paper presents a grammar for melodies in North Indian rāga music, showing not only that recursively applied neighbor and passing note operations underlie this style as well, but that larger intervals are generated as generalized neighbors, based on the tonal hierarchy of the underlying scale structure. The notion of a generalized neighbor is not restricted to rāgas but can be transferred to other musical styles, opening new perspectives on latent structure behind melodies and music in general. The presented grammar is based on a graph representation that allows one to express elaborations on both notes and intervals, unifying and generalizing previous graph- and tree-based approaches.

1. INTRODUCTION

North Indian classical music (Hindustani music) provides valuable evidence for theories of syntactic musical organization. Like Western art music, it takes the form of aesthetic communication with an attentive and experienced audience, and is also a subject of theoretical discourse. Like most music outside the Western canon, it is normally unwritten, depending instead on memorization and improvisation. Instead of a system of chordal harmony or polyphony, Indian music comprises a solo melody against a complex background drone (of at least two pitches).

Melodic elaboration is prized as a means of musical extension and aesthetic enhancement: it operates at many levels, from the ornamentation of a single pitch, to the expansion of a phrase, to the architecture of a piece or performance. Melodic coherence is ensured by selecting one of a set of modes (rāga), each comprising a scale, a pitch hierarchy, and a set of licensed pitch transitions; any phrase that evokes a different rāga from the one selected is regarded as an error. It has been noted that Indian music resembles language in several respects [17], and a rāga could be understood as a melodic grammar, in which melodies are constructed by recursive elaboration over a hierarchically organized set of pitches.

The idea of understanding music in a hierarchical fashion goes back to Schenker [21], and has developed through the integration of impulses from generative linguistics and the theory of formal grammars since the 1980s [24, 1, 11, 20, 16]. Approaches most commonly addressed harmonic structure [23, 18, 19, 3, 5] and melody [4, 12]. Several approaches proposed simplified formalizations of Schenkerian theory and corresponding computational implementations [12, 13, 27, 10]. There is still comparably little discussion concerning the extent to which such hierarchical frameworks extend to non-Western forms of music. Narmour's theory of melodic processes is explicitly directed to capture melodies outside the Western canon as well [15]. The application of Schenkerian methods to non-Western music has been discussed by Stock [25]. More recently, it has been proposed to adapt analytical tools from Schenkerian analysis and the GTTM to Indian music [14, 2].

This paper links with this discourse and proposes a generalized formal model of North Indian melodic and phrase structure. A common shortcoming in previous models of melodic elaboration is the treatment of leaps, which are usually either attributed to a latent harmonic structure that is assumed to be known [12, 9], or modelled as probabilistic intervals [4, 6] without explicit restrictions. This paper introduces a formalism for relating leaps in North Indian music to a latent tonal hierarchy that is stated explicitly. With respect to this hierarchy, leaps can be viewed as instances of generalized neighbor- and passing-note relations that take into account the stability of a pitch in a scale. As will be argued, the generalized neighbor idea applies beyond North Indian music to some degree.

A central question for elaborative models concerns the representation of the music. Since formal grammars – the standard formalism for recursive elaboration – operate on

* École Polytechnique Fédérale de Lausanne

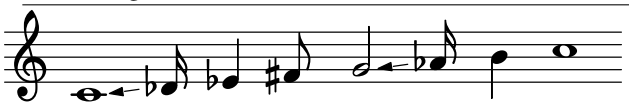


Figure 1: Rāga Multānī with pitches in an approximate Western notation. The notated duration denotes the hierarchical level, i.e. relative stability, of each pitch; arrows indicate a constraint on the resolution direction of an unstable pitch.

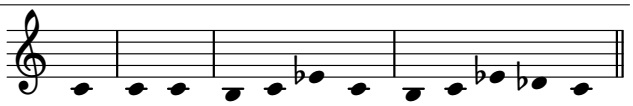


Figure 2: A short Multānī phrase and its derivation.

$d \in D_M$	1	♭2	♭3	♯4	5	♭6	7
$\delta_M(d)$	↕	↓	↕	↕	↕	↓	↕
$\lambda_M(d)$	4	0	2	1	3	0	2

Table 1: A formal description of the rāga Multānī, showing the direction and hierarchical level of each scale degree (as shown in Figure 1). ♭2 and ♭6 are directed downwards and can therefore only be used before 1 and 5, respectively.

strings of objects, most models of musical elaboration represent music as a sequence of objects, such as notes or chords. As a consequence, these models mostly focus on melodic [4, 6, 12] or homophonic settings [8].

A desirable property of a formal grammar is that it is *context-free*, meaning that elaborations on a single object are independent from the objects around it. Systems that are based on strings of notes have problems with being context-free since some elaboration operations (such as passing notes) depend on two notes [11]. Because of this, more recent approaches have been based on strings of intervals [12, 27, 4], which allow elaborations of both single notes and pairs of notes while remaining context-free. However, in an interval grammar, notes are represented implicitly and redundantly (as part of an incoming and outgoing interval). In addition, all notes generated by elaboration are derived from two parent notes, which is unintuitive for single-sided operations. As a unification and generalization of both approaches, this paper suggests a graph-based representation in which both notes and intervals are represented explicitly, with a graph grammar describing the elaboration rules. This goes beyond descriptions of *derivations* as graphs, which is already an established practice [12, 27, 10].

2. MELODIC OPERATIONS

Melody in Indian music is based on a set of modes called rāgas. A rāga is not only a collection of pitches that may be used, it also establishes a hierarchy of stability among these pitches. Stable pitches are those that can serve as resting points, while less stable pitches tend to move towards their more stable neighbors. Some pitches in a rāga have a preferred resolution direction and must resolve to the closest pitch in that direction. An example of a rāga with its scale, tonal hierarchy, and directional constraints is shown in Figure 1. The relative stabilities indicated in Figure 1 is based on observation of normal practice in this rāga.

The melodic elaboration of a rāga is performed most completely and systematically (though not exclusively) in ālāp: a type of improvisation in which the scale and melodic features of the rāga are gradually exposed in phrases unfolding an arch-shaped trajectory, starting from the root (scale-degree 1) and reaching the octave above (or higher) before finally returning to the root (a process called *vistār* or "scalar expansion" [26]). This background structure is filled and elaborated recursively, generating a complex foreground melody. Elaboration follows mainly two principles, inserting either passing or neighbor notes.

Passing notes fill intervals that are larger than steps. They can occur close to the surface (such as the ♭2 in ♭3 ♭2 1), but can also be understood to characterize dependencies in the background (e.g., filling the top-level interval 1 - 1′ with a 5). Two kinds of passing elaborations can be distinguished: Either a single note is introduced that subdivides the interval, potentially leaving non-step intervals that can be further elaborated; or the interval is filled with all scale notes enclosed by the interval.

Neighbor notes can be inserted before or after an existing note. While passing notes relate to both notes of an interval, neighbors are subordinate to single notes. When embellishing a note with a neighbor, a trade-off can be made between pitch proximity and stability: While unstable neighbors need to be very close to the main note's pitch, more distant neighbors can occur if they are sufficiently stable. In general, a pitch can only be perceived as a neighbor to some reference pitch if no pitch in the interval between the two is more stable than the proposed neighbor in the given mode.

Figure 2 shows the steps needed to derive a phrase using neighbors and passing notes. Starting with a single 1, the note is duplicated and elaborated twice, first with a lower neighbor 7, then with an upper neighbor ♭3. Finally, the space between ♭3 and 1 is filled with a passing ♭2.

3. MODES AND GENERALIZED NEIGHBORS

The idea of modes and generalized neighbors can be given a formal description: A *mode* M is a triple

$$\begin{aligned} M &:= (D, \delta, \lambda) \\ \delta &: \quad D \to \{\uparrow, \downarrow, \updownarrow\} \\ \lambda &: \quad D \to \mathbb{N} \end{aligned}$$

where D_M is a totally ordered set of *scale degrees*, δ_M is a function indicating the *direction* in which a scale degree is allowed to move, and λ_M returns the *hierarchical level* of a scale degree. For example, the rāga Multānī (Figure 1) would be formalized according to Table 1.

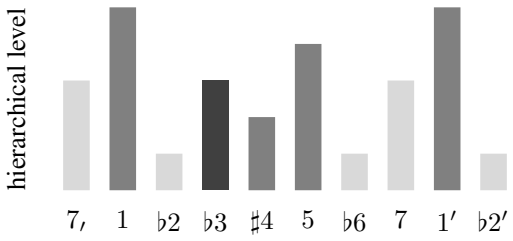


Figure 3: The upper and lower neighbors (dark) of ♭3 (black) in the Multānī rāga. Only pitches that can be reached without skipping a more stable pitch are neighbors. ♭2 is not a neighbor since it is directed downwards and can only be a neighbor to 1.

The same scale degree can be used as a *pitch* in different octaves, so pitches are indicated as scale degrees together with "′" for octaves above and "," for octaves below the default octave. The pitches of adjacent octaves are adjacent as well: $7,$ is directly below 1 and $1'$ is directly above 7. As a result, a mode gives rise to a set of pitches P_M, which corresponds to $\mathbb{Z}$ while scale degrees correspond to $\mathbb{Z}_{|D|}$. For convenience, δ_M and λ_M are assumed to be defined on pitches as well and return the values of the corresponding scale degrees.

The set of pitches *between* a pitch p_1 and a pitch p_2 is the set of pitches in the open interval (p_1, p_2) that agree with the direction of the interval:

$$\Delta_M(p_1,p_2) = \begin{cases} \{p \in P_M \mid & p_1 < p < p_2 \wedge \delta_M(p) \neq \downarrow\} \\ & \text{if } p_1 < p_2 \\ \{p \in P_M \mid & p_1 > p > p_1 \wedge \delta_M(p) \neq \uparrow\} \\ & \text{if } p_1 > p_2. \end{cases}$$

The *neighbors* of a pitch $p \in P_M$ are then all pitches $n \in P_M$ that have a higher level than all pitches *between* p and n. In addition, the direction of n must agree with the direction from n to p:

$$\begin{aligned} nb_M(p) = \{n \in P_M \mid p \neq n \\ \wedge \forall q \in \Delta_M(n,p) : \lambda_M(q) < \lambda_M(n) \\ \wedge\, n \rightarrowtail p\}, \end{aligned}$$

where

$$n \rightarrowtail p = \begin{cases} \delta_M(n) \neq \uparrow & \text{if } p < n \\ \delta_M(n) \neq \downarrow & \text{if } p > n \\ true & \text{otherwise.} \end{cases}$$

Thus, every pitch is a neighbor to p only if it can be reached from the reference pitch without skipping a more stable pitch than the neighbor, as illustrated in Figure 3. Directed pitches can only be inserted as left neighbors since they must move towards their resolution.

When a single passing note is generated, the passing note must be a neighbor to both notes of the interval it is inserted in. However, in this case the inserted note is moving away from the first note, so the direction is not towards the reference note but towards the neighbor. A *reverse neighbor* $r \in rnb_M(p)$ is defined in analogy to a neighbor but with inverted direction:

$$\begin{aligned} rnb_M(p) = \{r \in P_M \mid p \neq r \\ \wedge \forall b \in \Delta_M(p,r) : \lambda_M(b) < \lambda_M(r) \\ \wedge\, p \rightarrowtail r\}, \end{aligned}$$

For example, a passing ♭2 in the sequence ♭3 ♭2 1 is a neighbor to 1 but a reverse neighbor to ♭3, as it is directed away from ♭3 and towards 1.

Finally, a *fill* is the list of all pitches between two pitches p_1 and p_2, sorted according to the direction of the interval (p_1, p_2) and restricted to pitches agreeing with that direction (as given by Δ_M).

$$\mathit{fill}_M(p_1,p_2) = \begin{cases} sort(\Delta_M(p_1,p_2), asc) & \text{if } p_1 < p_2 \\ sort(\Delta_M(p_1,p_2), desc) & \text{otherwise.} \end{cases}$$

4. A FORMAL GRAMMAR OF RĀGA MELODIES

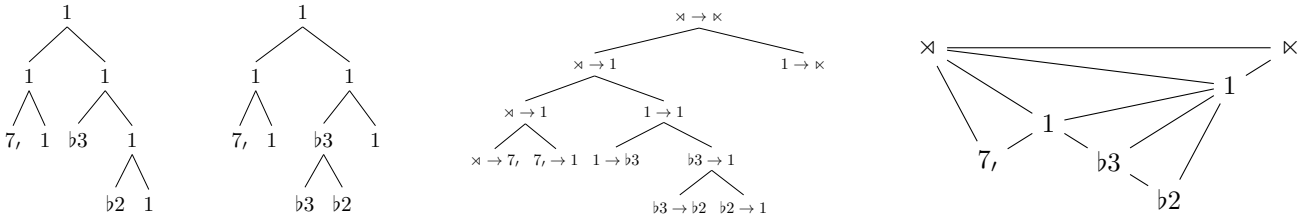
4.1 Representing Melodies as Graphs

As seen in Section 2, the two fundamental elaboration types – passing and neighbor notes – operate on two different musical entities: While neighbors elaborate single notes, passing notes fill intervals between two notes, elaborating both notes at the same time. As a consequence, two main formalisms describing hierarchical elaboration have emerged, note grammars and interval grammars.

Note grammars generate strings of notes, with derivation rules replacing single notes by several new notes. The resulting hierarchical structure is a tree of notes as shown in Figure 4a. However, elaborating single notes is problematic for passing notes, as they elaborate two notes. Not only is the resulting hierarchy ambiguous (the passing note must be attached to either its predecessor or its successor), but from a generative perspective, a passing note can only be derived from one of its parents. Thus, deciding where a passing note may be inserted becomes a context-sensitive problem.

Interval grammars [4, 9, 6, 12] solve the passing note problem (and two-sided operations in general) by elaborating pairs of notes, or intervals. Inserting a new note replaces an existing interval with two new intervals. The melody is then represented as a string of intervals with each note being represented twice, once as the second note of an interval and once as the first. To avoid this redundancy in notation, derivations are usually not given as trees (Figure 4b) but as outerplanar graphs (Figure 4c), giving each note two parents. However, for one-sided operations like neighbors, interval-based elaboration is conceptually misleading, as only one of the parent notes is considered while the other is ignored. This can lead to unwanted subordination of conceptually independent neighbors, as will be argued below.

As a unification and generalization of note- and interval-based systems, a graph-based representation of melodies is suggested here, representing notes as nodes



(a) Two analyses using note elaboration. The passing ♭2 must be attached either to the 1 on its right or to the ♭3 on its left.

(b) An analysis of the phrase using interval elaboration. The passing ♭2 is generated in the interval ♭3 → 1.

(c) The same analysis as in 4b, displayed as an outerplanar graph.

Figure 4: Conventional formal analyses of the phrase in Figure 2.

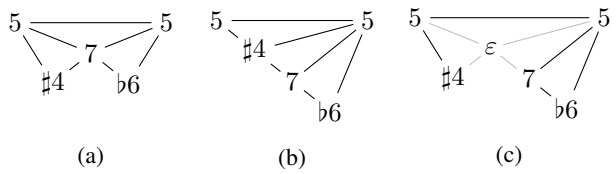


Figure 5: Three possible derivations of 5 ♯4 7♭6 5.

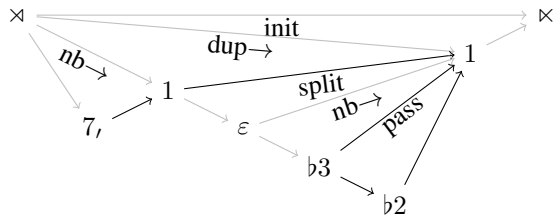


Figure 6: An analysis of the phrase in Figure 2 using the rāga grammar. Dark edges indicate the subgraph induced by removing non-note nodes ($\rtimes, \ltimes, \varepsilon$).

and note transitions as edges. Using graphs as the basis for elaboration has both conceptual and practical implications. Conceptually, graphs represent both notes and note transitions explicitly, which allows the use of both entities as a starting point for elaboration. Practically, while graphs can represent strings of objects (such as melodies) as a special case, they can easily describe much more complex structures, which potentially allows the description of elaboration operations in non-monophonic music. However, special cases such as monophony can still be defined graph-theoretically, ensuring consistency under elaboration. Thus, graphs provide a common framework for both melodic grammars and more complex formalisms.

While graphs in principle allow operations on both nodes and edges, a much simpler and more consistent system is obtained by operating only on edges, resulting in an *edge-replacement graph grammar*. All operations are then defined on edges (i.e., node transitions) with one-sided operations ignoring one node of the edge. One-sided operations still introduce an edge between the unused note and the new one in order to allow further elaboration between them. In order to express the independency between the new and the ignored note, a dummy node (written as ε) can be introduced first between any two notes. A dummy node does not generate a note and is analogous to the empty string in a conventional grammar.

Only strictly one-side operations can be performed on edges adjacent to a dummy node. This restriction expresses the independence between one-sided elaboration notes and their opposite side, and permits a more appropriate hierarchy: Suppose two one-sided neighbors are generated between two 5s, a ♯4, as a right neighbor to the first 5 and a 7 as a left neighbor to the second 5 with a passing ♭6, resulting in 5 ♯4 7♭6 5 (Figure 5). Without a dummy node, either 7 or ♯4 is subordinate to the other, depending on which is generated first (Figures 5a and 5b). By first introducing a dummy node, both neighbors can be derived independently (Figure 5c). Moreover, as dummy nodes are removed after the derivation, the resulting graph structure only retains edges that express elaboration dependence. Thus, dummy nodes allow the derivation to formally follow edge replacement while semantically expressing both one-sided and two-sided operations.

4.2 Formal Definition of the Grammar

A melody is formally represented as a directed linear graph with notes as nodes and transitions between notes as edges directed in time. The beginning and end of the melody are marked with the special nodes $\rtimes$ and $\ltimes$, respectively. The derivation is started from a single 1:

$$\rtimes \to 1_M \to \ltimes,$$

with 1_M indicating the root of mode M.

Derivation rules follow an edge-replacement paradigm: edges can be replaced with new subgraphs, retaining the nodes adjacent to the original edge. Some rules use only one of the adjacent nodes. In this case, a wildcard symbol ($* \in P_M \cup \{\rtimes, \ltimes, \varepsilon\}$) is used for the ignored node. The special symbol ε represents the *empty melody* and can be used to split an edge into two parts that may be elaborated independently. Only one-sided operations can be used on edges adjacent to an ε, $\rtimes$, or $\ltimes$.

For a given mode M the rāga grammar $\mathcal{G}_M^{rāga}$ is defined as the graph grammar $(\mathcal{T}, \mathcal{N}, \mathcal{I}, \mathcal{R})$ with

$$\mathcal{T} := \{n_1 \to n_2 \mid n_1 \in P_M \cup \{\rtimes, \varepsilon\}, n_2 \in P_M \cup \{\ltimes, \varepsilon\}\}$$
$$\mathcal{N} := \{\}$$
$$\mathcal{S} := \rtimes \to \ltimes$$

as terminals $\mathcal{T}$, non-terminals $\mathcal{N}$, and initial graph $\mathcal{S}$; and the following replacement rules $\mathcal{R}$:

initialize:
$(\rtimes \to \ltimes) \Rightarrow (\rtimes \to 1_M \to \ltimes)$

duplicate left: $\forall p \in P_M$:
$(p \to *) \Rightarrow (p \to p \to *)$

duplicate right: $\forall p \in P_M$:
$(* \to p) \Rightarrow (* \to p \to p)$

left neighbor: $\forall p \in P_M, n \in nb_M(p)$:
$(* \to p) \Rightarrow (* \to n \to p)$

right neighbor: $\forall p \in P_M, n \in nb_M(p) \wedge \delta_M(p) = \updownarrow$:
$(p \to *) \Rightarrow (p \to n \to *)$

passing: $\forall p_1, p_2 \in P_M, n \in rnb_M(p_1) \cap nb_M(p_2)$:
$(p_1 \to p_2) \Rightarrow (p_1 \to n \to p_2)$

fill: $\forall p_1, p_2 \in P_M$:
$(p_1 \to p_2) \Rightarrow (p_1 \to f_1 \to \ldots \to f_n \to p_2)$
where $f_1, \ldots, f_n = \mathit{fill}(p1, p2)$

split: $\forall p_1, p_2 \in P_M$:
$(p_1 \to p_2) \Rightarrow (p_1 \to \varepsilon \to p_2)$.

In this description, rules are given as templates that are instantiated for all (combinations of) pitches. A more elegant and efficient description is possible, if rules are considered to be functions on classes of structured symbols [7], allowing them to look inside their inputs.

Since the rāga grammar generates linear graphs, it is still possible to display derivations with outerplanar graphs. Figure 6 shows a derivation of the example phrase from Figure 2 using the rāga grammar. Each operation used to derive the phrase is written in the triangle formed by the old edge it replaces and the new edges it inserts. Later derivation graphs will omit operations and edge directions to remove visual clutter, as both are clear from the context.

In Figure 6, the ε inserted between the two 1s separates them and allows independent generation of neighbors. In particular, it would be possible to generate another right neighbor to the first 1 without subordinating it to the $\flat$3, or vice versa.

While the full derivation graph displays all derivation steps as they are formalized (i.e., as edge replacements), it does not distinguish one-sided and two-sided operations. Removing all non-note nodes ($\rtimes, \ltimes, \varepsilon$) and the adjacent edges induces a subgraph in which two-sided operations still use two edges while one-sided operations adjacent to non-note nodes only use one edge. The resulting graph resembles both note trees and outerplanar graphs in different regions, depending on the type of operation being used there. Thus, using ε nodes is an analytical option that reveals independencies between adjacent parts of the graph.

The graph grammar $\mathcal{G}^{rāga}$ is a special case of a graph grammar that is formally equivalent to a context-free grammar on strings of notes. Therefore, it can parse melodies efficiently. The context-free grammar can be obtained in two steps: First, the graph representation is transformed to an interval representation in all parts of the grammar. Second, a set of rules is added for generating notes from intervals by taking the second note of each interval and generating empty strings (ϵ) where necessary.

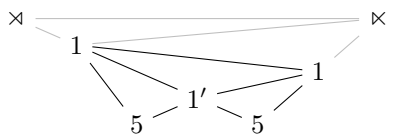


Figure 7: The spine modeling the deep structure of the octave expansion in an ālāp.

In a directed linear graph, edges are totally ordered by their direction, so the graph can be transformed into a sequence of edges (e.g., $a \to b \to c$ becomes $(a, b)(b, c)$). Let e be the function that transforms linear graphs to sequences of edges. Then the context-free melody grammar $G(\mathcal{G}_\mathcal{M}) := (T, N, I, R)$ induced by a melody-graph grammar $\mathcal{G}_\mathcal{M}$ is defined as follows:

$$
\begin{aligned}
T_G :=& P_M \\
N_G :=& (P_M \cup \{\rtimes, \varepsilon\}) \times (P_M \cup \{\varepsilon, \ltimes\}) \\
S_G :=& e(\mathcal{S}_\mathcal{G}) = (\rtimes, \ltimes) \\
R_G :=& \{e(l) \Rightarrow e(r) \mid l \Rightarrow r \in \mathcal{R}_\mathcal{G}\} \cup \\
& \{(x, p) \Rightarrow p \mid p \in P_M, x \in P_M \cup \{\rtimes, \ltimes, \varepsilon\}\} \cup \\
& \{(x, n) \Rightarrow \epsilon \mid n \in \{\varepsilon, \ltimes\}, x \in P_M \cup \{\rtimes, \ltimes, \varepsilon\}\}.
\end{aligned}
$$

5. DISCUSSION

A main motivation for introducing generalized neighbors is that they allow modelling leaps in the background structure of North Indian music. Figure 7 shows the architecture of a typical *ālāp* in rāga Multānī. The melody slowly ascends from 1 to $1'$ via 5 and returns back to 1 (again via 5). The upper $1'$ can be seen as a very stable and distant neighbor of 1 while the 5s in between are (again stable and distant) passing notes. Each stage of this *spine* is then further elaborated by neighbors and passing notes, using increasingly less stable pitches and smaller intervals (Figure 8).

North Indian music is not the only style of music in which melodies are based on hierarchical modes. If other mainly mode-based styles also follow the elaboration principles of passing notes and neighbors, then the grammar defined in Section 4 should permit sensible analyses for these cases. Consider, for example, the melody of *Nun komm der Heiden Heiland* (based on the Dorian mode) and a phrase from the Jazz standard *Moanin'* (based on a Blues scale). Their respective derivations (shown in Figure 9) suggest plausible reductions of the surface melody in both cases. Moreover, the proposed relations between notes match the intuitions of generalized neighbors and passing notes.

A natural generalization of the mode-based approach is to consider the mode as a latent variable that can change over the course of the piece but still organizes elaboration

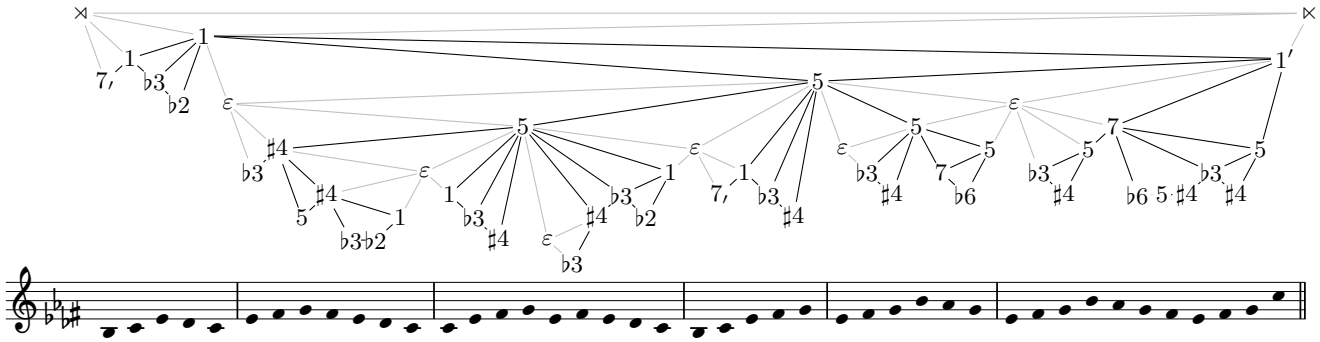


Figure 8: This example represents selected phrases, in order of performance, excerpted from the ascending part of an ālāp in rāga Multānī, recorded by the sitarist Dharambir Singh [22]. For reasons of space, one phrase has been selected for each of the stations between 1, 5 and 1′. Surface ornamentation and rhythmic durations have been omitted.

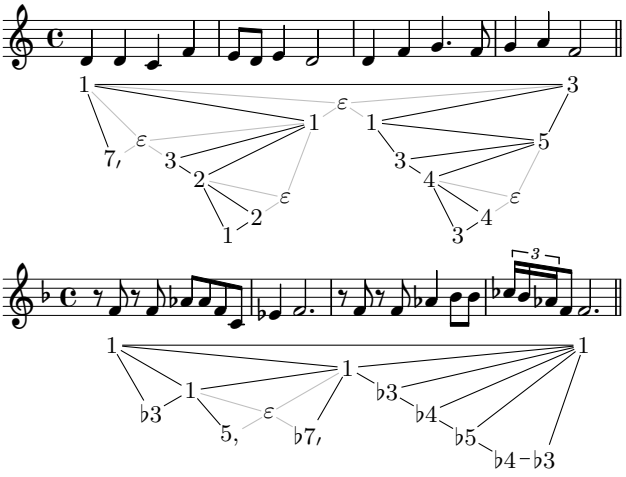


Figure 9: The first two phrases from *Nun komm der Heiden Heiland* and *Moanin'* (without repetitions), and their derivations based on a Dorian and a Blues scale, respectively.

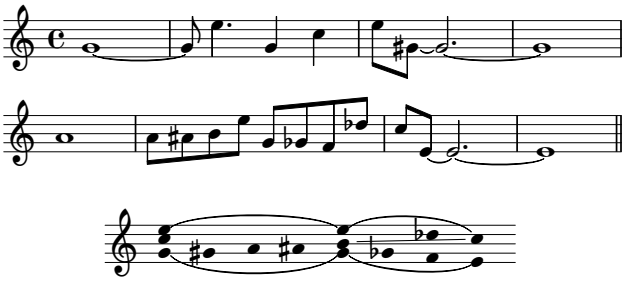


Figure 10: The A part of *Take the A-Train* and a summary of its underlying lines.

locally. Depending on the style, this hierarchy can be constant over longer regions of the piece or change rather frequently. The latter case occurs when harmonies are considered as the latent structure, as they also define a tonal hierarchy, ranging from the root to non-chord notes.

However, there are two issues concerning generalized neighbor elaboration on harmonies. First, when the latent hierarchy changes, it is not obviously clear what should happen at the transition point. This is not an issue when these transitions are rare and elaboration across these boundaries is avoided. However, when harmonies take the role of the latent hierarchy, then transitions occur more often and elaborations frequently cut across harmonic changes. Moreover, as melodic elaboration happens on every level of reduction, it can even be considered to *generate* harmonic change in the background, such as the passing $\hat{2}$ in the *Ursatz*, generating a V harmony.

Second, not all leaps in melodies can be explained as generalized neighbors. The melody of *Take the A-Train* (Figure 10), for example, features several leaps which cannot be consistently explained as neighbors. While the initial G_4 and E_5 might be seen as neighbors to C_5, the descent to E_4 in the end is left unexplained by that. Instead, it is more plausible to assume a set of several independent lines: A higher line descends from E_5 to C_5, a lower line from G_4 to E_4, and an intermediate line that connects G_4 and C_5. Internally these lines behave according to elaboration principles (passing notes in this case), but the surface melody freely switches between the lines. This suggests that the organizing latent structure in this case is a set of implicit lines, although the elaboration and coordination of these lines might still be governed by a mode or a harmonic sequence as another layer of latent structure.

6. CONCLUSION

This paper proposed a generalized graph grammar formalism to model North Indian rāga music. We propose that passing and neighbor note elaborations are both necessary and (in their generalized form) sufficient operations of recursive rāga melody. This strengthens their status as fundamental musical principles across cultures. As the two operations are based on different objects (intervals and notes), models of elaboration should be able to represent both notes and intervals explicitly.

The notion of a generalized neighbor, based on a tonal hierarchy, shows that melodic leaps do not happen arbitrarily but can be related to a latent background structure. Understanding and modelling this background structure is necessary for a deeper understanding of melodic elaboration.

7. ACKNOWLEDGEMENTS

The research presented in this paper is generously supported by the Volkswagen Foundation and Claude Latour. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 760081 – PMSB. We thank Dharambir Singh for the permission to use his recording of an ālāpin rāga Multānī. We also thank the anonymous reviewers for their helpful feedback.

8. REFERENCES

[1] M. Baroni, S. Maguire, and W. Drabkin. "The concept of musical grammar". In: *Music Analysis* 2.2 (1983), pp. 175–208.

[2] D. Clarke. "North Indian Classical Music and Lerdahl and Jackendoff's Generative Theory-a Mutual Regard." In: *Music Theory Online* 23.3 (2017).

[3] W. B. De Haas, M. Rohrmeier, R. C. Veltkamp, and F. Wiering. "Modeling harmonic similarity using a generative grammar of tonal harmony". In: *Proceedings of the Tenth International Conference on Music Information Retrieval (ISMIR)*. 2009.

[4] É. Gilbert and D. Conklin. "A Probabilistic Context-Free Grammar for Melodic Reduction". In: *International Workshop on Artificial Intelligence and Music, IJCAI-07*. 2007.

[5] M. Granroth-Wilding and M. Steedman. "A robust parser-interpreter for jazz chord sequences". In: *Journal of New Music Research* 43.4 (2014), pp. 355–374.

[6] R. Groves. "Towards the Generation of Melodic Structure". In: *Proceedings of the Fourth International Workshop on Musical Metacreation*. 2016, pp. 1–8.

[7] D. Harasim, M. Rohrmeier, and T. J. O'Donnell. "A Generalized Parsing Framework for Generative Models of Harmonic Syntax". In: *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*. Ed. by E. Gómez, X. Hu, E. Humphrey, and E. Benetos. 2018, pp. 152–159. ISBN: 978-2-9540351-2-3. URL: http://ismir2018.ircam.fr/doc/pdfs/258_Paper.pdf (visited on 04/12/2019).

[8] P. B. Kirlin and D. L. Thomas. "Extending a Model of Monophonic Hierarchical Music Analysis to Homophony". In: *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*. Ed. by M. Müller and F. Wiering. 2015, pp. 715–721. ISBN: 978-84-606-8853-2. URL: http://ismir2015.uma.es/articles/216_Paper.pdf (visited on 04/11/2019).

[9] P. B. Kirlin and J. Yust. "Analysis of Analysis: Using Machine Learning to Evaluate the Importance of Music Parameters for Schenkerian Analysis". In: *Journal of Mathematics and Music* 10.2 (May 3, 2016), pp. 127–148. ISSN: 1745-9737, 1745-9745. DOI: 10.1080/17459737.2016.1209588. URL: https://www.tandfonline.com/doi/full/10.1080/17459737.2016.1209588 (visited on 11/26/2018).

[10] P. B. Kirlin and P. E. Utgoff. "A Framework for Automated Schenkerian Analysis". In: (2008), p. 6.

[11] F. Lerdahl and R. S. Jackendoff. *A Generative Theory of Tonal Music*. MIT press, 1985.

[12] A. Marsden. "Representing Melodic Patterns as Networks of Elaborations". In: *Computers and the Humanities* 35.1 (Feb. 1, 2001), pp. 37–54. ISSN: 1572-8412. DOI: 10.1023/A:1002705506386. URL: https://doi.org/10.1023/A:1002705506386 (visited on 04/11/2019).

[13] A. A. Marsden. "Automatic Derivation of Musical Structure: A Tool for Research on Schenkerian Analysis." In: *International Conference on Music Information Retrieval*. 2007, pp. 55–58.

[14] S. Mukherji et al. "Generative Musical Grammar-A Minimalist Approach". In: *PhD Diss. Princeton University* (2014).

[15] E. Narmour. *The analysis and cognition of melodic complexity: The implication-realization model*. University of Chicago Press, 1992.

[16] M. Pearce and M. Rohrmeier. "Musical syntax II: empirical perspectives". In: *Springer handbook of systematic musicology*. Springer, 2018, pp. 487–505.

[17] H. S. Powers. "Language Models and Musical Analysis". In: *Ethnomusicology* 24.1 (1980), pp. 1–60. ISSN: 0014-1836. DOI: 10.2307/851308. URL: https://www.jstor.org/stable/851308 (visited on 04/11/2019).

[18] M. Rohrmeier. "Towards a generative syntax of tonal harmony". In: *Journal of Mathematics and Music* 5.1 (2011), pp. 35–53.

[19] M. Rohrmeier and M. Neuwirth. "Towards a syntax of the Classical cadence". In: *What is a Cadence?: Theoretical and Analytical Perspectives on Cadences in the Classical Repertoire*. Ed. by M. Neuwirth and P. Bergé. Leuven University Press, 2015, pp. 287–338.

[20] M. Rohrmeier and M. Pearce. "Musical syntax I: Theoretical perspectives". In: *Springer handbook of systematic musicology*. Springer, 2018, pp. 473–486.

[21] H. Schenker. *Der freie Satz Satz (=Neue musikalische Theorien und Phantasien 3)*. Ed. by J. Oswald. Vol. 3. Universal Edition, 1935.

[22] D. Singh. *Selected Alap Phrases in Raga Multani*. June 2019. DOI: 10.5281/zenodo.3258808. URL: https://doi.org/10.5281/zenodo.3258808.

[23] M. Steedman. "The blues and the abstract truth: Music and mental models". In: *Mental models in cognitive science*. 1996, pp. 305–318.

[24] M. J. Steedman. "A generative grammar for jazz chord sequences". In: *Music Perception* 2.1 (1984), pp. 52–77.

[25] J. Stock. "The Application of Schenkerian Analysis to Ethnomusicology: Problems and Possibilities". In: *Music Analysis* 12.2 (1993), pp. 215–240. ISSN: 0262-5245. DOI: 10.2307/854273. URL: https://www.jstor.org/stable/854273 (visited on 06/25/2019).

[26] R. Widdess. "Aspects of form in North Indian ālāp and dhrupad". In: *Music and tradition: essays presented to Laurence Picken*. Ed. by R. Widdess and R. Wolpert. 1981.

[27] J. Yust. "Voice-Leading Transformation and Generative Theories of Tonal Structure". In: *Music Theory Online* 21.4 (Dec. 1, 2015). URL: http://www.mtosmt.org/issues/mto.15.21.4/mto.15.21.4.yust.html (visited on 11/26/2018).

A COMPARATIVE STUDY OF NEURAL MODELS FOR POLYPHONIC MUSIC SEQUENCE TRANSDUCTION

Adrien Ycart* **Daniel Stoller*** **Emmanouil Benetos**
Centre for Digital Music, Queen Mary University of London, UK
{a.ycart,d.stoller,emmanouil.benetos}@qmul.ac.uk

ABSTRACT

Automatic transcription of polyphonic music remains a challenging task in the field of Music Information Retrieval. One under-investigated point is the post-processing of time-pitch posteriograms into binary piano rolls. In this study, we investigate this task using a variety of neural network models and training procedures. We introduce an adversarial framework, that we compare against more traditional training losses. We also propose the use of binary neuron outputs and compare them to the usual real-valued outputs in both training frameworks. This allows us to train networks directly using the F-measure as training objective. We evaluate these methods using two kinds of transduction networks and two different multi-pitch detection systems, and compare the results against baseline note-tracking methods on a dataset of classical piano music. Analysis of results indicates that (1) convolutional models improve results over baseline models, but no improvement is reported for recurrent models; (2) supervised losses are superior to adversarial ones; (3) binary neurons do not improve results; (4) cross-entropy loss results in better or equal performance compared to the F-measure loss.

1. INTRODUCTION

Automatic music transcription (AMT) is a core MIR problem [25]. Its aim is to extract what is played in a music signal into a human-readable score. Much of the literature has focused on the intermediate goal of detecting when and which notes were played, also called multi-pitch detection and note-tracking. It is a long-discussed topic, yet, unless it is constrained to a specific instrument and instrument model [17], it remains a challenging task, in particular in the case of polyphonic music [2].

Many AMT systems use a two-step workflow [3, 13]. First an *acoustic model* estimates the pitches present in each audio frame and produces a non-binary posteriogram representation over time and pitch. Then a post-processing step is applied to obtain a binary piano-roll representation, or a MIDI-like note-based representation (we focus here on piano-roll representation). The former task has been extensively discussed in the literature, but the latter has received much less attention.

Recent transcription and transduction models are commonly neural networks trained with a traditional maximum likelihood framework that assumes pitches to be independent [20, 23, 41]. Although this provides a simple and stable training loss, it can lead to unrealistic estimates as the model tries to cover all the modes of the output probability distribution [35]. More recently, Generative Adversarial Networks (GANs) [18] were proposed to rectify this issue, as they do not rely on an explicit likelihood and tend towards realistic outputs with high likelihood under the true output distribution. In GANs, a *generator* network makes its outputs as realistic as possible while a *discriminator* classifies examples as real (from the training set) or fake (created by the generator), which in turn gives feedback to improve the generator. GANs have made a breakthrough in high-quality image generation and reconstruction [8], and were subsequently also applied to symbolic music generation [39] and singing voice separation [36]. For AMT, GANs could help produce better transcriptions by taking into account correlations across both time and pitch.

The output of neural networks used in previous approaches [20, 23, 41] is real-valued to enable training by gradient descent.To obtain a piano roll at test time, the network outputs are binarised with a threshold as a post-processing step. However, this can lead to overly-fragmented notes, as in [41], when the output values are close to the threshold. Binary neurons [4] offer an alternative by integrating the binarisation of outputs into training while still allowing gradient back-propagation.

In this paper, we perform a comparative study of various neural-network-based methods for polyphonic sequence transduction, focusing on classical piano music. We systematically compare the influence of certain design choices, namely the network architecture, the training loss and the type of outputs. In particular, we introduce a GAN framework and assess its performance by comparing it to simpler training losses such as cross-entropy. We also evaluate whether binary neurons can bring improvements in this context as suggested for symbolic music generation [15] by making generator outputs binary and thus more similar to the real examples. In addition, we propose a method to directly use the F-measure as a training objective by using binary neurons. Each of these methods is evaluated with both a recurrent neural network (RNN) and a novel

*Authors 1 and 2 contributed equally to this work.

convolutional network as transduction model. These design choices are evaluated using inputs from two different multi-pitch detection systems (a piano-specific [23] and a multi-instrument acoustic model [5]). All experiments are conducted using a representation based on 16th note time steps, which can improve results [41] and reduce the input dimensionality compared to regular timesteps in the order of 10ms and thus the required training and test time.

Our paper is organised as follows. In Section 2, we describe previous work. Section 3 presents the dataset used. In Section 4, we detail the architecture used for our various sub-models, and the training objectives are described in Section 5. We present our evaluation metrics in Section 6, and describe our experiments and their results in Section 7. Finally, we discuss the limitations of our proposed methods along with future directions in Section 8.

2. RELATED WORK

2.1 Polyphonic Music Sequence Transduction

AMT acoustic models yield a posteriogram, which is a real-valued time-pitch representation that reflects the likelihood of pitches being present at different points in time, either as probabilities (between 0 and 1) or as saliences (unbounded). Polyphonic music sequence transduction aims to convert these posteriograms into a binary piano-roll representation (also called *note tracking*). Aside from simply thresholding the posteriogram, one of the most popular methods involves pitch-wise Viterbi decoding [31], where each pitch is considered as a 2-state (on/off) hidden Markov model (HMM).

More recently, a series of neural network-based methods has been proposed. In [6], a model combining a Restricted Boltzmann Machine (RBM) with a recurrent neural network (RNN) was proposed for symbolic music modelling. It was later used for AMT [7], using an audio representation from a Deep Belief Network as input. The same architecture was also used as a language model for piano-roll post-processing [34], to evaluate the likelihood of symbolic sequences rather than using it as a transduction model. In [41], a simpler Long Short-Term Memory (LSTM) model achieved improved performance when using musically-relevant time steps, but mostly because resulting outputs are quantized to the ground truth metrical grid.

2.2 Generative Adversarial Networks

Since their introduction in 2014, GANs have enjoyed considerable attention [18]. The main appeal in using GANs is that the model's likelihood only needs to be implicitly defined [29]. In contrast, defining a suitable and tractable likelihood function is difficult for complex output spaces such as the space of natural images, so simplistic assumptions are often made about the output density. In our context, a GAN could generate a coherent segment of music in one pass through the network, while other methods either have to assume independent outputs for each pitch and time-frame [17,20,23,41], or condition each output dimension on all previous ones in an auto-regressive fashion [34], which requires a computationally intensive decoding process.

Applications of GANs are increasing rapidly and cover a wide variety of fields [32, 36]. For music modelling, this includes the C-RNN-GAN [28], which uses an RNN to output note events with continuous duration, onset time, pitch and velocity, instead of the discrete values used in the MIDI representation. The MuseGAN [14] generates discrete piano-roll music representations. Here, a continuous-output generator is used and the training data is simply treated as being continuous despite its discrete nature.

GANs can also be used for conditional prediction models [26, 35, 36], where the output space is complex and high-dimensional and defining a suitable loss function is difficult. For these reasons we investigate GANs for AMT in this paper.

2.3 Binary neurons

Training a network with discrete outputs is challenging because back-propagation does not yield gradients from non-differentiable operations. Various methods have been proposed to solve that issue, such as REINFORCE [27,37], and the straight-through estimator [4, 21]. Binary neurons were used with GANs in the context of music generation [15], as it is very easy for the discriminator to separate binary, real samples from generated ones that have real values between 0 and 1. This can make generator training ineffective, as the discriminator's feedback focuses on only making the real-valued generator outputs more binary. They are reported as improving over a similar system with non-binary outputs [15] by being easier to train and yielding better results. More generally, binary outputs are appealing in our context, as they integrate the thresholding process into the network, removing the need for thresholding real-valued network outputs to obtain a binary piano-roll representation at test time and finding an optimal threshold value on an extra validation set.

3. DATASET

For our experiments, we use the MAPS dataset [16]. It contains MIDI files of polyphonic piano music, along with aligned audio renditions, generated using synthetic pianos and a Disklavier acoustic piano. It contains 238 pieces of classical music (18h total duration), with some pieces performed more than once, on different pianos. We split it into training, validation and test sets similarly to [23], where the test set features only acoustic piano and the training set only synthetic piano recordings, to obtain realistic performance estimates. We also ensure there is no overlap of pieces between the training and test set, and create a validation set by removing 10 random examples from the training set.

Rhythm annotations for this dataset are available in the A-MAPS annotations [40]. We use these annotations, in order to run our experiments using a time step of a 16th note. There are two main reasons for that choice. First, it was shown in [41] that using a 16th note time step can improve transcription performance over time steps of a fixed frame duration in seconds. Moreover, using 16th note time steps instead of fixed steps (usually on the order of 10 ms) results in a much more compact representation and faster training,

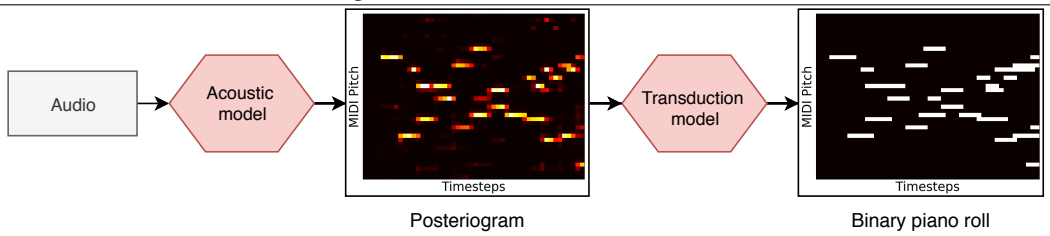


Figure 1. Overview of the whole transcription process. We focus on the second step.

enabling large-scale experiments. However, it introduces some imprecision when dealing with extra-metrical notes and tuples. In addition, the required beat positions are not usually available as annotations and would thus have to be obtained with a beat-tracking algorithm. Since we use beat annotations in our study, lower results than the ones reported here might be obtained in the presence of beat-tracking errors.

4. MODELS

The general workflow for our task is described in Figure 1. First an acoustic model produces a non-binary posteriogram. Then, this posteriogram is converted into a binary piano roll by a transduction model. Although this study focuses on the latter, we present both in this section.

4.1 Acoustic models

We use our transduction models with two different kinds of acoustic model outputs. The first model, described in [23], is a convolutional neural network (CNN) operating on spectrograms with logarithmically-spaced frequency bins and log-magnitude. It was designed specifically for piano transcription, and was trained on the MAPS dataset. At each time step, it outputs predictions for $N_p = 88$ pitches as probabilities between 0 and 1. We call this model *Kelz*.

The second model, described in [5], also uses CNNs, with a custom input representation that stacks harmonically-shifted Constant-Q transform representations of the signal. It was designed as a general multi-pitch detection system, and was trained on a multi-instrument dataset that includes piano. Besides, it has a frequency resolution of 20 cents, and has a smaller frequency span than a piano keyboard. In order to adapt it for piano transcription, we average the frequency bins that correspond to a given MIDI pitch. In this case, $N_p = 73$. It has to be noted that this averaging leads to lower activations. We call this model *Bittner*.

In both cases, outputs are then downsampled to a 16th note time step. To do so, we use the best-performing method in [41] referred to as *step*: let M and N be the original and downsampled posteriograms respectively, p a pitch, n a 16th note step, and i and j the time steps corresponding to n and $n+1$ respectively, we have:

$$s = i + round(\frac{j-i}{4}), \quad N[p,n] = \frac{\sum_{k=i}^{s} M[p,k]}{s-i+1}$$

4.2 Transduction model architectures

Our transduction models take the output of an acoustic model as input and are trained independently from the acoustic models. We compare two different kinds of transduction models. The first model is a Long Short-Term Memory (LSTM) network [22] with the same architecture as described in [41]. It has N_p inputs per time step, one hidden layer with 100 hidden nodes, followed by a fully connected layer with N_p outputs.

The second model is a newly-designed CNN that uses a series of convolutions with filter sizes (1) 5×5, (2) 5×1 with dilation of 12×1, and (3) 5×15, to capture both frequency and temporal correlations. A 1×1 convolution processes the combined output from convolutions (2) and (3), whose output is combined with the output from convolution (1) using another 1×1 convolution. While all above convolutions have 32 channels and use LeakyReLU activations, the final 1×1 convolution has only one filter and does not use an activation. Both models are closely matched in terms of the number of parameters at about $80,000$.

4.3 Real-valued vs. Binary outputs

We use two different strategies to convert the unnormalised, real-valued outputs obtained from the model. The first one is real-valued sigmoid outputs: the output of the network for each pitch is simply sent through a sigmoid. At test time, the outputs are binarised using a single threshold for all pitches, chosen to maximise frame-wise $\mathcal{F}$ on the validation set. The second strategy employs deterministic binary neurons, as described in [14]. Here, the output of the network for each pitch is sent through a sigmoid, and then thresholded at 0.5. This thresholding operation, however, is not differentiable, which makes it impossible to use gradient descent as training method. Therefore, we use the sigmoid-adjusted straight-through estimator [12], that was shown to provide good results [14]. It ignores the thresholding operation during back-propagation and instead multiplies the current gradient by the derivative of the sigmoid function.

4.4 Baseline methods

We compare our system against two baseline post-processing methods. The first binarises all outputs using a threshold optimised as explained in Section 4.3 (as opposed to using a fixed threshold of 0.5 as done in [23]). The second one, introduced in [31], and later used in various systems [10, 11], is to model each pitch by an on/off HMM, and then decode the most likely sequence of hidden states using the Viterbi algorithm. The initial probabilities and

transition parameters are computed from ground truth annotations, one set of parameters per pitch. The observations for 'on' and 'off' states are modelled as beta distributions, each fitted to the acoustic model outputs on the validation dataset. These distributions are fitted using data from all pitches and shared between them due to the rarity of extremely low and high pitches.

5. TRAINING OBJECTIVES

The networks are trained using the Adam optimiser [24] on sequences of 64 timesteps. For cross entropy and F-measure, we use a learning rate of 10^{-2}, as advised in [41]. With GANs, the learning rate is 10^{-3} for the discriminator, and $5 \cdot 10^{-4}$ for the generator. Early stopping is used: after no improvement in validation framewise $\mathcal{F}$ for 30,000 iterations (checked every 1000 iterations), training is stopped and the best model is kept.

5.1 Cross-entropy loss

As a commonly used baseline objective, we use the binary cross-entropy (CE) loss averaged over all entries $\hat{y}_t$ in the predicted piano-roll $\hat{y}$ as loss for the transducer model:

$$\mathbb{E}_{(x,y)\sim\mathbb{P}_d} \frac{1}{N_t N_p} \sum_{p=1}^{N_p} \sum_{t=1}^{N_t} y_{t,p} \log \hat{y}_{t,p} + (1 - y_{t,p}) \log(1 - \hat{y}_{t,p}) \tag{1}$$

where $y_{t,p} \in \{0, 1\}$ indicates whether a note at timestep $t \in [\![1, N_t]\!]$ with pitch index $p \in [\![1, N_p]\!]$ is active, and $\mathbb{P}_d$ denotes the joint distribution of acoustic model outputs and transcriptions. While this loss is conceptually simple and allows fast and stable training, it relies on the strong assumption that each note activity can be determined independently from all others, although some note combinations are more likely to occur than others. Since the model cannot assign high probability to specific note combinations, only to individual notes, it would assign high probability to all involved notes. Unrealistic note combinations might then be obtained when independently sampling from these probabilities.

5.2 F-measure loss

In most cases, transduction and transcription models are trained with CE, but finally evaluated with frame- or note-wise F-measure. To close this gap between training and testing, we propose maximising the F-measure directly during training. However, this requires a binary piano roll instead of real-valued network outputs. As a solution, we employ binary neurons introduced in Section 4.3. By replacing $\mathcal{P}$ and $\mathcal{R}$ in the formula for $\mathcal{F}$ by their complete expressions (see Section 6), simplifying the equation, and using the two identities $\mathrm{TP}(t) = \sum_{p=1}^{N_p} y_{t,p} \cdot \hat{y}_{t,p}$ and $\mathrm{FN}(t) + \mathrm{FP}(t) = \sum_{p=1}^{N_p} |y_{t,p} - \hat{y}_{t,p}|$ where $\mathrm{TP}(t)$, $\mathrm{FP}(t)$ and $\mathrm{FN}(t)$ are the numbers of true positives, false positives and false negatives at time step t, respectively, we obtain

$$\mathcal{F} = \frac{\sum_{t=1}^{N_t} \sum_{p=1}^{N_p} 2 \cdot y_{t,p} \cdot \hat{y}_{t,p}}{\sum_{t=1}^{N_t} \sum_{p=1}^{N_p} \left[2 \cdot y_{t,p} \cdot \hat{y}_{t,p} + |y_{t,p} - \hat{y}_{t,p}|\right]} \tag{2}$$

as our F-measure loss, which only makes use of differentiable operators. The only exception is the absolute value function, whose gradient is undefined only at 0, which occurs in the above equation if $y_{t,p} = \hat{y}_{t,p}$. Since the model output $\hat{y}_{t,p}$ does not need to change in that case, we assign it a gradient of 0. In conjunction with binary outputs, we can thus use gradient descent to maximise $\mathcal{F}$. It has to be noted that a similar, non-binary version of F-measure was proposed as training loss in [30]. We only investigate the binary F-measure here.

5.3 Adversarial loss

We adapt the improved Wasserstein GAN [19] framework for the adversarial loss to our conditional generation case, by using a discriminator network $D_\theta : x, y \to \mathbb{R}$ that takes a transcription y as input with the corresponding acoustic model output x as condition. It is trained to output scalar values that minimise the loss (see Equation (3) in [19]):

$$\begin{aligned} L(\theta) = \mathbb{E}_{x\sim\mathbb{P}_r}[D_\theta(x, G_\phi(x))] - \mathbb{E}_{x,y\sim\mathbb{P}_d}[D_\theta(x, y)] \\ + \lambda\, \mathbb{E}_{x\sim\mathbb{P}_r, \hat{y}\sim\mathbb{P}_i(x)}[(||\nabla_{\hat{y}} D_\theta(x, \hat{y})||_2 - 1)^2], \end{aligned} \tag{3}$$

where $\mathbb{P}_r$ denotes the distribution of real acoustic model outputs. The third term regularises the discriminator network responses to stabilise generator training. It is weighted by the scalar hyper-parameter λ (we use $\lambda = 10$), and involves drawing $\mathbb{P}_i(x)$, which yields $\hat{y} = r \cdot G_\phi(x) + (1 - r) \cdot y$ with a weight r uniformly chosen from $[0, 1]$ that interpolates between the generator output $G_\phi(x)$ and the ground truth transcription y for the input x.

During training, the generator is updated once using the negative of the first term in (3) as loss, before the discriminator is trained on (3) for 5 iterations to re-estimate the Wasserstein distance between real and generator samples.

As discriminator, we use a convolutional network similar to [42]. First, a 5×1 convolution with dilation of 12 and 16 channels is computed to capture relationships between octaves, and concatenated to the input features for further processing. Afterwards, a 3×3 convolution with a stride of two is applied four times, using 16, 32, 64, and 128 filters, respectively. The downsampled feature map is processed by a dense layer with 16 nodes, and by another dense layer with a single output and linear activation. All layers except the final one use LeakyReLU activations.

6. EVALUATION METRICS

We evaluate the performance of our system using the common MIREX transcription metrics [1], in both frame-wise and note-wise configurations. With *frame metrics*, the output and the ground truth are compared frame-by-frame. The metrics are computed directly on the 16th-note-timestep piano-rolls. With *note metrics*, the system output and the ground truth are viewed as a list of notes. A note is correctly detected if its pitch matches the ground truth and its onset is within a tolerance threshold of the correct onset. The threshold usually used is 50ms, however, since we operate on 16th note timesteps, we require that onset times correspond exactly to the ground truth. The offsets are usually

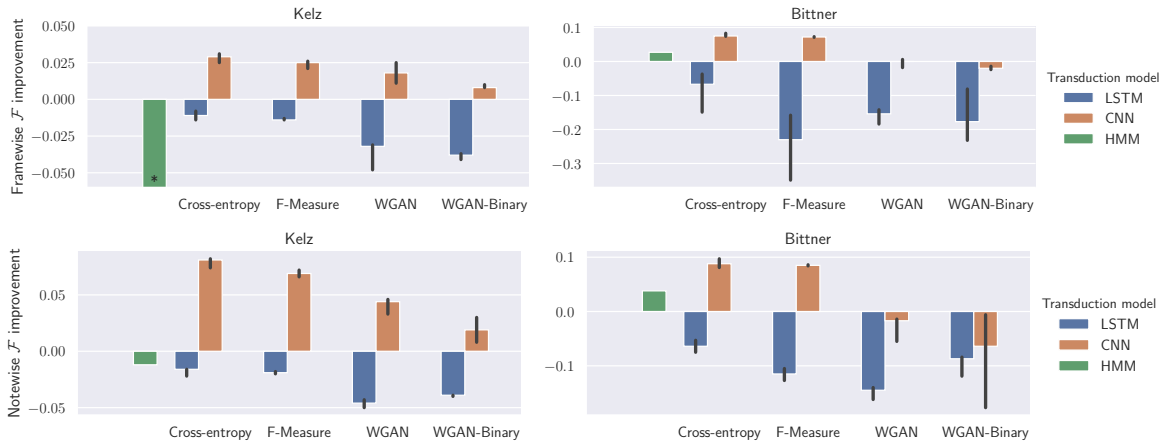


Figure 2. Model comparison across training objectives, transducer and acoustic models. Bars correspond to median improvement over simple thresholding, error bars correspond to maximum and minimum improvement across runs. * means that the bar was truncated for readability purposes.

not taken into account as they are difficult to estimate properly with percussive sounds such as piano notes. We use the `mir_eval` implementation [33] to find the maximal match between the two lists. In both cases, the precision ($\mathcal{P}$), recall ($\mathcal{R}$) and F-measure ($\mathcal{F}$) are computed as per [1]. These metrics are computed for each recording, and then averaged over sets of recordings.

7. RESULTS

We evaluate every possible combination of acoustic and transduction model as well as training loss and output type (continuous or binary neurons). To account for variability in training, we run every experiment 3 times and report results of the model that reaches the median frame $\mathcal{F}$ performance. To assess differences in performance across different conditions, we perform paired Wilcoxon signed-rank tests using the piece-wise results from each median model. All median results are given in Table 1 for Kelz, and Table 2 for Bittner. We also plot the absolute improvement over simple thresholding for frame and note $\mathcal{F}$ in Fig. 2. The error bars show the best and worst of the 3 runs.

7.1 Transduction model and acoustic model

The two transduction models (CNN and LSTM) clearly behave very differently. Most of the time, the CNN model improves the results compared to simple thresholding. On average, across all configurations, the CNN provides an average increase of 2.56% frame-wise and 3.39% note-wise $\mathcal{F}$. In particular, the CE-trained CNN improves for Kelz by 3.1% frame and 8.2% note $\mathcal{F}$, while for Bittner, it reaches 8.3% frame and 9.7% note $\mathcal{F}$. This difference in improvement can be explained by the fact that the Kelz acoustic model overfits a lot on the training set, reaching around 92% frame $\mathcal{F}$ on the training set alone, but only around 81% on the validation set. As a result, it is more difficult for transduction models to generalise with the Kelz model as input, as training and test data differ. The Bittner model is much more stable across subsets with 62% frame $\mathcal{F}$ on the training set, and 63% on the validation set, as it was trained on a completely separate dataset.

On the other hand, the LSTM model performs worse than thresholding in every configuration. This contradicts the improvement in frame $\mathcal{F}$ reported in [41] achieved using an LSTM, although one could argue that the experimental setup is quite different regarding the LSTM inputs, the lengths of training sequences, and dataset splits. We tried using training sequences of 256 instead of 64 timesteps and also tried a bi-directional LSTM to check whether a lack of context explains our result, but results did not improve. Since we observe much better training set than test set performance (e.g. the LSTM with CE loss on the Bittner model achieves about 80% on training but only 57% frame $\mathcal{F}$ on the test set), we suspect that strong overfitting is the sole reason. This effect might be amplified as training and test sets do not share any pieces or pianos, and synthetic pianos are used for training and real pianos for testing. Since the partitions used in [41] do share pianos, the improvement reported therein could be partly due to overfitting to the particular pianos featured in the training set. Our CNN appears to generalise better in this regard.

7.2 Comparing training objectives

Since our LSTM model does not generalise to the test set, we exclude it from the following comparisons between training objectives. It appears that supervised losses consistently outperform adversarial losses ($p < 10^{-9}$ for frame and note $\mathcal{F}$). Additionally, GANs performance varies more between training runs than with supervised losses. This contradicts the expected result that GANs should be more effective. Using binary neurons also failed to improve the performance of our system. In particular, in the context of GANs, non-binary neurons are significantly better than binary neurons for both frame and note $\mathcal{F}$ ($p < 0.02$), except for Bittner in terms of note $\mathcal{F}$ ($p = 0.33$). This shows that the findings of [14] do not generalise to our application, at least when we compare with strong CE-based models. GANs might require more data to realise their potential, better ways of training with discrete outputs, or more perceptually meaningful evaluation metrics to capture the improvement in

Metric		Thresh	HMM	Cross-entropy		F-measure		WGAN		WGAN-Binary	
				LSTM	CNN	LSTM	CNN	LSTM	CNN	LSTM	CNN
Frame	$\mathcal{F}$	67.9	49.6	66.8	**70.8**	66.5	70.4	64.7	69.7	64.1	68.7
	$\mathcal{P}$	70.9	**74.1**	72.6	73.4	70.2	72.2	72.5	**74.1**	74.4	73.9
	$\mathcal{R}$	66.7	40.1	63.2	69.6	64.6	**70.1**	59.8	67.2	57.8	65.5
Note	$\mathcal{F}$	45.0	43.8	43.4	**53.2**	43.1	51.6	40.4	49.4	41.1	46.9
	$\mathcal{P}$	44.0	**82.4**	42.8	50.9	39.3	50.7	39.5	50.1	40.5	44.6
	$\mathcal{R}$	47.5	31.3	45.6	**57.1**	49.4	53.9	43.0	50.0	43.6	51.0

Table 1. Results as percentages for median models with Kelz input. Bold corresponds to best values across models.

Metric		Thresh	HMM	Cross-entropy		F-measure		WGAN		WGAN-Binary	
				LSTM	CNN	LSTM	CNN	LSTM	CNN	LSTM	CNN
Frame	$\mathcal{F}$	58.8	61.5	52.1	**66.3**	35.8	66.0	43.4	58.8	41.1	56.8
	$\mathcal{P}$	59.6	52.6	48.0	68.5	26.9	**69.3**	43.9	58.7	35.9	61.9
	$\mathcal{R}$	61.5	**79.6**	60.5	66.1	65.4	65.3	47.7	60.9	50.8	56.4
Note	$\mathcal{F}$	44.6	48.4	39.3	**53.4**	31.9	53.1	30.1	43.2	36.2	44.0
	$\mathcal{P}$	42.2	**62.5**	35.7	49.3	25.2	50.7	27.6	40.8	34.4	44.8
	$\mathcal{R}$	48.6	40.4	45.1	**59.9**	45.6	57.3	34.5	47.2	39.2	44.5

Table 2. Results as percentages for median models with Bittner input. Bold corresponds to best values across models.

output quality. Additionally, improvements or alternatives to binary neurons might be needed to stabilise training further, as we observed high variance in training loss between batches compared to using a CE loss.

We find that using our F-measure loss results in slightly worse $\mathcal{F}$ than when using the CE loss with Kelz (decrease of 0.4 in frame- and 1.6 in note-wise $\mathcal{F}$, $p < 0.02$), but no significant differences with Bittner. This might be explained by the use of 16th note timesteps that strongly smooths the inputs and outputs, reducing the risk of fragmented notes substantially. Also, the binary outputs could make training more difficult due to the approximations used to overcome the non-differentiability problem; and because we perform threshold tuning on the validation dataset for the CE setting, thereby giving the CE model the chance to fit its parameters more to the validation set, whereas it is only used for early stopping with the F-measure loss.

The theoretical advantages of binary neurons do not translate into performance improvements, but they are conceptually elegant since they do not require a separate threshold tuning after training. Overall, the basic CE loss remains the best performing for our task.

7.3 Comparison against baseline

We compare the best-performing model on both inputs (i.e. CNN trained with CE loss) with both baseline systems. CNN post-processing improves substantially over both baselines, for both acoustic models, with both frame and note $\mathcal{F}$ ($p < 0.01$). Surprisingly, the HMM post-processing yields significantly worse results than simple thresholding on Kelz for frame $\mathcal{F}$, but not significantly for note $\mathcal{F}$. It seems to be too conservative, outputting a note only when it is long enough and with high enough activation, a problem already noted in [41]. On Bittner however, HMM post-processing significantly improves results over thresholding ($p < 10^{-9}$ for frame $\mathcal{F}$, $p = 0.08$ for note $\mathcal{F}$). This can be attributed to the fact that the thresholded outputs contain a particularly high proportion of fragmented notes, because activations are very low in general, with many values around the threshold. The HMM successfully smooths them.

8. CONCLUSION

In this study, we proved that post-processing a posteriogram with a convolutional network model can improve transcription performance compared to several baseline methods. Various other tasks include similar discretisation problems, such as polyphonic sound event detection [9] or automatic drums transcription [38], and could probably benefit from this finding. We showed that some theoretically-motivated approaches did not actually result in increased performance as evaluated by the F-measure: WGANs do not perform better than a simple CE loss, binary neurons do not improve WGANs for this task, and training directly with the F-measure as loss is not better than using CE and then thresholding outputs as a post-processing step. We also showed that the results obtained in [41] actually do not generalise to the case where the recording conditions of test and training datasets differ, highlighting the importance of carefully selecting dataset partitions in future work.

This study resulted in many unexpected results that prompt us to investigate further. More transduction architectures could be investigated, such as C-RNNs. Data augmentation (e.g. pitch transposition) could provide sufficient data to make GANs worthwhile over CE. The F-measure loss could also improve further with other methods for back-propagating through the binary neurons. The fact that binary neurons seemed to improve results with Kelz and the LSTM model is also a good motivation to keep exploring their use with various architectures. Finally, all these configurations were evaluated with standard metrics. Especially for the WGAN models, it would be interesting to see whether the performance increases in ways that are not captured by the F-measure metric but turn out to be perceptually important in listening tests.

9. ACKNOWLEDGEMENTS

A. Ycart is supported by a QMUL EECS Research Studentship. D. Stoller is funded by EPSRC grant EP/L01632X/1. E. Benetos is supported by UK RAEng Research Fellowship RF/128.

10. REFERENCES

[1] Mert Bay, Andreas F. Ehmann, and J. Stephen Downie. Evaluation of Multiple-F0 Estimation and Tracking Systems. In *10th International Society for Music Information Retrieval Conference (ISMIR)*, pages 315–320, 2009.

[2] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[3] Emmanouil Benetos and Tillman Weyde. An efficient temporally-constrained probabilistic model for multiple-instrument music transcription. In *16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 701–707, 2015.

[4] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[5] Rachel M Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan Pablo Bello. Deep salience representations for f0 estimation in polyphonic music. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 63–70, 2017.

[6] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. *29th International Conference on Machine Learning*, pages 1159–1166, 2012.

[7] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. High-dimensional Sequence Transduction. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3178–3182, 2013.

[8] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

[9] Emre Cakır, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1291–1303, 2017.

[10] F.J. Cañadas Quesada, N. Ruiz Reyes, P. Vera Candeas, J.J. Carabias, and S. Maldonado. A multiple-f0 estimation approach based on gaussian spectral modelling for polyphonic music transcription. *Journal of New Music Research*, 39(1):93–107, 2010.

[11] Dorian Cazau, Yuancheng Wang, Olivier Adam, Qiao Wang, and Gregory Nuel. Improving note segmentation in automatic piano music transcription systems with a two-state pitch-wise hmm method. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 523–530, 2017.

[12] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical Multiscale Recurrent Neural Networks. *arXiv preprint arXiv:1609.01704*, 2016.

[13] Andrea Cogliati, Zhiyao Duan, and Brendt Wohlberg. Piano transcription with convolutional sparse lateral inhibition. *IEEE Signal Processing Letters*, 24(4):392–396, 2017.

[14] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment. In *AAAI Conference on Artificial Intelligence*, pages 34–41, 2018.

[15] Hao-Wen Dong and Yi-Hsuan Yang. Convolutional Generative Adversarial Networks with Binary Neurons for Polyphonic Music Generation. In *19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 190–196, 2018.

[16] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010.

[17] Sebastian Ewert and Mark B. Sandler. An augmented Lagrangian method for piano transcription using equal loudness thresholding and LSTM-based decoding. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA, 2017.

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[19] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved Training of Wasserstein GANs. *CoRR*, abs/1704.00028, 2017.

[20] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. *19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[21] Geoffrey Hinton, Nitsh Srivastava, and Kevin Swersky. Neural networks for machine learning. *Coursera, video lectures*, 264, 2012.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[23] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Bock, Andreas Arzt, and Gerhard Widmer. On the Potential of Simple Framewise Approaches to Piano Transcription. *17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 475–481, 2016.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015.

[25] Anssi Klapuri. Introduction to music transcription. In *Signal processing methods for music transcription*, pages 3–20. Springer, 2006.

[26] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[27] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*, 2014.

[28] Olof Mogren. C-RNN-GAN: A continuous recurrent neural network with adversarial training. In *Constructive Machine Learning Workshop (CML) at NIPS 2016*, 2016.

[29] Shakir Mohamed and Balaji Lakshminarayanan. Learning in Implicit Generative Models. In *5th International Conference on Learning Representations (ICLR)*, 2017.

[30] Joan Pastor-Pellicer, Francisco Zamora-Martínez, Salvador España-Boquera, and María José Castro-Bleda. F-measure as the error function to train neural networks. In *International Work-Conference on Artificial Neural Networks*, pages 376–384. Springer, 2013.

[31] Graham E. Poliner and Daniel P. W. Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, 5(1):154–162, Oct 2006.

[32] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *CoRR*, abs/1511.06434, 2015.

[33] Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Dan P. W. Ellis. mir_eval: A transparent implementation of common MIR metrics. In *15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[34] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):927–939, May 2016.

[35] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. In *5th International Conference on Learning Representations (ICLR)*, 2017.

[36] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Adversarial Semi-Supervised Audio Source Separation applied to Singing Voice Extraction. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2391–2395, Calgary, Canada, 2018. IEEE.

[37] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[38] Chih-Wei Wu, Christian Dittmar, Carl Southall, Richard Vogl, Gerhard Widmer, Jason Hockman, Meinard Muller, and Alexander Lerch. A review of automatic drum transcription. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 26(9):1457–1483, 2018.

[39] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[40] Adrien Ycart and Emmanouil Benetos. A-MAPS: Augmented MAPS dataset with rhythm and key annotations. In *ISMIR Late Breaking and Demos Papers*, 2018.

[41] Adrien Ycart and Emmanouil Benetos. Polyphonic Music Sequence Transduction with Meter-Constrained LSTM Networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018.

[42] Yang Yu, Zhiqiang Gong, Ping Zhong, and Jiaxin Shan. Unsupervised representation learning with deep convolutional neural network for remote sensing images. In *International Conference on Image and Graphics*, pages 97–108. Springer, 2017.

LEARNING SIMILARITY METRICS FOR MELODY RETRIEVAL

Folgert Karsdorp[1] **Peter van Kranenburg**[1,2] **Enrique Manjavacas**[3]

[1] KNAW Meertens Instituut, The Netherlands, [2] Utrecht University, The Netherlands
[3] University of Antwerp, Belgium

folgert.karsdorp@meertens.knaw.nl

ABSTRACT

Similarity measures are indispensable in music information retrieval. In recent years, various proposals have been made for measuring melodic similarity in symbolically encoded scores. Many of these approaches are ultimately based on a dynamic programming approach such as sequence alignment or edit distance, which has various drawbacks. First, the similarity scores are not necessarily metrics and are not directly comparable. Second, the algorithms are mostly first-order and of quadratic time-complexity, and finally, the features and weights need to be defined precisely. We propose an alternative approach which employs deep neural networks for end-to-end similarity metric learning. We contrast and compare different recurrent neural architectures (LSTM and GRU) for representing symbolic melodies as continuous vectors, and demonstrate how duplet and triplet loss functions can be employed to learn compact distributional representations of symbolic music in an induced melody space. This approach is contrasted with an alignment-based approach. We present results for the Meertens Tune Collections, which consists of a large number of vocal and instrumental monophonic pieces from Dutch musical sources, spanning five centuries, and demonstrate the robustness of the learned similarity metrics.

1. INTRODUCTION

The question of how melodic similarity can be computationally modeled is of crucial importance for various Music Information Retrieval (MIR) tasks [35]. One classic MIR scenario is a user posing a sung or hummed query to a retrieval system in order to retrieve resembling pieces of music from a music collection [6, 24]. This query-by-humming scenario requires melodic matching methods that are robust against different kinds of melodic variation arising from imprecise memory or limited singing skills. Melody matching is also an important aspect in cover-song detection, where the predominant melody contains information for song identification [29]. Musicologists benefit from melodic similarity measures for exploring and mapping folk songs [19, 30], or other collections with monophonic musical material, such as themes from classical compositions [18], or score incipits [34]. Finally, music similarity detection plays an important role in cases of copyright violation [31], where objective similarity measures can support the court in making decisions.

In this paper, we present a Neural Network approach for melodic similarity learning. The neural encoders learn complex mappings from input sequences into distributed melody representations. The primary aim of our system is to be employable for music retrieval. In addition to accurately modeling melodic similarity, desirable properties of a retrieval system are speed and indexability. Neural Networks very well accommodate both. Generally, once trained, a neural network can compute results very fast by making use of GPUs. Moreover, indexability is served by the application of similarity metrics on top of the learned encodings [3].

Various other approaches to melodic similarity have been taken [35], including sequence alignment and other dynamic programming approaches, such as edit distance and dynamic time warping, and, recently, Recurrent Neural network representations [4, 9, 11, 16, 27, 30, 33]. In this study, we contrast our approach with a sequence alignment method that has been successfully applied in the context of folk melodies. Alignment-based methods suffer from at least three drawbacks. First, they typically are first-order, taking into account only adjacent items in sequences. Second, they have quadratic time-complexity. Finally, an alignment score is not a proper metric. Our neural network approach overcomes these disadvantages but does so at the cost of being a supervised learning algorithm.

2. DATA AND FEATURES

2.1 Data sets

The Meertens Tune Collections (MTC) [1] contains a series of data sets with melodic material from Dutch sources (mainly manuscripts, printed sources, and audio recordings), spanning five centuries of music history [20, 22]. These data sets are subsets of the Dutch Song Database, maintained by the KNAW Meertens Institute [21]. The lat-

[1] http://www.liederenbank.nl/mtc/

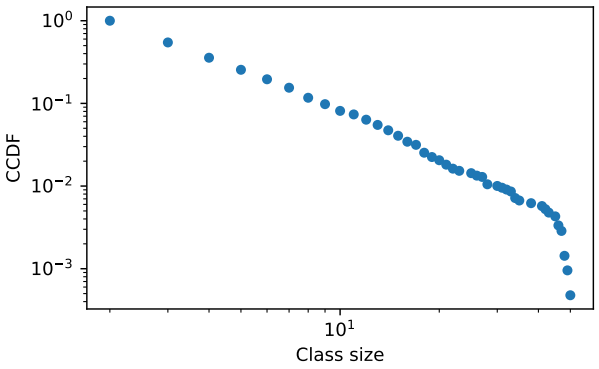


Figure 1. Complementary Cumulative Distribution Function of the tune family sizes in the subset of MTC-FS-INST 2.0 which we use in our experiments.

est release, MTC-FS-INST 2.0, contains 18,109 digitized melodies with rich metadata. Many of these melodies occur in more than one source. Due to oral and semi-oral transmission, these different occurrences typically show melodic variation. As an example, Figure 3 depicts three variant melodies, illustrating the kind and extent of variation. To denote such a group of variant melodies, we adopt the concept of *tune family* from folk song research [1]. In a long-term effort, the collection specialists of the Meertens Institute aim to identify each melody in terms of tune family membership.

In this paper, we use MTC-FS-INST 2.0, which reflects the diversity of the contents of the Dutch Song Database. One main distinction in the data set is between vocal and instrumental music as illustrated in Figure 2. Generally, the instrumental part of the data set dates from the 17th and 18th centuries. It contains melodies that were played in bars and brothels as well as theaters and upper-class private settings. The vocal part of the data set mainly consists of songs from the 19th and 20th centuries. As a whole, the data set provides a rich variety in melodic styles, which renders it a perfect source for training general purpose melodic similarity measures.

To obtain training, development, and test sets, we filter and split the data set. First, we exclude all 5,765 unlabeled melodies and all 3,008 singleton tune families. This leaves a selection of 9,336 melodies in 2,094 tune families. The complementary cumulative distribution function of the class sizes is presented in Figure 1. The distribution of class sizes is heavy-tailed.

An important criterion to measure the level of success achieved by the metric learning approach is its capability to cluster together tune melodies belonging to families unseen during training. In order to make this possible, we perform a controlled test set split, ensuring that all instances from a proportion of tune families do not appear in the training data. The actual proportions of seen and unseen families is shown in Table 1 together with further data set size statistics. [2]

[2] Supplementary material, data sets and code to replicate the experiments are available from `https://github.com/fbkarsdorp/melodic-similarity`.

	# Mel	# TF	# TF in Train	$\mu_{\|TF\|}$	$\sigma_{\|TF\|}$
Train	5,975	1,572		3.80	5.06
Dev	1,492	495	255	3.01	1.64
Test	1,869	611	287	3.06	1.55

Table 1. Composition of the subsets of MTC-FS-INST 2.0 used for training, development and testing. The table provides the number of melodies (Mel) and tune families (TF) in each set, the number of tune families that are shared with the training set, and mean and standard deviation of tune family sizes.

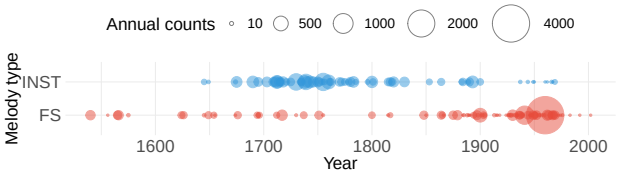


Figure 2. Number of melodies per year in MTC-FS-INST 2.0. The plot displays frequencies for instrumental melodies (INST) and vocal melodies (FS).

2.2 Features

Melodies are represented as sequences of notes, and notes as sets of feature-values. Since the MTC provide a rich melody encoding, including key, meter, and phrase boundaries, we can assemble a diverse feature-set in which various musical parameters are represented: pitch, metric structure, rhythm, tonality, and phrase structure. See the supplementary material for an exact list of features.

3. METHODOLOGY

Our approach is based on two components. First, we deploy distributional melody encoders implemented with Neural Networks. Secondly, we train the encoders with Stochastic Gradient Descent to minimize a Contrastive Loss that we describe below.

3.1 Distributional Encoder

An input melody from the dataset $x_i \in X$ can be represented by a sequence $x_i = [x_{(i,1)}, \ldots, x_{(i,k)}]$ of length $k = |x_i|$, where each $x_{(i,t)}$ is a bundle of m features explained in Section 2.2. For simplicity, we refer to the j^{th} feature of sequence step t as x_t^j, thus dropping data set indices. Our goal is to compute an encoding $h = f(x)$ as a function of the input sequence x, parameterized by a Neural Network $f(x)$.

The encoding process can be described as follows. We first process each time step in the input sequence independently by concatenating all features into a single vector $e_t = [e_t^1; \ldots; e_t^m]$. Categorical features are first encoded into a one-hot vector and projected into their own embedding space with model parameters $W_j \in \mathbf{R}^{JxE}$, where J

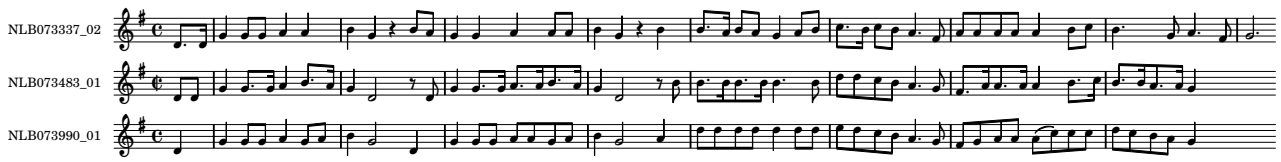


Figure 3. Three members of tune family *Daar was laatstmaal een ruiter 2*, showing various kinds of melodic variation.

is the total number of possible values of the j^{th} categorical feature and E is the dimensionality of the embedding space. Continuous features are normalized to have a mean of 0 and standard deviation of 1. For all feature types, the sequences are padded at the beginning and the end using special symbols in the case of categorical features and the feature mean after re-scaling for continuous features. The resulting sequence of input embeddings is fed to a stack of recurrent layers [3] with the t^{th} hidden activation at layer l given by $h_{(t,l)} = RNN_l(h_{(t,l-1)}, h_{(t-1,l)})$.

We also experiment with bidirectional RNNs, which extend each RNN layer with an additional RNN run backwards. In the case of the bidirectional RNN, the final melody embedding is given by the concatenation of the last activations of the forward and backward RNNs at the last layer: $h = [\overrightarrow{h_{(k,|L|)}}; \overleftarrow{h_{(1,|L|)}}]$. In the case of the unidirectional RNN, the embedding is given by a feature-wise max-pooling operation over the sequence of activations at the last layer, with the p^{th} output feature defined by $h_p = max([h_{(1,|L|)}]_p, \ldots, [h_{(k,|L|)}]_p)$. Instead of traditional RNN cells [7], we use LSTM [14] and GRU [5] cells which have been shown to offer stronger performance and better training behavior.

3.2 Contrastive Loss

The goal of our approach is to learn a distributional encoder such that melodic sequences of the same class are embedded into neighboring regions and far from melodic sequences belonging to different families. To this end, we train the encoder using a contrastive loss [12].

3.2.1 Duplet Loss

Let the encodings of two input sequences with tune family labels y_i and y_j be denoted by x_i and x_j. The goal we want to achieve is that the similarity between x_i and x_j is high when y_i and y_j are equal, and low otherwise. More formally, we seek to achieve the following inequality:

$$D(x_i, x_j) < D(x_i, x_k) + \alpha \tag{1}$$

$\forall (x_i, x_j, x_k) \in X \mid y_i = y_j \wedge y_i \neq y_k$, where D is a distance function and α is a pre-specified margin. In order to achieve this goal, we optimize a contrastive loss function defined over input pairs that is therefore known as the duplet loss. The contrastive loss function is decomposed in a positive term L_+:

$$L_+(x_i, x_j) = (\beta) D(x_i, x_j)^2 \tag{2}$$

[3] During preparatory work, we also experimented with convolutional stacks but found no improvements over the recurrent counter-part, which was, therefore, singled out for the purpose of the present study.

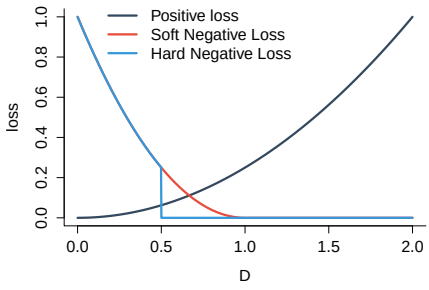


Figure 4. Visualization of the two loss variants with a hard margin and a soft margin.

and a negative term L_-:

$$L_-(x_i, x_j) = \max(0, \alpha - D(x_i, x_j))^2 \tag{3}$$

where β is a parameter used to weight the contribution of the negative term. [4] The two terms can be combined into a single loss function with the help of a variable $Y_{i,j}$ that takes value of 1 for $y_i = y_j$ and 0 when $y_i \neq y_j$:

$$L_D(x_i, x_j) = (Y_{ij}) L_+(x_i, x_j) + (Y_{ij} - 1) L_-(x_i, x_j) \tag{4}$$

For the current study, we restrict ourselves to the cosine distance as defined by Eq. 5:

$$D(x_i, x_j) = 1 - \frac{f(x_i) \cdot f(x_j)}{\|f(x_i)\| \|f(x_j)\|} \tag{5}$$

Naturally, other distance functions are equally applicable, but the two-sided boundedness of the cosine distance (i.e. distances fall between [0, 2]) allows more efficient optimization of the parameters α and β. Moreover, the loss specified above employs a soft margin. By contrast, [28] propose the use of a hard margin, effectively reducing the loss to zero if it falls below some value. With the hard margin, the negative term in Eq. 3 becomes:

$$L_-(x_i, x_j) = \begin{cases} (1 - D(x_i, x_j))^2 & D(x_i, x_j) < \alpha \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Figure 4 visualizes the two loss variants. In the experiments below, we compare both versions of the loss.

3.2.2 Triplet Loss

The triplet loss [13, 32] differs from the duplet loss in that it considers input example triplets x_i, x_j, x_k consisting of a positive example x_i, a negative example x_j such that

[4] The scaling parameter β and margin α are optimized on a development data set, which we will discuss below.

$y_j \neq y_i$ and an anchor x_k such that $y_k = y_i$. The triplet loss shares the goal of the duplet loss from Eq. 1, but employs anchor positive examples to ensure that the distance between any two instances of the same family is less than the distance to an instance of a different family by at least a pre-specified margin α. More formally, the triplet loss is defined by Eq. 7:

$$L_T(x_i, x_j, x_k) = \max(0, D(x_i, x_k) - D(x_j, x_k) + \alpha) \tag{7}$$

The triplet loss, therefore, presents a more relaxed form than the duplet loss, allowing instances of the same family to occupy larger regions. As opposed to the composite form of the duplet loss, the triplet loss is simple. However, the triplet loss is more heavily dependent on the quality of the sampled negative examples and anchors, which might lead to poorer training dynamics.

3.3 Online Duplet and Triplet Mining

We train our encoder using mini-batches of duplets or triplets from the data set. The number of possible duplets and triplets grows, respectively, quadratically and cubically with the number of instances in the data set, which renders exhaustive training costly. Feasibility aside, training with all possible duplets or triplets is not desirable as a large proportion of the resulting duplets and triplets make it either too easy or too difficult to fulfill the objective of Eq. 4 and Eq. 7. Such examples prevent the network from learning, and lead to slower convergence.

As suggested by [32] in the context of face recognition, efficient and fast converging training can be achieved by online selection of 'hard' duplets or triplets, i.e. the most dissimilar *positive* examples, and the least dissimilar *negative* examples. We apply this approach by first sampling a mini-batch of k instances per each of n sampled unique tune families. Subsequently, using the current model we compute the encodings of all $n \times k$ instances and for each instance we sample positive and negative, or anchor and negative examples from the mini-batch. In the case of the duplet loss, for each instance we select all possible positive examples (i.e. all other instances in the mini-batch from the same family) and an equal number of negative examples from the least dissimilar negatives. In the case of the triplet loss, for each instance x_i we select pairs of anchor x_k and negative example x_j such that the distance between positive and anchor is smaller than the distance between negative and anchor, while the difference between the distances lies inside the margin α:

$$0 < D(x_i, x_k) - D(x_j, x_k) < \alpha \tag{8}$$

In case no negative example can be found that satisfies this condition, we select a random negative.

3.4 Baseline: Alignment

We compare our results with the performance of a previously proposed alignment method [23]. In this method, the Needleman-Wunsch-Gotoh algorithm is used [10], which computes a global alignment score for two sequences of symbols. The alignment is constructed by inserting gaps at appropriate locations in the sequences following a dynamic programming approach. The alignment score is based on a similarity function for symbols and a gap scoring scheme. The Gotoh-variant of the algorithm applies an affine gap scoring function in which the continuation of a gap obtains a different score than the opening of a gap, opposed to the basic variant of the algorithm in which all gaps obtain the same score. We use the best scoring configuration in [23], which uses pitch, metric weight and the position of a note in its phrase.

3.5 Evaluation

We formulate the task of tune family identification as a ranking problem: given a query melody q_i and a data set of melodies $X, q_i \notin X$, the models should provide a ranked list of the melodies in X. To evaluate how well our models solve this problem, we measure the performance of the models by means of three evaluation measures: (i) 'Average Precision', (ii) 'Precision at rank 1', and (iii) 'Silhouette Coefficient'. Each of these measures addresses a different aspect of the performance quality of the models. First, Average Precision (AP) addresses the question whether given a query melody, all or most relevant melodies are high up in the ranking:

$$\text{AP} = \frac{\sum_{k=1}^{N} P(k) \times rel(k)}{\text{number of relevant melodies}}, \tag{9}$$

where k is the position in the ranked list of N retrieved melodies. $P(k)$ represents the precision at position k, and $rel(k) = 1$ if the melody at position k is relevant, $rel(k) = 0$ otherwise. By computing the average AP over all query melodies, we obtain the Mean Average Precision (MAP). As a second ranking measure, we focus on the Precision at rank 1 score (P@1), which computes the fraction of queries for which the highest ranked sequence is relevant. Third and finally, we compare these two ranking based evaluation measures with the Silhouette Coefficient, which is a measure of cluster homogeneity and separation. The Silhouette Coefficient contrasts the mean similarity between a sample and all other samples from the same family with the similarity of that sample with members of other families. By taking the average over all silhouette scores, we obtain a measure of cluster homogeneity ranging from -1 (incorrect clustering) to 1 (perfect clustering). [5]

3.6 Training and Hyper-Parameter Optimization

The networks were trained on the training data sets specified in Table 1. We use the Adam optimizer [17] and stop training after no improvement in MAP score was made on the development data for ten consecutive epochs. The neural network consists of a large number of hyper-parameters, making hyper-parameter tuning expensive and time-consuming. Following [2], we perform a randomized hyper-parameter search, in which we train n differ-

[5] See the Supplementary Materials for more information.

	MAP			P@1	Sil.
	all	seen	unseen		
RNN_D	0.72	0.71	0.73	0.78	0.34
RNN_T	0.71	0.70	0.71	0.77	0.29
Alignment	0.69	–	–	0.78	0.23

Table 2. Best evaluation scores for the development data. RNN_D is an RNN with duplet loss, and RNN_T is an RNN with triplet loss.

	MAP			P@1	Sil.
	all	seen	unseen		
RNN_D	0.72	0.70	0.74	0.78	0.33
RNN_T	0.68	0.64	0.71	0.75	0.28
Alignment	0.67	–	–	0.78	0.22

Table 3. Evaluation scores for the test data. RNN_D is an RNN with duplet loss; RNN_T is an RNN with triplet loss.

ent models with random hyper-parameter settings sampled from parameter-specific, relatively flat distributions. [6]

4. RESULTS

Table 2 presents the results for the development data set with MAP scores for the best performing models trained with duplet (RNN_D) and triplet loss (RNN_T). The best RNN_D achieves a MAP of 0.72, which is markedly better than the Alignment method (0.69), and slightly better than the best RNN_T (0.71). However, as will be discussed in more detail below, RNN_T models are significantly harder to optimize than RNN_D models (at least for the current data set). The columns 'seen' and 'unseen' represent MAP scores for queries of which the corresponding tune families were either seen or unseen during training. Crucially, the scores are almost equivalent, indicating that the neural networks are capable of actually learning a similarity metric, and not just a clustering or classification procedure, in which the systems learn to assign sequences to known class labels and data points. The performance differences between the models are further expressed by the Silhouette coefficient, which indicates superior performance of the RNN_D model. However, note that for P@1, all systems perform equally well.

The best performing models were employed to encode the melodies in the test set. The test results in Table 3 show a similar picture. Again, RNN_D outperforms the other systems. Note that the performance of RNN_T slightly dropped in comparison to the development results and that the performance difference with respect to RNN_D has become larger. Overall, the RNNs appear to have adequately learned how to form compact distributional representations of symbolic music in an induced melody space. This capability is further illustrated by the two-dimensional projection of the melody space in Figure 5. The left subplot demonstrates that the learned representations clearly separate vocal (FS) from instrumental melodies (INST). The subplot to the right serves as a validation of the clustering capabilities of the encoder. It highlights the positions of a small number of randomly chosen tune families, the members of which all cluster together.

[6] The full list of hyper-parameters and the predefined priors are listed in Section 2 of the Supplementary Materials.

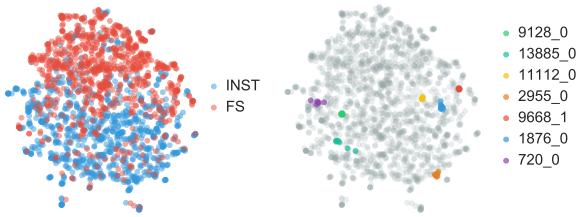


Figure 5. Two-dimensional UMAP [25] projection of the induced melody space obtained with RNN_D. The left subplot visualizes the positions of instrumental melodies (INST) and vocal melodies (FS). The subplot to the right highlights the positions of a small number of randomly chosen tune families.

4.1 Hyper-Parameter Importance

We assess the importance of the different hyper-parameters of the neural networks by modelling their influence on the MAP scores resulting from the randomized parameter search [26]. To this end, we fit the following linear regression model:

$$\text{MAP}_i \sim \mathcal{N}(\mu_i, \sigma) \quad (10)$$

$$\mu_i = \gamma + \beta_l l_i + \beta_m m_i + \beta_h h_i + \beta_d d_i + \beta_b b_i + \beta_c c_i + \beta_{lm} l_i m_i, \quad (11)$$

where Eq. 10 specifies the likelihood function with mean μ and standard deviation σ, and Eq. 11 represents the linear model. Here, γ represents the intercept of the linear model, l_i is the loss type of model i (i.e. triplet or duplet loss), m_i is the margin value α, h_i is the dimension of the hidden layer, d_i refers to the embedding dropout value, b_i dummy encodes whether a model employed bidirectional versus unidirectional RNNs, and c_i is the cell type of the RNN (i.e. LSTM or GRU). Since the margin functions differently in the triplet and duplet loss, we model the interaction between margin and loss type ($\beta_{lm} l_i m_i$). All categorical predictors are dummy encoded, and the continuous predictors are zero centered. β priors are sampled from uninformative Normal distributions, $\mathcal{N}(0, 1)$, and σ is sampled from a weakly regularizing half-Cauchy prior with location 0 and scale 1.

Table 4 presents the posterior distribution estimates of the model along with their estimation errors, their 95% credible intervals (CI_{95}), and the $\hat{R}$ statistic. [7] The mean

[7] Since the linear model is Bayesian, the credible intervals can be interpreted straightforwardly as the 95% probability that the estimates fall in a particular range. The 'No U-Turn Sampler' (NUTS) was used for sampling [15], which is a specific type of Hamiltonian Monte Carlo (HMC).

	Estimate	Error	l-CI$_{95}$	u-CI$_{95}$	$\hat{R}$
γ	0.66	0.00	0.65	0.67	1.0
β_l	-0.08	0.01	-0.09	-0.07	1.0
β_m	0.01	0.01	-0.02	0.04	1.0
β_d	-0.05	0.02	-0.09	0.00	1.0
β_b	0.03	0.01	0.02	0.04	1.0
β_c	-0.05	0.00	-0.06	-0.04	1.0
β_h	0.02	0.00	0.01	0.03	1.0
β_{lm}	-0.18	0.02	-0.22	-0.13	1.0
σ	0.04	0.00	0.04	0.04	1.0

Table 4. Posterior distribution estimates for the hyper-parameters of the Neural Networks. In addition to the mean estimates, the table provides the estimation errors, 95% Credible Intervals, and the $\hat{R}$ statistic.

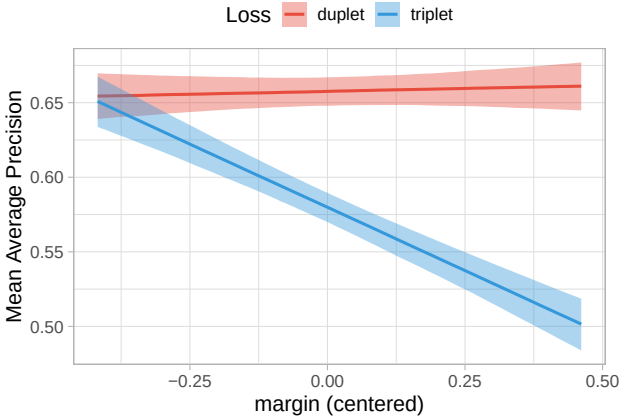


Figure 6. Marginal effects plot showing the interaction between loss type (i.e. Duplet and triplet loss) and the margin α.

intercept $\gamma = 0.66$ represents the mean posterior estimate of MAP values for unidirectional models, fit with GRU cells ($c_i = 0$), duplet loss ($l_i = 0$) and mean (i.e. 0) values for the continuous predictors. Given this base model, several interesting observations can be made. First, as suggested by the negative β_l estimate, triplet loss models markedly underperform duplet loss models with, *ceteris paribus*, a mean drop in performance of 0.08. Second, employing larger hidden dimensions (β_h) and using bidirectional RNNs (β_b) both positively influence the MAP scores. Third, on average adding too much dropout hurts performance ($\beta_d = -0.05$). Fourth, the strong negative posterior distribution estimate for the interaction between the margin α and loss type indicates that careful tuning of α is especially important for the triplet loss. By contrast, different values of α barely impact the performance of duplet loss models. The marginal effects plot in Figure 6 highlights this interaction. Finally, RNNs trained with GRU cells markedly outperform models trained with LSTM cells ($\beta_c = -0.05$). Figure 7 illustrates this performance difference. Additionally, the plot demonstrates the

$\hat{R}$ is a statistic to assess the convergence of the sampler, and should be below 1.1 [8]. For more information about the convergence and parameters, see the Supplementary Information.

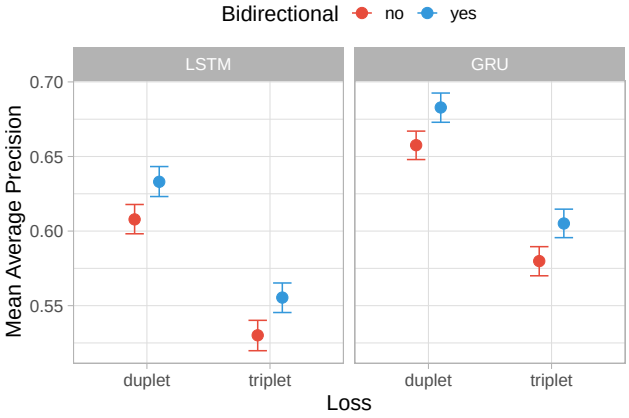


Figure 7. Marginal effects plot of RNN cell type (i.e. LSTM or GRU) and bidirectional versus unidirectional RNNs.

benefits of employing bidirectional RNNs, which consistently outperform unidirectional models.

5. CONCLUSION & FUTURE WORK

This paper proposed a method for end-to-end melody similarity metric learning using deep neural networks. We trained distributional melody encoders to minimize Duplet and Triplet Contrastive loss functions with which we achieve state-of-the-art retrieval performance on a large set of instrumental and vocal melodies. A thorough statistical analysis of the hyper-parameters of the Neural Networks indicates that on average Duplet Loss RNNs are easier to tune and less sensitive to specific hyper-parameter settings. Additionally, RNNs trained with GRU cells consistently outperform LSTM cell implementations. Our system has several major advantages over more traditional, alignment-based methods. First, thanks to its ability to infer complex interactions between input variables, the Neural Network approach is less sensitive to specific feature combinations and feature selection. Second, as shown by our study, the Neural Network approach displays more robustness, achieving similar MAP scores across exclusive sets of tune families (seen vs unseen).

For future work, we have the following three recommendations. First, the applicability of the proposed approach should be carefully examined on more diverse data sets, in order to test for the cross-domain robustness of the learned similarity metrics. Second, a more extensive and thorough comparison (including error analysis) with other existing melodic similarity methods is desired to highlight advantages and possible disadvantages of the neural systems. Finally, we acknowledge that, while successful, the proposed architecture still leaves room for improvement. Inspired by progress in similarity metric learning within the fields of Paraphrase Detection and Semantic Textual Similarity, we would like to experiment with more expressive neural architectures and feature extraction to explore the performance limits of the Neural Network approach on melodic similarity metric learning.

6. REFERENCES

[1] S.M. Bayard. Prolegomena to a study of the principal melodic families of british-american folk song. *Journal of American Folklore*, 63(247):1–44, 1950.

[2] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[3] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[4] Tian Cheng, Satoru Fukayama, and Masataka Goto. Comparing rnn parameters for melodic similarity. In *ISMIR*, pages 763–770, 2018.

[5] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111. Association for Computational Linguistics, 2014.

[6] Roger B. Dannenberg, William P. Birmingham, Bryan Pardo, Ning Hu, Colin Meek, and George Tzanetakis. A comparative evaluation of search techniques for query-by-humming using the musart testbed. *Journal of the American Society for Information Science and Technology*, 58(5):687–701, 2007.

[7] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[8] Andrew Gelman, Donald B Rubin, et al. Inference from iterative simulation using multiple sequences. *Statistical science*, 7(4):457–472, 1992.

[9] Mathieu Giraud, Ken Déguernel, and Emilios Cambouropoulos. *Fragmentations with Pitch, Rhythm and Parallelism Constraints for Variation Matching*, volume 8905 of *Lecture Notes in Computer Science*, chapter Fragmentations with Pitch, Rhythm and Parallelism Constraints for Variation Matching. Springer, 2014.

[10] O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162:705–708, 1982.

[11] Maarten Grachten, Josep Lluís Arcos, and Ramon López de Mántaras. Melody retrieval using the implication/realization model. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, 2005.

[12] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[13] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[15] Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

[16] Berit Janssen, Peter Van Kranenburg, and Anja Volk. Finding occurrences of melodic segments in folk songs employing symbolic similarity measures. *Journal of New Music Research*, 46(2):118–134, 2017.

[17] Diederik P. Kingma and Jimmy Lei Ba. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations 2015*, pages 1–15, 2015.

[18] Andreas Kornstädt. Themefinder: A web-based melodic search tool. *Computing in Musicology*, 11:231–236, 1998.

[19] Peter Van Kranenburg. *A Computational Approach to Content-Based Retrieval of Folk Song Melodies*. PhD thesis, Utrecht University, Utrecht, October 2010.

[20] Peter Van Kranenburg and Martine De Bruin. The meertens tune collections: Mtc-fs-inst 2.0. Meertens Online Reports 2019-1, Meertens Institute, Amsterdam, 2019.

[21] Peter Van Kranenburg, Martine De Bruin, and Anja Volk. Documenting a song culture: the dutch song database as a resource for musicological research. *International Journal on Digital Libraries*, 20(1):13–23, 2019.

[22] Peter Van Kranenburg, Martine De Bruin, Louis P. Grijp, and Frans Wiering. The meertens tune collections. Meertens Online Reports 2014-1, Meertens Institute, Amsterdam, 2014.

[23] Peter Van Kranenburg, Anja Volk, and Frans Wiering. A comparison between global and local features for computational classification of folk song melodies. *Journal of New Music Research*, 42(1):1–18, 2013.

[24] Velankar Makarand and Kulkarni Parag. Unified algorithm for melodic music similarity and retrieval in query by humming. In *Intelligent Computing and Information and Communication: Proceedings of 2nd International Conference, ICICC 2017*. Springer Singapore, 2018.

[25] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[26] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. An analysis of neural language modeling at multiple scales. *arXiv preprint arXiv:1803.08240*, 2018.

[27] Marcel Mongeau and David Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24:161–175, 1990.

[28] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 148–157, 2016.

[29] Justin Salamon, Joan Serrà, and Emilia Gómez. Tonal representations for music retrieval: from version identification to query-by-humming. *International Journal of Multimedia Information Retrieval*, 2(1):45–58, 2013.

[30] Patrick E. Savage and Quentin D. Atkinson. Automatic tune family identification by musical sequence alignment. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, pages 162–168, 2015.

[31] Patrick E. Savage, Charles Cronin, Daniel Müllensiefen, and Quentin D. Atkinson. Quantitative evaluation of music copyright infringement. In *Proceedings of the 8th International Workshop on Folk Music Analysis (FMA2018)*, pages 61–66, 2018.

[32] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[33] Julián Urbano, Juan Lloréns, Jorge Morato, and Sonia Sánchez-Cuadrado. Melodic similarity through shape similarity. In Sølvi Ystad, Mitsuko Aramaki, Richard Kronland-Martinet, and Kristoffer Jensen, editors, *Exploring Music Contents*, volume 6684 of *Lecture Notes in Computer Science*, pages 338–355. Springer Berlin Heidelberg, 2011.

[34] Jelmer Van Nus, Geert-Jan Giezeman, and Frans Wiering. Melody retrieval and composer attribution using sequence alignment on rism incipits. In *TENOR 2017 International Conference on Technologies for Music Notation & Representation*, 2017.

[35] Valerio Velardo, Mauro Vallati, and Steven Jan. Symbolic melodic similarity: State of the art and future challenges. *Computer Music Journal*, 40(2):70–83, 2016.

MULTI-TASK LEARNING OF TEMPO AND BEAT: LEARNING ONE TO IMPROVE THE OTHER

Sebastian Böck[1,3] **Matthew E.P. Davies**[2] **Peter Knees**[3]
[1] Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria
[2] INESC TEC, Porto, Portugal [3] TU Wien, Vienna, Austria
sebastian.boeck@ofai.at

ABSTRACT

We propose a multi-task learning approach for simultaneous tempo estimation and beat tracking of musical audio. The system shows state-of-the-art performance for both tasks on a wide range of data, but has another fundamental advantage: due to its multi-task nature, it is not only able to exploit the mutual information of both tasks by learning a common, shared representation, but can also improve one by learning only from the other. The multi-task learning is achieved by globally aggregating the skip connections of a beat tracking system built around temporal convolutional networks, and feeding them into a tempo classification layer. The benefit of this approach is investigated by the inclusion of training data for which tempo-only annotations are available, and which is shown to provide improvements in beat tracking accuracy.

1. INTRODUCTION

By definition, the music analysis tasks of tempo estimation and beat tracking are highly interconnected. Considering the goal of a beat tracking system is to produce a sequence of time instants that reflect how a human listener might tap their foot in time to a piece of music, we understand the tempo as the rate at which these beats occur, as measured in beats per minute (BPM). With the exception of a specific class of musical recordings which are both perfectly quantised (i.e. adhering strictly to a fixed metronome), and which begin precisely at the onset of a beat, e.g. drum loops, tempo information alone is insufficient to derive the beats since it provides no information about phase. In practice, a more flexible and musically realistic approach to beat tracking is required to contend with deviations from purely isochronous beat sequences without a trivial phase component. These deviations can take the form of continuous changes in tempo and/or timing which are common in expressive musical performances, more abrupt "step" changes in tempo, or short pauses after which a previously established tempo is resumed [21]. The presence and extent of these deviations from isochrony have been identified as contributing to the difficulty of musical examples for computational beat tracking [14] as well as for human annotators annotating ground truth [27].

When reflecting on the history of computational approaches for beat tracking, we consider that the role and usage of data has fundamentally changed. For the earliest work in beat tracking in the 1990s [18, 37], annotated data was scarce. By the mid-to-late 2000s, several beat tracking datasets (both public and private) came into use [12, 19, 20, 22, 24, 29] and were widely adopted as the primary means for reporting beat tracking performance. Even allowing for parameter optimisation or some training to maximise the performance of beat tracking algorithms on given datasets, a closed loop (in a strict end-to-end sense) did not exist between annotated data and beat tracking algorithms until the advent of deep neural network (DNN) approaches [7]. Both the high learning power and explicit use of annotations of DNN approaches led to a significant leap in the state of the art.

Similarly, tempo induction algorithms formerly tried to identify the main periodicity of musical accent features, such as band-passed signals, discrete onsets or a continuous detection function by means of auto-correlation [1, 13, 36], resonating comb filters [29, 37] or Fourier analysis [9], and available data was only used to evaluate the algorithms. The first attempts to learn something meaningful from data for tempo estimation sought to devise ways to choose among multiple tempo hypotheses [15, 16, 26, 38, 45] or to predict the perceptual tempo [35]. Only recently, DNN approaches have been used to infer tempo directly from spectral features [40].

At the present time, DNN approaches are highly prevalent among music analysis and generation research within the music information retrieval (MIR) community, and thus access to large amounts of high-quality annotated data is of paramount importance for the development and training of new models. For beat tracking, the hand annotation of beat locations is particularly arduous due to the need to make several hundred temporally-dependent annotations per full piece of music, and the work-load only increases in the presence of challenging musical and signal conditions [27]. By contrast, global tempo annotation, while still dependent on some approximate beat marking, can typically be created with far less effort. As a result,

there is a far greater amount of tempo annotated data available than for beat tracking.

Our motivation is therefore towards a new approach for beat tracking which can be trained not only on beat annotations but also from tempo-only annotated data. We formulate this as a multi-task learning problem [8] for simultaneous tempo estimation and beat tracking. Our hypothesis is that due to the multi-task nature, we can not only exploit the mutual information of both tasks by learning a common, shared representation, but also improve one by learning only from the other.

We implement our multi-task approach by extending a recent beat tracking system [11] built around temporal convolutional networks (TCNs) [2, 44]. The primary addition in this paper takes the form of globally aggregating the skip connections of the TCN and feeding them into a tempo classification layer. A graphical overview of the inputs and outputs of our system is shown in Figure 1, with details of the architecture in Figure 2.

We evaluate our proposed multi-task system on a wide range of existing beat- and tempo-annotated datasets and compare performance against reference systems in both tasks. Our results demonstrate that the multi-task formulation achieves state-of-the-art performance in both tempo estimation and beat tracking. The most notable increase in performance occurs on a dataset where the network has been trained on tempo labels but whose beat annotations remain totally unseen by the network.

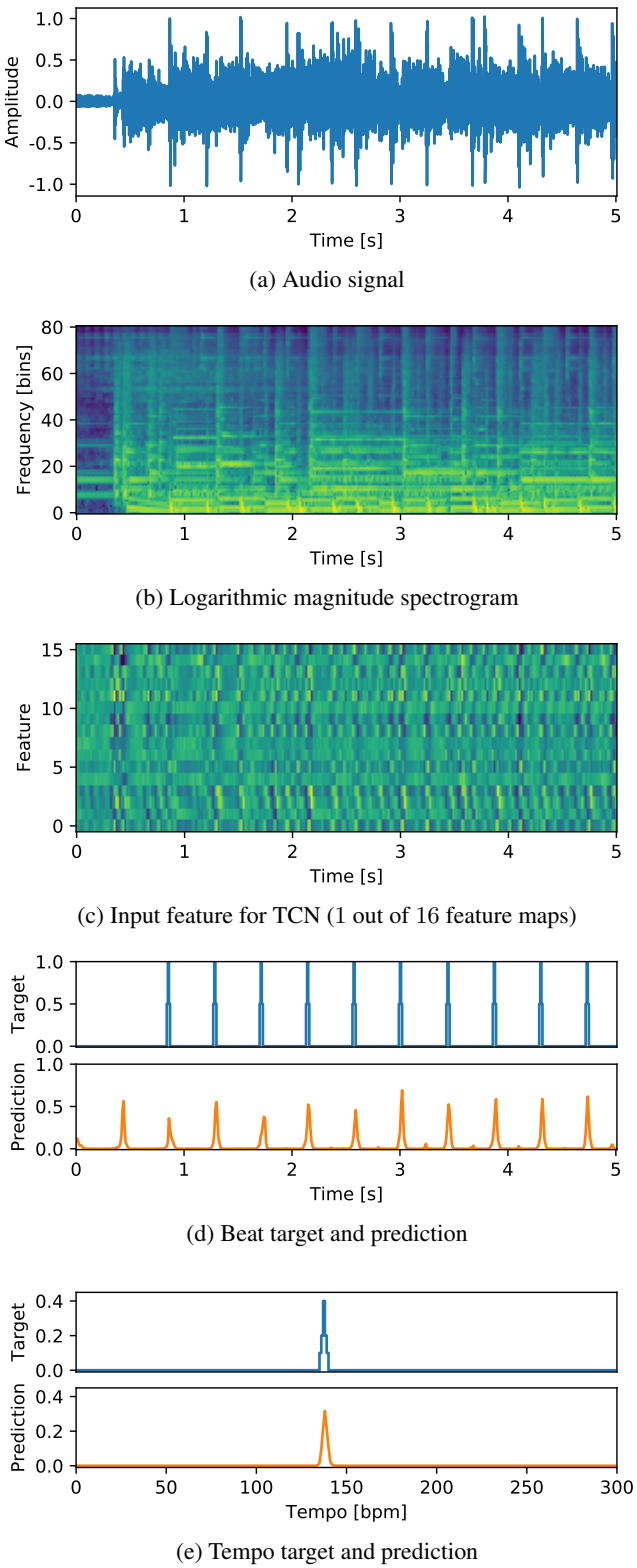
The remainder of this paper is structured as follows. In Section 2 we provide an overview of the existing beat tracking approach and then detail our multi-task formulation. In Section 3 we conduct a rigorous evaluation of beat tracking and tempo estimation. Finally, in Section 4 we discuss the context of the results and propose areas for future work.

2. APPROACH

In this section, we provide an overview of the beat tracking system [11] around which our multi-task learning approach is formulated. Following this, we describe the extension for multi-task learning via the inclusion of an additional output layer which performs tempo classification.

2.1 Beat Tracking Approach

The underlying beat tracking approach is inspired by two well-known deep learning methods: i) the *WaveNet* model [44] which uses dilated convolutions for generative audio synthesis by learning directly on raw audio waveforms, and ii) the current state of the art in musical audio beat tracking [4, 6], which uses a bi-directional long short-term memory (BLSTM) recurrent architecture. Based on the work of Bai et al. [2], who demonstrated improved performance of TCNs over recurrent architectures for numerous sequential data analysis and classification tasks, we developed a TCN approach for musical audio beat tracking [11] which, at a high-level, addressed the substitution of the BLSTM in [4, 6] with a TCN. However, since the



(a) Audio signal

(b) Logarithmic magnitude spectrogram

(c) Input feature for TCN (1 out of 16 feature maps)

(d) Beat target and prediction

(e) Tempo target and prediction

Figure 1: Signal flow of a 5 second audio excerpt through the proposed multi-task system. From the time domain signal (a), a logarithmic magnitude spectrogram is computed (b). This input representation is processed by intermediate convolutional and max pooling layers to obtain a single 16-dimensional feature (c), which is fed into the TCN. Both targets and predictions for beats and tempo are shown in (d) and (e), respectively.

TCN from *WaveNet* is both causal and operates on raw audio, several modifications were required, which are summarised below.

Instead of using raw audio as input, the dilated convolutions are performed on a highly sub-sampled low-dimensional feature representation (cf. Figure 1c). This 16-dimensional feature vector is derived by applying multiple convolution and max pooling operations to a log magnitude spectrogram of the input audio signal. The spectrogram is computed with a window and FFT size of 2048 samples, a hop size of 441 samples (i.e. 100 frames per second for audio sampled at 44100 Hz), and filtered with a bank of overlapping triangular filters with 12 bands per octave covering a frequency range of 30 to $17,000$ Hz (cf. Figure 1b). Alternating convolutional and max pooling layers are applied to slices of 5 frames in length to reduce the dimensionality both in time and frequency to a single dimension. The convolutional layers contain 16 filters each, with kernel sizes of 3×3 for the first two, and 1×8 for the last layer. The intermediate max pooling layers apply pooling only in the frequency direction over 3 frequency bins. A dropout [42] rate of 0.1 is used with the exponential linear unit (*ELU*) [10] as activation function.

The main TCN component from *WaveNet* was modified to operate non-causally, meaning that, for any time frame of the input representation, the dilated convolutions extend in both directions (i.e. back to the past and forward to the future). This provides a receptive field which is centred on the time frame in question, rather than directed solely towards the past.

In terms of the parameterisation of the TCN approach we used 11 layers with 16 1-dimensional filters of size 5 and geometrically spaced dilations ranging from 2^0 up to 2^{10} time frames. The resulting receptive field is ~ 81.5 seconds. We applied spatial dropout with rate 0.1 and used the *ELU* activation function instead of the gated activations of *WaveNet*. As output we used a single *sigmoid* unit. In order to obtain a final beat tracking output, the beat activation function produced by the network was passed to a dynamic Bayesian network, approximated by a hidden Markov model, from [31]. For further details on the TCN approach for beat tracking, see [11].

In this work we slightly changed the architecture of [11] by adding another 1×1 convolution layer with 16 filters into the residual path of the TCN layers (cf. Figure 2). We found that this layer helped to increase tempo estimation performance.

2.2 Multi-Task Extension

We extend this beat tracking system to be able to estimate the tempo of a musical piece by adding a second output branch to the network. As output, a classification layer with linear spacing as in [40] is used. It has 300 units, representing a tempo range from 0 (indicating that the piece has no tempo) up to 300 BPM. This additional output allows for multi-task learning of the whole system, the details of which are outlined in Figure 2. In order to be able to process input sequences of variable length, global average pooling (over time) is used to aggregate the features for the tempo classification layer.

While it is possible to feed the output of the TCN (or indeed the output of any other sequential beat tracking model) directly to the tempo classification layer, in practice we found that using a beat activation function led to reasonable "coarse" tempo estimation performance (i.e. determining whether a musical piece is either fast or slow), but lacked absolute precision. However, utilising skip connections of the TCN boosted tempo estimation accuracy considerably. Our intuition is that this way the subtleties of the intermediate representation of the dilated convolutions (which represent different time scales) are preserved and can be better exploited.

In the original *WaveNet* [44], skip connections were used to speed up convergence and enable training of deep models. Since the TCN beat tracking system [11] has only 11 layers, skip connections were not needed to successfully train a model and thus were not utilised. In this work, we branch off the skip connections at the same location as in *WaveNet* (i.e. from the 1×1 convolutions inside the TCN layers), but use them solely for the tempo branch of the network (cf. Figure 2).

We aggregate the skip connections of the individual layers by summation. Since the 1×1 convolutions have 16 filters each, this results in a single 16-dimensional feature vector for classification. Compared to concatenating the skip connections, this low-dimensional input to the tempo classification layer reliably prevents over-fitting to the training data. We apply dropout [41] with rate 0.5 before feeding this vector in the final tempo classification layer with a *softmax* function. During inference, quadratic interpolation of the output probability distribution is used to determine the final tempo in BPM.

The whole system has only $29{,}901$ trainable parameters, from which the multi-task tempo classification extension accounts for $5{,}100$. We contrast our compact model with the reported $2.9M$ parameters of the current state of the art in tempo estimation [40].

2.3 Network Training

To train the system, we represent annotated beat training data as impulse trains at the same temporal resolution as our input feature (i.e. 100 frames per second). To allow for slight deviations of the annotated beat locations and partially address the imbalance between the number of beat and non-beat frames, we use the neighbouring frames of the annotated beat positions as positive examples, but weight them by a scaling factor of 0.5 (cf. Figure 1d).

Given beat annotations, we derive tempo annotations by counting the inter-beat-intervals (IBI) to build a histogram. We smooth this histogram with a Hamming window of size 15 frames (i.e. 150 ms) to counteract small fluctuations of the beat annotations and determine the most dominant IBI by quadratic interpolation. This interval is then converted to tempo in BPM and mapped to tempo targets representing integer BPM values. In a similar way to the widening of the beat annotations, we smooth the tempo targets, but

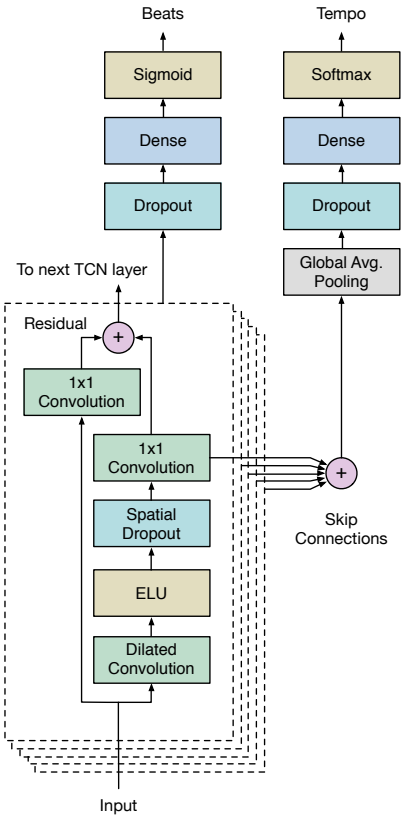


Figure 2: Structure of the neural network with the TCN for beat tracking (left) and the multi-task extension for tempo estimation (right).

extend the range to ± 2 BPM, weighting the neighbouring BPM targets with 0.5 and 0.25, respectively. We then normalise the tempo targets to form a probability distribution (as shown in Figure 1e) in order for it to be usable with the *softmax* activation function.

For training, we combine the cross-entropy losses of both network outputs by weighting them equally. Since the training sequences have different lengths, we train on whole sequences and minimise the combined loss with stochastic gradient descent (i.e. using a batch size of 1). We use *Adam* [28] with an initial learn rate of 0.002, and reduce it by a factor of 5 whenever the validation loss reaches a plateau and stop training if no improvement in validation loss is observed for 50 consecutive epochs or if a maximum of 150 epochs have elapsed. To avoid exploding gradients, we clip the gradients to a maximum norm of 0.5. If only tempo targets are present for training, we mask the loss of the beat tracking output. This way, only the error of the tempo output is backpropagated through the network and used to update the weights. It is important to note, that even in this scenario the shared beat and tempo feature representation gets adapted and optimised.

3. EXPERIMENTS AND EVALUATION

For experiments and evaluation we use the datasets listed in Table 1. Those listed in the upper part are used for training using 8-fold cross validation, and those in the lower part are independent test sets held back for evaluation only. If available, updated annotations are used and indicated by additional references. We chose these datasets in order to be able to compare the performance of our proposed systems to the best performing reference systems for both beat tracking and tempo estimation.

Dataset	files	length
Ballroom [23, 32] [1]	685	5 h 57 m
Beatles [12]	180	8 h 09 m
Hainsworth [24]	222	3 h 19 m
Simac [20]	595	3 h 18 m
SMC [27]	217	2 h 25 m
HJDB [25] $*$	235	3 h 19 m
ACM Mirum [35] $\star$	1410	15 h 05 m
GiantSteps [30, 39] $\star$	664	22 h 05 m
GTZAN [33, 43]	999	8 h 20 m

Table 1: Datasets used for training (upper half), and testing (lower half). The $*$ symbol denotes that only tempo annotations were used during training and beat annotations are used for evaluation only, and the $\star$ symbol indicates those datasets for which only tempo annotations exist.

The *HJDB* (Hardcore, Jungle, Drum & Bass) dataset is used to demonstrate the effectiveness of our multi-task extension w.r.t. its ability to improve beat tracking performance using only the tempo annotations of this set. This dataset was chosen, since its distinct music style is not well represented within any of the other training sets.

3.1 Beat Tracking Evaluation

We compare our proposed multi-task system to existing state-of-the-art beat tracking systems, namely to the underlying TCN approach presented in [11], and the two BLSTM approaches for beat [4] and joint beat and downbeat tracking [6]. Our goal is that the inclusion of the tempo classification layer is never detrimental to the performance of the beat tracking component.

Following the *de facto* standard for beat tracking evaluation, we report a set of different metrics with the parameterisation defined in [12]. We use the standard *F-measure*, as well as the continuity based measures *CMLc* and *CMLt* which require the beats to be tracked at the correct metrical level, as well as *AMLc* and *AMLt* which also allow alternate metrical interpretations such as double/half and offbeat. They either consider only the longest consecutive correctly tracked segment (*xMLc*) or all correctly tracked beats of a musical piece (*xMLt*).

From the results given in Table 2 it can be seen that all systems achieve essentially the same level of beat tracking accuracy, independent of the evaluation method. There are, however, smaller deviations from this general tendency. The beat output of the downbeat tracking system [6] performs slightly better on the *Ballroom* set, which might be

[1] The 13 identified duplicates were removed: `http://media.aau.dk/null_space_pursuits/2014/01/ballroom-dataset.html`

due to the characteristic rhythmic patterns which can be better exploited by explicit modelling of whole bars.

	F	*CMLc*	*CMLt*	*AMLc*	*AMLt*
			Ballroom		
BLSTM [4]	0.917	0.832	0.849	0.905	0.926
BLSTM [6]	0.938	0.872	0.892	0.932	0.953
TCN [11]	0.933	0.864	0.881	0.909	0.929
Multi-task	0.931	0.864	0.883	0.908	0.930
			Hainsworth		
BLSTM [4]	0.884	0.769	0.808	0.873	0.916
BLSTM [6]	0.871	0.732	0.784	0.849	0.910
TCN [11]	0.874	0.755	0.795	0.882	0.930
Multi-task	0.877	0.756	0.798	0.880	0.928
			SMC		
BLSTM [4]	0.529	0.296	0.428	0.383	0.567
BLSTM [6]	0.516	0.307	0.406	0.429	0.575
TCN [11]	0.543	0.315	0.432	0.462	0.632
Multi-task	0.535	0.295	0.415	0.440	0.613
			GTZAN		
BLSTM [4]	0.864	0.750	0.768	0.901	0.927
BLSTM [6]	0.856	0.716	0.744	0.876	0.919
TCN [11]	0.843	0.695	0.715	0.889	0.914
Multi-task	0.847	0.702	0.724	0.886	0.916

Table 2: Beat tracking results on datasets used for training with 8-fold cross validation (top), and on completely unseen test data (bottom).

Given these results, we infer that the multi-task system achieves at least the same performance as the same system without the multi-task extension.

3.2 Multi-Task Evaluation

In the previous section, our evaluation focused on the use of both tempo- and beat-annotated training data within our multi-task model. In order to test our hypothesis that tempo-only information can indeed lead to improved beat tracking accuracy, and thus demonstrate the ability of multi-task learning to strengthen one target by learning additionally from the other, we perform a further experiment. To this end, we add a new dataset, but only use its tempo annotations for training.

We believe that the effect of this learning strategy should be most visible when performed with data, which is otherwise underrepresented in the training set. In our opinion, the *HJDB* dataset is a perfect fit since it contains musical genres from the early 1990s, namely Hardcore, Jungle, and Drum & Bass, which are characterised by their very distinct rhythmic structure. For the details on this dataset, see [25].

We train our new multi-task approach in two different ways. Once with the data as outlined in Table 1, but without *HJDB* (i.e. as in the previous section), and once including the tempo annotations of this dataset.

Inspection of the first two rows of Table 3 reveals that both the original TCN beat tracking system, and the system with the multi-task extension achieve roughly the same performance across all evaluation methods. However, once the additional tempo information is utilised (last row marked with the $*$ symbol), the performance increases by up to ~ 5 percentage points. The jump in accuracy is best observed in the *CMLc* and *CMLt* evaluation methods. This indicates that the system is able to exploit the additional information to track the beats at the correct metrical level more often than without this information. Within the context of the *HJDB* dataset where the "correct" metrical level is largely unambiguous, we consider this to be an important contribution.

	F	*CMLc*	*CMLt*	*AMLc*	*AMLt*
			HJDB		
TCN [11]	0.842	0.802	0.810	0.903	0.912
Multi-task	0.850	0.800	0.804	0.921	0.927
Multi-task $*$	0.882	0.848	0.858	0.937	0.947

Table 3: Multi-task learning beat tracking results on the *HJDB* dataset. All results obtained with 8-fold cross validation. The $*$ symbol denotes that tempo annotations of the *HJDB* set were used as additional targets during training.

3.3 Tempo Evaluation

Further to the beat tracking oriented evaluation results reported in the previous two sections, we also explore the effectiveness of our proposed approach for the task of global tempo estimation. To discover how our multi-task approach compares to the state of the art, we contrast its performance against four reference systems [5, 17, 36, 40]. Following the established evaluation practice for tempo estimation [23] we report the *Accuracy 1* and *Accuracy 2* scores with a tolerance of $\pm 4\%$ for each of these methods, with the results shown in Table 4.

Given that human perception of tempo is known to be subjective [34], this very reasonably manifests in multiple, valid interpretations of the beat among listeners and thus more than one acceptable tempo. Thus, in the context of automatic tempo estimation, it may not be realistic to expect to obtain near perfect performance on the *Accuracy 1* score on datasets of arbitrary musical makeup. To this end, we rely on the *Accuracy 2* score (which permits so-called "tempo octave errors") to better gauge performance.

On all of the reported datasets in Table 4, our proposed approach is the only one to consistently obtain an *Accuracy 2* greater than or equal to 0.938, which shows the high potential of our method to accurately find tempo across diverse musical data. Even with the stricter *Accuracy 1* evaluation, our method achieves at least a score of 0.697 which is ahead of all other methods, albeit by a small margin. It is important to stress that the *ACM Mirum*, *GiantSteps*, and *GTZAN* datasets are completely unseen by our multi-task approach, and this pattern even holds for *HJDB* when not included in the training set.

Concerning the *HJDB* set, we can observe a different overall pattern of performance compared to the other datasets, with a much smaller gap between *Accuracy 1* and

	Accuracy 1	*Accuracy 2*
ACM Mirum		
Gkiokas et al. [17]	0.725	0.979
Percival and Tzanetakis [36]	0.733	0.972
Böck et al. [5]	0.741	0.976
Schreiber and Müller [40]	0.795	0.974
Multi-task	0.757	0.977
Multi-task ∗	0.749	0.974
GiantSteps		
Gkiokas et al. [17]	0.721	0.922
Percival and Tzanetakis [36]	0.506	0.956
Böck et al. [5]	0.589	0.864
Schreiber and Müller [40]	0.730	0.893
Multi-task	0.697	0.958
Multi-task ∗	0.764	0.958
GTZAN		
Gkiokas et al. [17]	0.651	0.931
Percival and Tzanetakis [36]	0.658	0.924
Böck et al. [5]	0.697	0.950
Schreiber and Müller [40]	0.694	0.926
Multi-task	0.697	0.939
Multi-task ∗	0.673	0.938
HJDB		
Gkiokas et al. [17]	0.783	0.911
Percival and Tzanetakis [36]	0.285	1.0
Böck et al. [5]	0.796	0.868
Schreiber and Müller [40]	0.902	0.991
Multi-task	0.826	0.962
Multi-task ∗ †	1.0	1.0

Table 4: Tempo estimation results on completely unseen data. The ∗ symbol denotes that tempo annotations of the *HJDB* set were used as additional targets during training, the † symbol results obtained with 8-fold cross validation.

Accuracy 2 for most systems. Echoing the situation in the beat tracking evaluation in Table 3, we believe that this is a direct result of the unambiguous tempo for these styles of music. Looking across the performance of the other algorithms on *HJDB*, we discover that the method of Percival and Tzanetakis [36], while it also obtains a perfect score for *Accuracy 2*, is largely unable to identify the annotated tempo as shown by the disproportionately low score for *Accuracy 1*.

When trained with the additional tempo annotations of the *HJDB* set, our multi-task method is the only one able to detect the correct tempo for all pieces of this dataset for both *Accuracy 1* and then trivially for *Accuracy 2*. Although results are obtained with cross-validation, this was to be expected because of the homogeneity of the dataset. *Accuracy 1* on the *GiantSteps* set also greatly benefits from this additional training material, since this dataset contains a huge proportion of music labelled with the musical genre "drum and bass". On the other hand, having access to this kind of data (the system of Schreiber and Müller [40] was trained on an extended version of the *GiantSteps* dataset) can in turn result in very good scores on the *HJDB* set.

4. DISCUSSION AND CONCLUSIONS

In this paper, we have proposed a novel formulation for the simultaneous estimation of tempo and beat from musical audio signals within a multi-task learning framework. Via an extensive evaluation of both beat tracking and tempo estimation, we have demonstrated that our proposed multi-task approach leads to state-of-the-art performance across a wide variety of test datasets and relevant evaluation methods. Perhaps most critically, we have shown that, within this multi-task learning framework, we can improve the performance of beat tracking by providing it tempo-only annotations. In light of the challenges of obtaining high-quality annotated data for training beat tracking systems, the ability to profit from alternative training data which is both far more prevalent and easier to annotate, may have a significant impact on beat tracking moving forward.

In order to train our model, we made use of all of the available beat and tempo annotations within the allocated training sets in Table 1, and subsequently provided additional tempo-only annotations for evaluation on the *HJDB* dataset. We consider this split between beat and tempo annotated data to be one that is worthy of further exploration, in particular by seeking to understand how little beat annotated data is sufficient to achieve the same performance, assuming we can supplement the model with additional tempo annotations. This reduction of beat information could be posed in two ways, either by a lower number of fully annotated excerpts/pieces, or by restricting the duration of annotated sections across many pieces. If successful, the latter option would offer the possibility to rapidly increase the availability of training data by drastically reducing the burden of annotating long pieces of music—at least for those with roughly constant tempo.

We frame this discussion within the computational context of our proposed multi-task approach and the TCN beat tracker [11] which it extends. As previously stated, our multi-task model is highly effective in terms of objective performance, but with a fraction of the number of weights of other state-of-the-art approaches. This has two particularly beneficial properties. First, it allows for very efficient training (thanks in part to the ease of parallelisation of dilated convolutional models compared to recurrent architectures). Second, the training of networks with very few weights drastically reduces the degrees of freedom of the network, and hence strongly mitigates over-fitting. Thus, when looking beyond the limited domain of existing annotated datasets and considering generalisation capabilities of beat tracking and tempo estimation methods (and the subsequent re-use of this information for end-users) on totally unseen data, we believe that such "compact" deep models are worthy candidates for future research.

Supplementary material can be found online at `https://github.com/superbock/ISMIR2019` with executable code and pre-trained models being included in *madmom* [3] (`https://github.com/CPJKU/madmom`).

5. ACKNOWLEDGEMENTS

Sebastian Böck is supported by the Austrian Promotion Agency (FFG) under the "BASIS, Basisprogramm" umbrella program. Matthew E.P. Davies is supported by Portuguese National Funds through the FCT-Foundation for Science and Technology, I.P., under the project IF/01566/2015.

6. REFERENCES

[1] M. Alonso, G. Richard, and B. David. Accurate tempo estimation based on harmonic + noise decomposition. *EURASIP Journal on Applied Signal Processing*, pages 161–161, 2007.

[2] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[3] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer. Madmom: A new python audio and music signal processing library. In *Proc. of the 2016 ACM Multimedia Conf.*, pages 1174–1178, 2016.

[4] S. Böck, F. Krebs, and G. Widmer. A multi-model approach to beat tracking considering heterogeneous music styles. In *Proc. of the 15th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 603–608, 2014.

[5] S. Böck, F. Krebs, and G. Widmer. Accurate tempo estimation based on recurrent neural networks and resonating comb filters. In *Proc. of the 16th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 625–631, 2015.

[6] S. Böck, F. Krebs, and G. Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *Proc. of the 17th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 255–261, 2016.

[7] S. Böck and M. Schedl. Enhanced beat tracking with context-aware neural networks. In *Proc. of the 14th Intl. Conf. on Digital Audio Effects (DAFx)*, pages 135–139, 2011.

[8] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[9] A. T. Cemgil, B. Kappen, P. Desain, and H. Honing. On tempo tracking: Tempogram representation and Kalman filtering. *Journal of New Music Research*, 28:4:259–273, 2001.

[10] D. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *Proc. of the 4th Intl. Conf. on Learning Representations (ICLR)*, 2016.

[11] M. E. P. Davies and S. Böck. Temporal convolutional networks for musical audio beat tracking. In *Proc. of the 27th European Signal Processing Conf. (EUSIPCO)*, 2019.

[12] M. E. P. Davies, N. Degara, and M. D. Plumbley. Evaluation methods for musical audio beat tracking algorithms. Technical Report C4DM-TR-09-06, Centre for Digital Music, Queen Mary University of London, 2009.

[13] S. Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30:39–58, 2001.

[14] S. Dixon. An empirical comparison of tempo trackers. In *Proc. of the 8th Brazilian Symp. on Computer Music*, pages 832–840, 2001.

[15] A. Elowsson. Beat tracking with a cepstroid invariant neural network. In *Proc. of the 17th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 351–357, 2016.

[16] A. Gkiokas, V. Katsouros, and G. Carayannis. Reducing tempo octave errors by periodicity vector coding and SVM learning. In *Proc. of the 13th Intl Society for Music Information Retrieval Conf. (ISMIR)*, pages 301–306, 2012.

[17] A. Gkiokas, V. Katsouros, G. Carayannis, and T. Stafylakis. Music tempo estimation and beat tracking by applying source separation and metrical relations. In *Proc. of the 37th IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 421–424, 2012.

[18] M. Goto and Y. Muraoka. A beat tracking system for acoustic signals of music. In *Proc. of the 2nd ACM Intl. Conf. on Multimedia*, pages 365–372, 1994.

[19] M. Goto. AIST annotation for the RWC music database. In *Proc. of the 7th Intl. Conf. on Music Information Retrieval (ISMIR)*, pages 359–360, 2006.

[20] F. Gouyon. *A computational approach to rhythm description — Audio features for the computation of rhythm periodicity functions and their use in tempo induction and music content processing*. PhD thesis, Universitat Pompeu Fabra, 2005.

[21] F. Gouyon and S. Dixon. A review of automatic rhythm description systems. *Computer Music Journal*, 25(1):34–54, 2005.

[22] F. Gouyon and P. Herrera. Determination of the meter of musical audio signals: Seeking recurrences in beat segment descriptors. In *Audio Engineering Society Convention 114*, 2003.

[23] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1832–1844, 2006.

[24] S. Hainsworth and M. Macleod. Particle filtering applied to musical tempo tracking. *EURASIP Journal on Applied Signal Processing*, 15:2385–2395, 2004.

[25] J. Hockman, M. E. P. Davies, and I. Fujinaga. One in the Jungle: Downbeat detection in Hardcore, Jungle, and Drum and Bass. In *Proc. of the 13th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 169–174, 2012.

[26] J. Hockman and I. Fujinaga. Fast vs slow: Learning tempo octaves from user data. In *Proc. of the 11th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 231–236, 2010.

[27] A. Holzapfel, M. E. P. Davies, J. R. Zapata, J. L. Oliveira, and F. Gouyon. Selective sampling for beat tracking evaluation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(9):2539–2548, 2012.

[28] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. of the 3rd Intl. Conf. for Learning Representations (ICLR)*, 2015.

[29] A. Klapuri, A. Eronen, and J. Astola. Analysis of the meter of acoustic musical signals. *IEEE Transactions Speech and Audio Processing*, 14(1):342–355, 2006.

[30] P. Knees, A. Faraldo, P. Herrera, R. Vogl, S. Böck, F. Hörschläger, and M. Le Goff. Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections. In *Proc. of the 16th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 364–370, 2015.

[31] F. Krebs, S. Böck, and G. Widmer. An efficient state space model for joint tempo and meter tracking. In *Proc. of the 16th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 72–78, 2015.

[32] F. Krebs, S. Böck, and G. Widmer. Rhythmic pattern modeling for beat and downbeat tracking in musical audio. In *Proc. of the 14th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 227–232, 2013.

[33] U. Marchand and G. Peeters. Swing ratio estimation. In *Proc. of the 18th Intl. Conf. on Digital Audio Effects (DAFx)*, pages 423–428, 2015.

[34] D. Moelants and M. McKinney. Tempo perception and musical content: What makes a piece fast, slow or temporally ambiguous. In *Proc. of the 8th Intl. Conf. on Music Perception and Cognition*, pages 558–562, 2004.

[35] G. Peeters and J. Flocon-Cholet. Perceptual tempo estimation using GMM-regression. In *Proc. of the 2nd ACM workshop on music information retrieval with user-centered and multimodal strategies (MIRUM)*, pages 45–50, 2012.

[36] G. Percival and G. Tzanetakis. Streamlined tempo estimation based on autocorrelation and cross-correlation with pulses. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(12):1765–1776, 2014.

[37] E. D. Scheirer. Tempo and beat analysis of acoustic musical signals. *The Journal of the Acoustical Society of America*, 103(1):588–601, 1998.

[38] H. Schreiber and M. Müller. A post-processing procedure for improving music tempo estimates using supervised learning. In *Proc. of the 18th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 235–242, 2017.

[39] H. Schreiber and M. Müller. A crowdsourced experiment for tempo estimation of electronic dance music. In *Proc. of the 19th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 409–415, 2018.

[40] H. Schreiber and M. Müller. A single-step approach to musical tempo estimation using a convolutional neural network. In *Proc. of the 19th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 100–105, 2018.

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[42] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 648–656, 2015.

[43] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[44] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR abs/1609.03499*, 2016.

[45] F.-H. F. Wu and J.-S. R. Jang. A supervised learning method for tempo estimation of musical audio. In *22nd Mediterranean Conf. of Control and Automation (MED)*, pages 599–604, 2014.

CAN WE INCREASE INTER- AND INTRA-RATER AGREEMENT IN MODELING GENERAL MUSIC SIMILARITY?

Arthur Flexer and Taric Lallai
Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria
arthur.flexer@ofai.at

ABSTRACT

We present a pilot study on ways to increase inter- and intra-rater agreement in quantification of general similarity between pieces of music. By using a more controlled group of human subjects and carefully curating song material, we try to increase overall agreement between raters concerning the perceived general similarity of songs. Repeated conduction of the experiment with a two week lag shows that intra-rater agreement is higher than inter-rater agreement. Analysis of the results and interviews with test subjects suggests that the genre of songs was a major factor in judging similarity between songs. We discuss the impacts of our results on evaluation of respective machine learning models and question the validity of experiments on general music similarity.

1. INTRODUCTION

One of the successful applications of Music Information Retrieval (MIR) is the automatic recommendation of music or creation of playlists as is now commonplace and ubiquitous in music streaming services like Spotify, Deezer, Pandora or Tidal. These services often recommend music which is in some way similar to what users have been listening before. Therefore objective assessment of the quality of such services requires a quantification of similarity between recommended songs that mirrors the human perception of music similarity. Previous research [6, 8, 10, 14, 20] has shown that perception of music similarity is highly subjective with low inter-rater agreement. This is especially true for perception of *general* music similarity. Because it is not meaningful to have computational models that go beyond the level of human agreement, these levels of inter-rater agreement present a natural upper bound for any algorithmic approach [6,10,15,22,25]. To overcome this principal problem, a range of solutions have been proposed including better control of subject groups and song material, analysis of more specific aspects of music similarity, personalization of recommendations or holistic evaluation of complete MIR systems in specific use cases [6, 20]. In this paper we present a pilot study on the feasibility to improve inter-rater agreement in modeling music similarity by confining subject groups and carefully curating song material. We also report on levels of intra-rater agreement when the experiment is repeated with a two week time-lag. To the best of our knowledge, levels of intra-rater agreement have never been explored in MIR so far.

2. RELATED WORK

It seems a fundamental fact that human perception of music is highly subjective with potentially low inter-rater agreement. To give one example, if different human subjects are asked to rate the same song pairs according to their perceived similarity, only a certain amount of agreement can be expected due to a number of subjective factors [6, 20] like personal taste, musical expertise, familiarity with the music, listening history, current mood, etc. The same holds for annotation of music where different human subjects will not always agree on genre labels or other semantic tags. It was shown [23] that the performance of humans classifying songs into 19 genres ranges from modest 26% to 71% accuracy, depending on the test subject. A study [14] on transcription of chords found that annotators disagree on about 10% of harmonic annotations. A related study [10] on chord annotation showed that annotators, if given full freedom to choose chords, tend to use different chord-label vocabularies, with overlap among all annotators being less than 20%. Audio-based grounding of everyday musical terms also showed problematic results [1].

Going even further, the argument has been made [29] that music itself does not exist without the psychophysiological effect of a stimulus on humans. Therefore no such thing as an immovable 'ground' exists in the context of music, which itself is subjective, highly context-dependent and not constant. A similar conclusion has been drawn in a study on the feasibility of automatically annotating acousmatic music [9]. The same problem is also well known in general IR, where already fifty years ago it has been documented that the implicit use orientation strongly influences manual rating of retrieved items [4].

Connected to this problem, a certain level of inter-rater agreement naturally presents an upper bound for any algorithmic approach trying to provide models which are valid for a multitude of users. Whenever these models are tested by new users, there will be a certain amount of disagree-

ment making it impossible that these computational models surpass the level of human agreement. This has been documented [6,8,20] for the MIREX [1] tasks of 'Audio Music Similarity and Retrieval' (AMS) and 'Music Structural Segmentation' (MSS). MIREX ('Music Information Retrieval Evaluation eXchange') is an annual evaluation campaign for MIR algorithms [5]. Since our experiment reported in Section 4 is closely connected to the MIREX task of 'Audio Music Similarity and Retrieval' (AMS), we now review previous results concerning rater agreement in the AMS task [6]. The essence of the AMS task was to have human graders evaluate pairs of query/candidate songs according to their general similarity. The query songs were randomly chosen from a test database and the candidate songs are recommendations automatically computed by participating algorithms. The human graders rated whether these query/candidate pairs "sound similar" using both a BROAD ('not similar', 'somewhat similar', 'very similar') and a FINE score (from 0 to 10 or from 0 to 100, depending on the year the AMS task was held, indicating degrees of similarity ranging from failure to perfection).

Only in the year 2006 [2] every query/candidate pair in the AMS task has been evaluated by three different human graders, which makes 2006 the only year inter-rater agreement can be accessed. The average Pearson correlation between pairs of graders was found to be at the rather low level of 0.40. The same authors [6] also derived an upper bound based on ratings within the highest interval of scores, where query/candidate pairs that have been rated between 9 and 10 by one grader have received an average rating of 6.54 by the respective other two graders. This was explained to constitute an upper bound B^{AMS} as the maximum of average scores that can be achieved within the AMS evaluation setting, based on a considerable lack of agreement between human graders. What sounds very similar to one of the graders will on average not receive equally high scores by other graders. For AMS the upper bound has already been reached in 2009 by a number of algorithms [6].

Related results exist for the MIREX 'Music Structural Segmentation' (MSS) task, where an upper bound for MSS has been reported which is already within reach for some genres of music [6]. Additional results for music structure analysis [15, 25] and segment boundary recognition [22] exist. The level of inter-rater agreement has also been explored for melody extraction [2, 3, 18], metrical structure [17], rhythm and timbre similarity [16] as well as chord estimation [12]. An interesting new approach [11] is to use deep learning to account for different annotator styles in the task of chord labeling, essentially personalizing chord labels for individual annotators.

To the best of our knowledge, intra-rater agreement has so far not been explored in MIR, but in general IR it is a well documented fact that items are judged differently over time, even by the same people [19].

[1] http://www.music-ir.org/mirex

[2] https://www.music-ir.org/mirex/wiki/2006:Audio_Music_Similarity_and_Retrieval

3. EXPERIMENT

Our experiment is closely connected to the MIREX task of 'Audio Music Similarity and Retrieval' (AMS) as reviewed in Section 2. We still aim at quantification of general similarity among song pairs by human graders, but try to increase inter- and intra-rater agreement by introducing a number of changes. Essential differences of our experiment include: (i) a more controlled group of human graders; (ii) carefully curated song material; (iii) query/candidate pairs are not results of algorithmic recommendation but of constrained random assignment.

Participants: In selecting test subjects for this study, we were aiming at a more uniform group of persons compared to the MIREX AMS task, where participants were drawn form the larger MIR community without any restrictions and without collecting any personal data. The major selection criteria for taking part in our study was to have had musical training in the past in some way, which should give all participants a comparable background in music. We also selected participants from an essentially young age group, which resulted in all participants having been born after 1984 and their age ranging from 25 to 34 years. The sample consisted of three females and three males. All participants were personal contacts of one of the authors.

Song Material: Contrary to the MIREX AMS task with songs from nine genres, songs in this study belong to only five different genres. The genres are: (i) American **Soul** from the 1960s and 1970s with only male singers singing; (ii) **Bebop**, the main jazz style of the 1940s and 1950s, with excerpts containing trumpet, saxophone and piano parts; (iii) **High Energy** (Hi-NRG) dance music from the 1980s, typically with continuous eighth note bass lines, aggressive synthesizer sounds and staccato rhythms; (iv) **Power Pop**, a Rock style from the 1970s, with chosen songs being guitar-heavy and with male singers; (v) **Rocksteady**, which is a precursor of Reggae with a somewhat soulful basis. The full list of songs can be found here.

The five genres were chosen to have small stylistic overlap. All songs originate between the 1940s and 1980s, making it more unlikely that participants are overly familiar with the music since all of them have been born after 1984. For every genre, we chose 18 songs. To further ensure unfamiliarity of song material to participants, we proceeded as follows. The songs were mainly chosen with the help of the recommendations of similar songs and artists on the music platform Spotify. For each genre, we always started with one stereotypical artist of the genre and then searched for other similar songs using the similar artist function of Spotify, with the goal of finding similar music from rather unknown artists. The criterion for each song's degree of popularity was to have under 50.000 accesses on Spotify. Post-experiment questioning confirmed that very few songs were familiar to the participants. No artists appears more than once on the song list. Within genres, we tried to find a homogeneous set of songs in order to evoke high similarity ratings which are crucial for determination of upper bounds in rater agreement. For presentation in the

questionnaire, 15 seconds of a representative part of every song (usually the refrain) were chosen and normalized to 89db to control for volume effects.

Questionnaire: The study was conducted online at www.soscisurvey.com, which is a free-access platform to compile questionnaires also allowing to include audio files. The first page of the questionnaire contained an introduction that explains the purpose of the study, the expected temporal effort as well as the note that the collection of data is held anonymously. Subsequently, the procedure of the study was explained to the participants. The subjects were asked to "assess the similarity between the query song and each of the five candidate songs by adjusting the slider" and "to answer intuitively since there are no wrong answers". Before starting with the assessment, a test page was shown consisting of one randomly chosen query song and five randomly chosen candidate songs of all five genres. That was done to introduce all five genres and to give an idea of the variation of the song material used in the study.

For the main part of the questionnaire the pairings of query and candidate songs were determined as follows. The complete song material consists of excerpts of 90 songs with a duration of 15 seconds, with 18 songs belonging to each of the 5 genres. We randomly drew 3 songs of each genre as query songs yielding a total of 15 query songs. For every query song we randomly chose five candidate songs with the constraint that at least one of them belongs to the same genre as the corresponding query song. This yields 15 groups of 6 songs each, with the sequential order being held constant for all participants. Each group contains one query song paired with each of the five candidate songs of the group. In sum, comparisons of five pairs had to be made for every group yielding a total of $15 \times 5 = 75$ pairs. The participants were asked to indicate their rating of the similarity on a slider ranging from 0 to 100 %. The more similar a pair was assessed, the higher the percentage was, and the further to the right the slider had to be shifted.

At the end of the questionnaire, data regarding gender, age and musical education and experience was collected. On the last page of every questionnaire, the subjects had the possibility to leave a comment.

About two weeks after filling in the first questionnaire at time point **t1**, all subjects filled in the same questionnaire with identical randomized items a second time (time point **t2**). The test page as well as the questions about the personal experiences with music were omitted in the second round of surveys.

4. RESULTS

First we analyse the degree of **inter-rater agreement** by computing the Pearson correlation ρ between graders for the 75 pairs of query/candidate songs. The correlations between graders S1 to S6 are given in Table 2 for time t1 and in Table 3 for time t2. The correlations range from 0.59 to 0.86, with an average of 0.73 at t1 and 0.75 at t2 (see also Table 1 for an overview of results). This is considerably higher than $\rho^{AMS} = 0.40$ which has been reported for the MIREX AMS task 2006 [6]. The differences in correlation between ρ^{AMS} and correlations in our experiment are also statistically significant at both t1 ($|t| = |8.3322| > t_{95,df=16} = 2.120$) and t2 ($|t| = |8.8519| > t_{95,df=16} = 2.120$). Therefore the inter-rater agreement over the full range of scores in our experiment is increased compared to the MIREX AMS task.

	t1	t2	t1 → t2
ρ	$0.73 \pm .065$	$0.75 \pm .065$	$0.80 \pm .103$
B_{80}	67.7 ± 19.5	57.5 ± 25.6	82.1 ± 14.6
ρ^{AMS}	$0.40 \pm .027$		
B_{80}^{AMS}	61.65 ± 27.0		

Table 1. Overview of results for time points t1, t2 and between t1 and t2 (t1 → t2). Shown are average correlations ρ and upper bounds $B_{80} \pm$ standard deviations, also for MIREX AMS task (last two lines).

Next we explore the level of agreement for specific intervals of scores. We plot the average score of a rater i for all query/candidate pairs, which he or she rated within a certain interval of scores v, versus the average scores achieved by the other five raters $j \neq i$ for the same query/candidate pairs. We therefore explore how human graders rate pairs of songs which another human grader rated at a specific level of similarity. The average results across all raters and for intervals v ranging from $[0, 1], (1, 2]$... to $(9, 10]$ are plotted in Figure 1 for t1 and in Figure 2 for t2. It is evident that there is a certain deviation from the theoretical perfect agreement which is indicated as a dashed line, especially for time t2. Pairs of query/candidate songs which are rated as being very similar (score between 90 and 100) by one grader are on average only rated at around 72.9 by the five other raters at t1 and at only 55.17 at t2. On the other end of the spectrum, query/candidate pairs rated as being not similar at all (score between 0 and 10) receive average scores of 17.4 (t2) and 16.1 (t2) by the respective other raters.

In a previous study [6] an upper bound B^{AMS} has been reported based on scores within the highest interval. Since for our experiment only 4 (out of 75 song pairs × 6 raters = 450) scores are higher than 90 for t1 and only 15 for t2, we compute a broader upper bound based on all scores between 80 and 100. There are 37 such scores for t1 and 47 for t2. This upper bound B_{80} is 67.7 for t1 and 57.5 for t2 (see Table 1). If we multiply scores from AMS 2006 by ten for better comparability and compute a similar upper bound B_{80}^{AMS} this yields 61.65. Since our upper bounds B_{80} are either above (t1) or below (t2) the upper bound B_{80}^{AMS}, we have to conclude that our experiment was not able to raise the upper bound in modeling general music similarity.

Next we looked at **intra-rater agreement** measured between time points t1 and t2. The Pearson correlation ρ between t1 and t2 for the 75 pairs of query/candidate songs for graders S1 to S6 are given in Table 4. The correla-

	S1	S2	S3	S4	S5	S6
S1	1.00	0.77	0.74	0.72	0.74	0.82
S2		1.00	0.72	0.75	0.62	0.83
S3			1.00	0.70	0.67	0.76
S4				1.00	0.64	0.80
S5					1.00	0.64
S6						1.00

Table 2. Inter-rater correlation at time **t1**, with mean $0.73 \pm .065$ standard deviation.

	S1	S2	S3	S4	S5	S6
S1	1.00	0.79	0.73	0.77	0.74	0.83
S2		1.00	0.73	0.74	0.75	0.86
S3			1.00	0.69	0.69	0.80
S4				1.00	0.59	0.79
S5					1.00	0.73
S6						1.00

Table 3. Inter-rater correlation at time **t2**, with mean $0.75 \pm .065$ standard deviation.

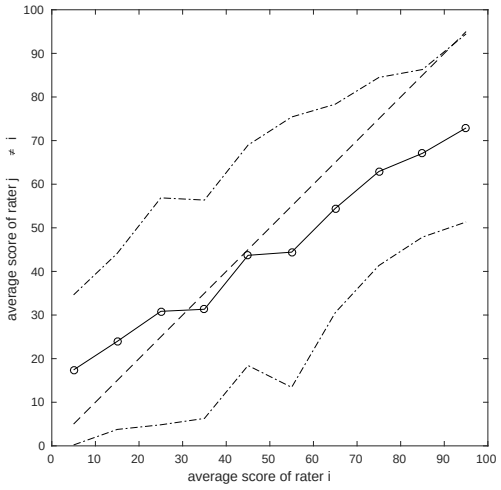


Figure 1. Average score inter-rater agreement for different intervals of scores (solid line) $\pm$ one standard deviation (dash-dot lines) at time **t1**. Dashed line indicates theoretical perfect agreement.

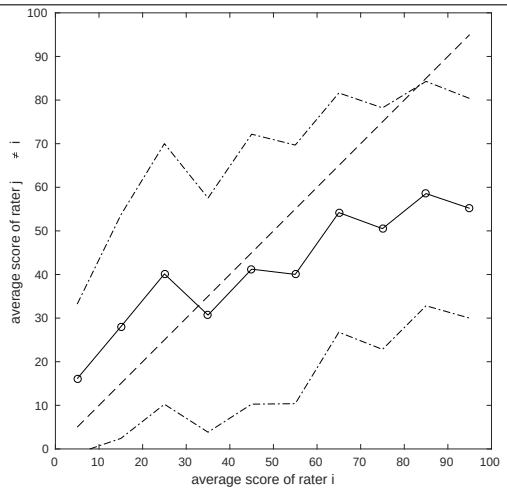


Figure 2. Average score inter-rater agreement for different intervals of scores (solid line) $\pm$ one standard deviation (dash-dot lines) at time **t2**. Dashed line indicates theoretical perfect agreement.

S1	S2	S3	S4	S5	S6
0.81	0.85	0.75	0.81	0.64	0.95

Table 4. Intra-rater correlation between times **t1** and **t2**, with mean $0.80 \pm .103$ standard deviation.

tions range from 0.64 to 0.95, with an average of 0.80, which is somewhat higher than inter-rater correlation of 0.73 and 0.75 at time t1 and t2 (see Table 1). The differences between inter-agreement correlations and intra-agreement correlation are however not statistically significant, neither for t1 ($|t| = |-1.9742| < t_{95,df=19} = 2.093$) nor for t2 ($|t| = |-1.3682| < t_{95,df=19} = 2.093$).

Similar to what we did for inter-rater agreement, we also plotted the average score of a rater i for all query/candidate pairs, which he or she rated within a certain interval of scores v at time t1, versus the average scores achieved by the same rater i for the same query/candidate pairs at time t2 in Figure 3. Compared to Figures 1 and 2, we see better agreement within persons between t1 and t2, with intra-agreement being very close to theoretical perfect agreement (dashed line) for scores ranging from 0 to 50, but also from 90 to 100.

We also computed an upper bound B^{80} based on ratings within the interval $(80, 100]$, where query/candidate pairs that have been rated between 80 and 100 by a grader i at t1 have received an average rating of 82.1 by the same grader i at t2. This is higher than the upper bound for inter-rater agreement at both t1 (67.7) and t2 (57.5). The differences between inter-agreement upper bounds and intra-agreement upper bound are statistically significant, both for t1 ($|t| = |-4.2537| > t_{95,df=220} = 1.960$) and t2 ($|t| = |-5.7121| > t_{95,df=19} = 1.960$) Therefore we conclude that the upper bound within participants measured with a two week time lag is higher then the upper bound based on inter-rater agreement.

Because a number of participants in this study com-

	Soul	Bebop	High Energy	70s Rock	Rocksteady
Soul	**46.9**	16.2	-	38.3	25.1
Bebop	19.3	**73.4**	10.4	6.7	14.3
High Energy	30.4	8.1	**71.2**	32.0	15.5
70s Rock	17.4	-	20.9	**48.2**	11.0
Rocksteady	35.0	-	23.3	13.3	**66.1**

Table 5. Genre confusion matrix at time **t1**, shown are average scores per genre combination.

	Soul	Bebop	High Energy	70s Rock	Rocksteady
Soul	**46.6**	13.5	-	36.4	19.6
Bebop	16.5	**64.7**	10.2	9.5	11.6
High Energy	33.3	5.8	**58.6**	29.8	15.8
70s Rock	18.6	-	16.8	**50.6**	11.0
Rocksteady	26.1	-	15.9	8.8	**62.5**

Table 6. Genre confusion matrix at time **t2**, shown are average scores per genre combination.

mented that the **genre of the songs was an important factor** when evaluating the similarity of songs, we analysed the results with respect to genre also. In Tables 5 and 6 we present genre confusion matrices at times t1 and t2. Just to give one example, the first entry in the first line in Table 5 shows that whenever both query and candidate song were from genre 'Soul', on average such pairs have been judged at 46.9 by the graders. The average score for query songs from 'Soul' and candidate songs from 'Bebop' was 16.2, etc. An entry with a dash ('-') signifies that none such query/candidate pairs existed in our questionnaire. The confusion matrices are not symmetric, since there is a difference whether a song from a certain genre is used as a query or a candidate song.

At both times t1 and t2, average scores in the main diagonals are higher then all off-diagonal entries. This shows that participants indeed rated similarities within genres higher than between genres, at least on average. For both times t1 and t2 average scores are highest within genre 'Bebop', followed by 'Rocksteady' and 'High Energy'. Genre 'Soul' has lowest within genre scores and considerable off-diagonal confusion with e.g. '70s Rock'.

To make clearer how often query/candidate pairs within one genre were rated higher than pairs with mixed genres, we computed average R-precision. In our scenario, for each of the 15 groups of songs (consisting of one query and five candidate songs), R is equal the number of candidate songs with genre identical to the respective query song. For our questionnaire R ranged from 1 to 4. When we order all five candidate songs from highest to lowest score, R-precision is then the fraction of candidate songs with correct genre among the first R candidate songs. Average R-precision across questionnaires of all all six participants is 0.86 for t1 and 0.88 for t2. These high values corroborate self-reports by participants that genre was an important aspect when rating similarity of songs.

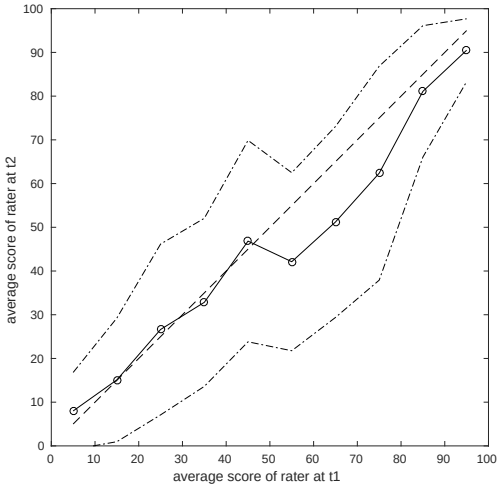


Figure 3. Average score intra-rater agreement for different intervals of scores (solid line) $\pm$ one standard deviation (dash-dot lines) between times **t1** and **t2**. Dashed line indicates theoretical perfect agreement.

5. DISCUSSION

Our principal reason for conducting this research and experiment is the quest to raise the level of rater agreement when judging music similarity. This is needed to quantify the success of computational models of music similarity which are used in automatic music recommendation services. Previous research has criticized the low level of inter-rater agreement and derived an upper bound for algorithms modeling music similarity, which has been reached already in 2009 and has not been outperformed ever since [6]. As a matter of fact, the respective MIREX task (AMS) has been dormant since 2015. In an attempt to improve the AMS task, we have conducted an experiment with a more controlled group of participants and with more carefully curated song material.

The overall inter-rater agreement measured via correla-

tion could indeed be increased compared to the AMS task. However the upper bound, which is only based on higher scores of music similarity, is not improved compared to AMS. Therefore our new version of the AMS task based on more controlled participants and better curated song material is not better suited to measure differences between algorithms that have already reached the AMS upper bound.

One proposal to overcome the problem of upper bounds in measuring music similarity is to personalize models, i.e. to have separate models of music similarity for individual persons. Certainly this is what many commercial services do by offering individual recommendations tailored to their users. This of course brings us to the question how stable assessment of music similarity is within persons, i.e. when the same persons have to judge music similarity repeatedly. The result concerning this intra-rater agreement in our experiment is divided. On the one hand the overall agreement as measured via correlation could not be improved, or at least not sufficiently to allow for statistical significance. On the other hand the upper bound could indeed be raised, opening up the possibility to better measure progress in computational models of personalized music similarity.

For future efforts to improve on the original AMS task design, one should probably rethink which songs from what genre to use. In our experiment, genres were so distinct that at least some participants used membership to a certain genre as the main criterion when assessing similarity between songs. This might impact generalizability of our results when judging music similarity within individual genres. It is also not completely clear, what contribution to improvements our more controlled group of subjects had. We are however convinced that the more uniform group of subjects with a certain musical expertise did lower variation of results. The same holds for the rather young age of participants and the choice of generally not well known song material. Post experiment questioning of participants corroborated that very few songs were known to them, hence less connotations to influence assessment of similarity existed.

One should also point out that this is a pilot study with only six participants. A higher number of test subjects might have allowed for more statistically significant results, e.g. concerning differences between inter and intra-rater correlations. Future work should also explore alternatives to Pearson correlation like generalizations of Kappa measures to the interval scale, which take into account the possibility of rater agreement occurring by chance (e.g. intra-class-correlation [24]). Another open question is whether a time lag of more than two weeks might change results on intra-rater agreement.

A possible route to further improvements is to ask a more specific question than just assessing general music similarity, as in the original AMS task and in the experiment reported in this paper. Criticism of this unclear abstract notion of general music similarity brings us to the concept of 'validity' of our experiment. A valid experiment is an experiment that actually measures what the experimenter intended to measure (see e.g. [27] or [28] for a discussion in relation to MIR). Precisely this intention of the experimenter in the original MIREX AMS task is completely unclear, since it is rather dubious what general music similarity is supposed to mean in the first place. The argument that users apply very different, individual notions of similarity when assessing the output of music retrieval systems has been made before [20]. After all, music similarity is a multi-dimensional notion including timbre, melody, harmony, tempo, rhythm, lyrics, mood, etc, with many of these dimensions meaning different things to different people. It has also been noted before that evaluation of abstract music similarity without reference to a specific usage scenario is not very meaningful [7, 21]. It is therefore our belief that the intention of a music similarity experiment can only be made clearer if it will be tied to a user scenario, e.g. creating a playlist for a specific occasion. Identifying specific use cases has already been advocated as a method for better problem definition [26] in MIR. Previous reviews [13] of user studies in MIR could serve as valuable input for formalization of the use cases.

6. CONCLUSION

We have presented a pilot study aimed at improving experiments to measure general music similarity. By using a more controlled group of subjects and music material from more well defined genres, we were able to improve overall inter-rater agreement but did not succeed in raising an upper bound for models of music similarity, which constitutes an obstructive glass ceiling for any machine learning approach. We did however succeed in raising this upper bound for intra-rater agreement, which corroborates the rationale of personalizing music services. We also discussed the doubtful validity of experiments on general music similarity making it clear that definition of a specific use case might be necessary for conduction of a truly valid experiment. The fact that MIR needs to care much more about the proper design of its experiments is also the main insight going beyond the scope of this paper. Although a small but growing number of publications concerning design and evaluation of MIR experiments exists, they have so far not been able to change the research culture of MIR as a whole.

Acknowledgements: This work was supported by the Austrian Science Fund (FWF P31988) and the Vienna Science and Technology Fund (WWTF MA14-018).

7. REFERENCES

[1] Aucouturier J.J.: Sounds like teen spirit: Computational insights into the grounding of everyday musical terms, in Minett J.W., Wang W. S-Y. (eds.): Language, Evolution and the Brain, City University of Hong Kong Press, pp. 35-64, 2009.

[2] Balke S., Driedger J., Abeßer J., Dittmar C., Müller, M.: Towards Evaluating Multiple Predominant Melody

Annotations in Jazz Recordings, in Proc. of the 17th Int. Society for Music Information Retrieval Conference, pp. 246-252, 2016.

[3] Bosch J., Gómez E.: Melody extraction in symphonic classical music: a comparative study of mutual agreement between humans and algorithms, in Proc. of the 9th Conference on Interdisciplinary Musicology, 2014.

[4] Cuadra C., Katter R.: Opening the black box of "relevance", J. of Documentation, 23(4), 291-303, 1967.

[5] Downie J.S.: The Music Information Retrieval Evaluation eXchange (MIREX), D-Lib Magazine, Volume 12, Number 12, 2006.

[6] Flexer A., Grill T.: The Problem of Limited Inter-rater Agreement in Modelling Music Similarity, J. of New Music Research, Vol. 45, No. 3, pp. 239-251, 2016.

[7] Hu, X., Liu J.: Evaluation of music information retrieval: Towards a user-centered approach, in Proceedings of the 4th Workshop on Huan-Computer Interaction and Information Retrieval, pp. 111-114, 2010.

[8] Jones M.C., Downie J.S., Ehmann A.F.: Human similarity judgments: Implications for the design of formal evaluations, in Proc. of the 8th Int. Conference on Music Information Retrieval, pp. 539-542, 2007.

[9] Klien V., Grill T., Flexer, A.: On automated annotation of acousmatic music, J. of New Music Research, 41(2), 153-173, 2012.

[10] Koops H.V.: Computational modelling of variance in musical harmony, PhD thesis, University of Utrecht, The Netherlands, 2019.

[11] Koops H.V., de Haas W.B., Bransen J., Volk A.: Automatic chord label personalization through deep learning of shared harmonic interval profiles, Neural Computing and Applications, 2018.

[12] Koops H.V., de Haas W.B., Burgoyne J.A., Bransen J., Kent-Muller A., Volk A.: Annotator subjectivity in harmony annotations of popular music, J. of New Music Research, 48:3, 232-252, 2019.

[13] Lee J.H., Cunningham S.J.: Toward an understanding of the history and impact of user studies in music information retrieval, J. of Intelligent Information Systems, 41, pp. 499-521, 2013.

[14] Ni Y., McVicar M., Santos-Rodriguez R., De Bie T.: Understanding effects of subjectivity in measuring chord estimation accuracy, IEEE Transactions on Audio, Speech and Language Processing, 21(12):2607-2615, 2013.

[15] Nieto O., Farbood M.M., Jehan T., and Bello, J.P.: Perceptual analysis of the f-measure for evaluating section boundaries in music, in Proc. of the 15th Int. Society for Music Information Retrieval Conference, pp. 265-270, 2014.

[16] Panteli M., Rocha B., Bogaards N., Honingh A.: A model for rhythm and timbre similarity in electronic dance music, Musicae Scientiae, 21(3), 338-361, 2017.

[17] Quinton E., Harte C., Sandler M.: Extraction of metrical structure from music recordings, in Proc. of the 18th Int. Conference on Digital Audio Effects, 2015.

[18] Salamon J., Gómez E., Ellis D.P.W., Richard G.: Melody extraction from polyphonic music signals: Approaches, applications, and challenges, IEEE Signal Processing Magazine, 31(2):118-134, 2014.

[19] Schamber L.: Relevance and Information Behavior, Annual Review of Information Science and Technology, 29:3-48, 1994.

[20] Schedl M., Flexer A., Urbano J.: The neglected user in music information retrieval research, Journal of Intelligent Information Systems, 41(3), pp. 523-539, 2013.

[21] Serrá X., Magas M., Benetos E., Chudy M., Dixon S., Flexer A., Gomez E., Gouyon F., Herrera P., Jorda S., Paytuvi O., Peeters G., Schlüter J., Vinet H., Widmer G., *Roadmap for Music Information ReSearch*, Peeters G. (ed.), 2013, Creative Commons BY-NC-ND 3.0 license, ISBN: 978-2-9540351-1-6.

[22] Serrà J., Müller M., Grosche P., Arcos J.L.: Unsupervised music structure annotation by time series structure features and segment similarity, IEEE Transactions on Multimedia, 16(5): 1229-1240, 2014.

[23] Seyerlehner K., Widmer G., Knees P.: A Comparison of Human, Automatic and Collaborative Music Genre Classification and User Centric Evaluation of Genre Classification Systems, in Proc. of the 8th Int. Workshop on Adaptive Multimedia Retrieval, pp. 118-131, 2010.

[24] Shrout P.E., Fleiss J.L.: Intraclass correlation: Uses in assessing rater reliability, Psychological Bulletin, 86, 420-428, 1979.

[25] Smith J.B., Chew E.: A meta-analysis of the mirex structure segmentation task, in Proceddings of the 14th International Society for Music Information Retrieval Conference, pp. 45-47, 2013.

[26] Sturm B.L.: The state of the art ten years after a state of the art: Future research in music information retrieval, J. of New Music Research, 43 (2), pp. 147-172, 2014.

[27] Trochim, W.: The Research Methods Knowledge Base, 2nd edn, Atomic Dog Publishing, Cincinnati, OH, 2000.

[28] Urbano J., Schedl M., Serra X.: Evaluation in music information retrieval, J. of Intelligent Information Systems, 41 (3), pp. 345-369, 2013.

[29] Wiggins G.: Semantic Gap?? Schemantic Schmap!! Methodological Considerations in the Scientific Study of Music, in Proc. of the 11th IEEE Int. Symposium on Multimedia, pp. 477-482, 2009.

AIST DANCE VIDEO DATABASE: MULTI-GENRE, MULTI-DANCER, AND MULTI-CAMERA DATABASE FOR DANCE INFORMATION PROCESSING

Shuhei Tsuchida **Satoru Fukayama** **Masahiro Hamasaki** **Masataka Goto**
National Institute of Advanced Industrial Science and Technology (AIST), Japan
{s-tsuchida, s.fukayama, masahiro.hamasaki, m.goto}@aist.go.jp

ABSTRACT

We describe the *AIST Dance Video Database (AIST Dance DB)*, a shared database containing original street dance videos with copyright-cleared dance music. Although dancing is highly related to dance music and dance information can be considered an important aspect of music information, research on dance information processing has not yet received much attention in the Music Information Retrieval (MIR) community. We therefore developed the AIST Dance DB as the first large-scale shared database focusing on street dances to facilitate research on a variety of tasks related to dancing to music. It consists of 13,939 dance videos covering 10 major dance genres as well as 60 pieces of dance music composed for those genres. The videos were recorded by having 40 professional dancers (25 male and 15 female) dance to those pieces. We carefully designed this database so that it can cover both solo dancing and group dancing as well as both basic choreography moves and advanced moves originally choreographed by each dancer. Moreover, we used multiple cameras surrounding a dancer to simultaneously shoot from various directions. The AIST Dance DB will foster new MIR tasks such as dance-motion genre classification, dancer identification, and dance-technique estimation. We propose a dance-motion genre-classification task and developed four baseline methods of identifying dance genres of videos in this database. We evaluated these methods by extracting dancer body motions and training their classifiers on the basis of long short-term memory (LSTM) recurrent neural network models and support-vector machine (SVM) models.

1. INTRODUCTION

The Music Information Retrieval (MIR) community started with standard music information such as musical audio signals and musical scores, and then extended its scope to other types of music-related multimodal information such as images, videos, lyrics, and social-media data. Since dance music – a popular target of MIR research [7, 19, 20, 22, 33, 40, 42, 46, 55, 61, 66, 80] – is originally written for dance and has a strong connection to dance motions, *dance information*, that is, any kind of information related to dance, such as dance music, dance motions, and dancers, can be considered an important aspect of music information that the MIR community should cover. Research on dance motions, however, has not yet received much attention in the community. The goal of this research is to develop a dance information database including original dance videos with copyright-cleared dance music and make it publicly available to the community so that research on dance can attract more attention in the community and new MIR tasks related to dance can emerge in the future.

As a sub-area of MIR research, we propose defining *dance information processing* as meaning various types of processing and research related to dance information.

Figure 1. Snapshots of AIST Dance Video Database (AIST Dance DB), which features **multiple genres** (10 major dance genres), **multiple dancers** (solo and group dancing by 40 professional dancers), and **multiple cameras** (at most 9 video cameras surrounding a dancer). These color snapshots are cropped to enlarge dancers in videos.

First, it is necessary for dance information processing to handle dance music. There have been studies on traditional dance tunes [7, 22, 40, 61], electronic dance music [19, 42, 46, 55, 66, 80, 82], and dance-music classification [20, 33]. Second, it is important for dance information processing to advance research related to dance motions. There have been various related studies such as on dance-motion choreography generation driven by music [2, 28, 29, 32, 52, 53, 73], rhythm estimation from dance motions in dance videos [18], music retrieval by dance motions [75], controlling music tempo by dance motions [37], dance-motion identification [60], and dance-motion genre classification [43]. In addition, research on dance motions could have good synergy with research on performer motions made during musical performances [30, 47, 48, 51, 67] since both deal with music-related human body movements. This emerging field of dance information processing, however, has lacked a large systematic dance information database that is available to researchers for common use and research purposes.

We therefore built the *AIST Dance Video Database (AIST Dance DB)*, the first large-scale copyright-cleared database focusing on street dances (Figure 1). The AIST Dance DB consists of 13,939 original dance videos covering 10 major street dance genres (break, pop, lock, waack, middle hip-hop, LA-style hip-hop, house, krump, street jazz, and ballet jazz) and 60 original pieces of dance music composed for these genres, each having 6 different musical pieces with different tempi. We had 40 professional dancers (25 male and 15 female), each having more than 5 years of dance experience, dance to those pieces. In addition to 13,890 videos for which each of the 10 genres has 1,380 videos, we added 49 videos that cover 3 typical dancing situations (showcase, cypher, and battle) in which a group of dancers enjoy dancing. The database is carefully designed to cover both solo dancing (12,990 videos) and group dancing (949 videos) as well as both basic choreography moves (10,800 videos) and advanced moves (3,139 videos) originally choreographed by each dancer. We used at most nine video cameras surrounding a dancer to simultaneously shoot from various directions.

To the best of our knowledge, such street dance videos have not been available to researchers, so they will become valuable research materials to be analyzed in diverse ways and used for various machine-learning purposes. For example, the AIST Dance DB will foster new MIR tasks such as dance-motion genre classification, dancer identification, and dance-technique estimation. As a basic task of dance information processing, we propose a dance-motion genre-classification task for street dances. We developed four baseline methods for identifying dance genres of a subset of the dance videos in this database and evaluated them by extracting dancer body motions and training their classifiers on the basis of long short-term memory (LSTM) recurrent neural network models and support vector machine (SVM) models. In our preliminary experiments, we tested the methods on 210 dance videos of 10 genres done with 30 dancers and found that dance genres were identified with a high accuracy of 91.4% when a 32-sec excerpt was given; however, the accuracy dropped to 56.6% when a 0.67-sec excerpt was given. These results can be used as a baseline performance for this task.

2. RELATED WORK

Since the importance of research databases has been widely recognized in various research fields, researchers interested in dance have also spent considerable effort on building dance-related databases [62]. For example, the Martial Arts, Dancing and Sports Dataset [82] has six videos for both the hip-hop and jazz dance genres. These videos were shot from three directions, and the video data includes depth information. However, six videos for each genre is not enough for some tasks, and the videos do not contain enough professional dance motions to effectively express the characteristics of each genre. Several relatively small dance datasets have also been published [14, 17, 63, 79].

Some databases not only have dance videos but also provide different types of sensor data. Stavrakis et al. [70] published the Dance Motion Capture Database, which provides high-quality motion capture data as well as dance videos. This database includes Greek and Cypriot dances, contemporary dances, and many other dances such as flamenco, the belly dance, salsa, and hip-hop. Tang et al. [72] also used a motion capture system to build a dance dataset that contains four dance genres: cha-cha, tango, rumba, and waltz. Essid et al. [23] published a dance dataset that consists of videos of 15 salsa dancers, each performing 2 to 5 fixed choreographies. They were captured using a Kinect camera and five cameras. This dataset is unique since all video data contain inertial sensor (accelerometer + gyroscope + magnemeter) data captured from multiple sensors on the dancers' bodies. All these databases, however, do not handle street dance videos performed by multiple dancers and recorded with multiple camera directions.

Although YouTube-8M [1] and Music Video Dataset [65] were not built for research on dance, the former contains 181,579 dance videos and the latter contains 1,600 music videos including dance-focused videos. It is difficult to use these videos for dance information processing because the videos are not organized from this viewpoint. In comparison, the AIST Dance DB contains videos that are systematically recorded and organized to have attributes such as dance genre names, the names of basic dance moves, labels of dance situations, and information on dancers and musical pieces.

3. AIST DANCE VIDEO DB

3.1 Design policy

We designed the AIST Dance DB by considering the following three important points.

- **Suite of video files of original dance and audio files of copyright-cleared dance music**
  To analyze and investigate the relationships between a musical piece and the dance motions that go along

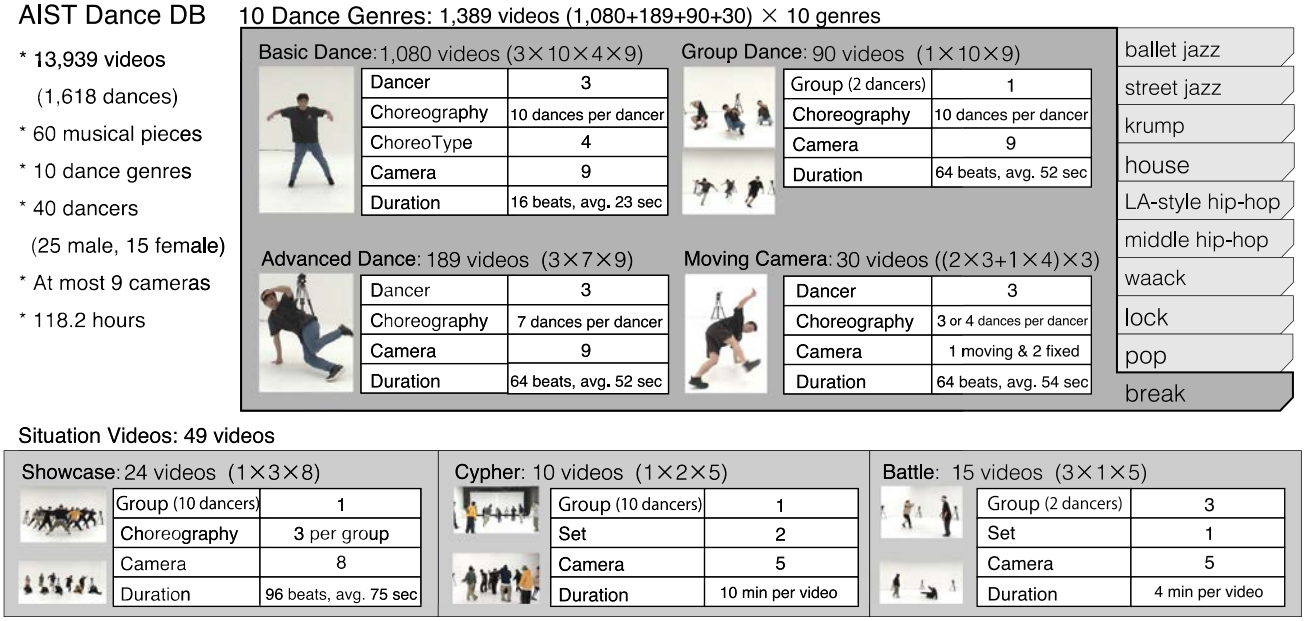


Figure 2. Overview of AIST Dance DB. Each of 10 dance genres has 1,389 videos that consist of 4 categories: **Basic Dance** (120 dances in 1,080 videos for basic genre-specific dance moves with 4 impressions (ChoreoType): intense, loose, hard, and soft), **Advanced Dance** (21 dances in 189 videos for advanced dance moves originally choreographed by each individual dancer), **Group Dance** (10 dances in 90 videos for group dance done with 3 dancers having different original choreographies), and **Moving Camera** (10 dances in 30 dolly-shot videos with moving camera). Other 49 Situation Videos consist of **Showcase** (3 dances in 24 videos assuming stage dance performances done by 10 dancers in front of audiences), **Cypher** (2 dances in 10 videos where 10 dancers line up in a circle and keep dancing in turns), and **Battle** (3 dances in 15 videos where 2 dancers face each other and dance).

with the piece, we first asked professional musicians to create pieces of dance music in different genres for our database, then recorded dance videos in which professional dancers danced while listening to one of the musical pieces. As a result, each video includes not only dance motions but also the musical piece used as background music.

- **Variety of dance genres and choreographies**
  Since processing diverse dance information would require a variety of genres and choreographies, we ensured such a variety by including 10 dance genres, 40 male and female dancers, different numbers of dancers (solo dancing and group dancing), different choreographies, and different levels of difficulty in choreography (basic choreography moves and original advanced moves).
- **Shooting from various directions**
  To analyze the same dance motions from different views and angles, we used multiple video cameras surrounding a dancer so that the cameras can simultaneously shoot from various directions. Even if some body parts cannot be seen from the front camera, they can be seen from the back camera.

3.2 Contents

An overview of the AIST Dance DB is shown in Figure 2. We built it on the basis of the design policy discussed in Section 3.1. The AIST Dance DB consists of 1,618 street dances in 13,939 videos: 13,890 videos of 10 street dance genres and an additional 49 *Situation Videos* of 3 different situations. The dance genres were decided in consultation with experts on street dances and divided into *Old School* styles (break, pop, lock, and waack), which are dance styles from about the 1970s to 1990s, and *New School* styles (middle hip-hop, LA-style hip-hop, house, krump, street jazz, and ballet jazz), which are dance styles since about the 1990s.

The database also includes 60 musical pieces that are categorized into 10 dance genres. The tempi of the 6 pieces for each genre except for house were set to 80, 90, 100, 110, 120, and 130 beats per minute (BPM); the tempi of the 6 pieces for house were set to 110, 115, 120, 125, 130, and 135 BPM since slow tempi are not fitting for house.

To cover choreographic variations within dance genres, we had 40 professional dancers (25 male and 15 female) participate in video recordings in a professional studio. At least three dancers were assigned to each genre. All dancers had more than 5 years of dance experience. All videos were recorded in full color, although dancers mainly wore monotone clothing when dancing.

For each of the 10 dance genres, we recorded a total of 1,380 videos consisting of 1,080 basic choreography dance videos, 189 advanced choreography dance videos, 90 group dance videos, and 30 dance videos with a moving camera for dolly-in and dolly-out shots. All camera positions were fixed except for the moving camera. As shown in Figure

2, we also recorded 49 *Situation Videos* that consist of 24 videos for showcase, 10 videos for cypher, and 15 videos for battle. Dancers were asked to choreograph their dance to fit the given genre. Basically, each choreography was shot in one take; however, it was taken again when there was a clear mistake. The location of the front camera was designed carefully to capture the full body of the dancer, and the other cameras were located to capture as much of the dancer's body as possible except for group dance. This database is available at `https://aistdancedb.ongaaccel.jp`.

4. DANCE-MOTION GENRE CLASSIFICATION

To illustrate the utility of the AIST Dance DB, we tackled the dance-motion genre-classification task. Since genre classification of music is a popular research topic in the MIR community, genre classification of dance motions could be a good starting point, which would also have applications such as personalized dance-video recommendation.

We developed four baseline methods to provide baseline results. We investigated four research questions the answers of which could contribute to research on dance-motion genre classification: (RQ1) *"Can we classify the 10 genres by using their video frames only?"*, (RQ2) *"How many video frames should be used to train a model?"*, (RQ3) *"Is the ease of classification different by dance genre?"*, and (RQ4) *"Can beat positions help improve classification accuracy?"*.

4.1 Experimental Conditions

As a dataset for our experiments, we created a subset of the advanced choreography dance videos. The dataset consists of 210 dance videos shot from the front camera only and covers the 10 dance genres. Each genre has 21 dance videos by 3 dancers, each of whom uses 7 original choreographies. In total, the dataset covers 210 different choreographies by 30 different dancers. The musical piece in each video has 64 beats (4 beats $\times$ 16 measures in four-four time), where the term "beat" denotes a quarter note.

We split 210 dance videos into a training set (126 videos), a validation set (14 videos), and a test set (70 videos). For each genre, 14 videos by two dancers were used for the training and validation sets, and 7 videos by the remaining dancer was used for the test set. Every dancer and every choreography in the training and validation sets thus does not appear in the test set.

4.2 Methods

An overview of our baseline methods is shown in Figure 3. Each method is trained to classify an excerpt of the input video into one of the 10 dance genres. In the first step of motion-feature extraction, we use the OpenPose library [12] to estimate the dancer's skeleton (body pose and motion) in all video frames (60 frames per second). This can reduce the dependency on the AIST Dance DB since the estimated body pose and motion do not have original RGB pixel information.

Since both pose and motion are important elements that characterize dancing, we obtain dancer poses by calculating 21 joint angles from the skeleton per video frame. Each joint angle is then converted into two dimensional values θ_x and θ_y by calculating the sine and cosine of the angles to make the distance calculation between angular values easier. As a result, we convert the 21-dimensional angular values into a 42-dimensional feature vector. Let us represent this feature vector at the n-th frame of the i-th video as $v_\theta^{(i)}(n)(1 \leq n \leq N^{(i)}\ and\ 1 \leq i \leq I)$, where $N^{(i)}$ is the number of frames in the i-th video and I is the number of videos in the dataset. When some joints in the video have not been detected, we substitute zeros for the values that correspond to the undetected joints. Our methods then represent the body motions by calculating the velocity and acceleration between frames. The velocity $v_{\Delta\theta}^{(i)}(n)$ and acceleration $v_{\Delta^2\theta}^{(i)}(n)$ at the n-th frame are calculated as follows:

$$v_{\Delta\theta}^{(i)}(n) = v_\theta^{(i)}(n) - v_\theta^{(i)}(n-1), \tag{1}$$

$$v_{\Delta^2\theta}^{(i)}(n) = v_{\Delta\theta}^{(i)}(n) - v_{\Delta\theta}^{(i)}(n-1). \tag{2}$$

We then concatenate the above three: $v_\theta^{(i)}(n)$, $v_{\Delta\theta}^{(i)}(n)$, and $v_{\Delta^2\theta}^{(i)}(n)$, into one 126-dimensional vector $v^{(i)}(n)$.

In the second step, we aggregate the 126-dimensional vectors representing body motions within a unit (temporal interval) determined if using beat positions or not. All beat positions are automatically determined by the tempo of each musical piece. Below are the details of the two methods:

- Adaptive method: when using beat positions, the vectors are aggregated within four kinds of tempo-dependent variable length units: we used one, two, three, or four beats as one unit. The length L_a corresponding to a beat ranges from 27 to 45 video frames as the tempo ranges from 80 to 135.
- L-fixed method: the vectors are aggregated within various fixed-length units. The length L of one unit is 20, 40, 60, ..., or 500 video frames.

Each method calculates a unit-level feature vector for every unit in the video. It calculates the mean vector and standard deviation vector (126 dimensions each) among body motions from the n_{start}-th to n_{end}-th video frames of every unit and concatenates those vectors into the 252-dimensional unit-level vector of the k-th unit $\tilde{v}^{(i)}(k)$.

In the third step, each method calculates a window-level feature vector. We use five different window lengths to aggregate unit-level feature vectors into a window-level one to see how many video frames are necessary to identify a dance genre. The window-level feature vector is obtained by concatenating all unit-level vectors $\tilde{v}^{(i)}(k)$ within a window of 2, 4, 8, 16, and 32 units, respectively. The window is then shifted by 1 unit to obtain the next window-level feature vector. We thus obtain five different window-level feature vectors.

In the fourth step, we prepare four baseline methods by combining adaptive or L-fixed methods with LSTM-based or SVM-based models. Each method classifies every

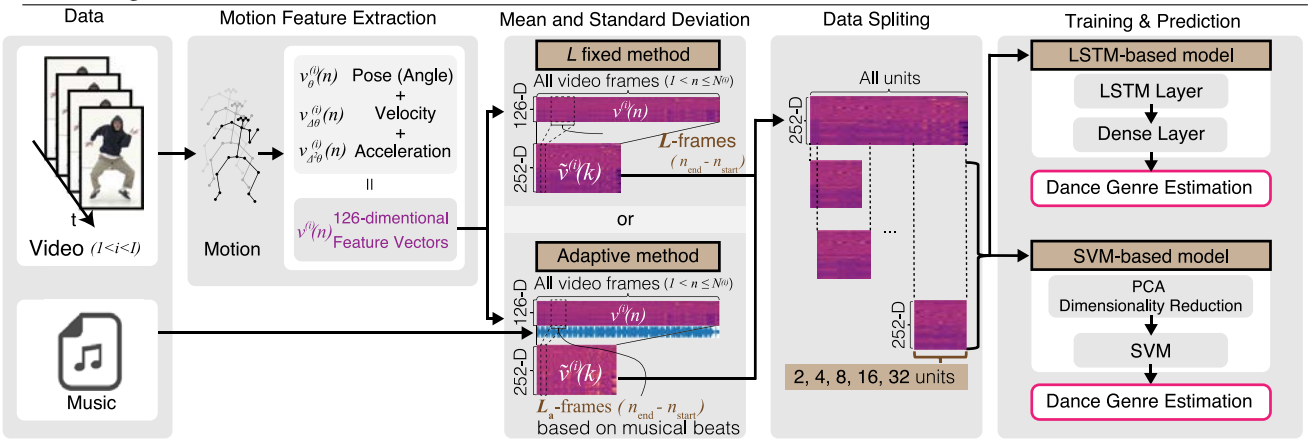


Figure 3. Overview of four baseline methods for dance-motion genre-classification task: (1) L-fixed method with LSTM-based model, (2) L-fixed method with SVM-based model, (3) adaptive method with LSTM-based model, and (4) adaptive method with SVM-based model.

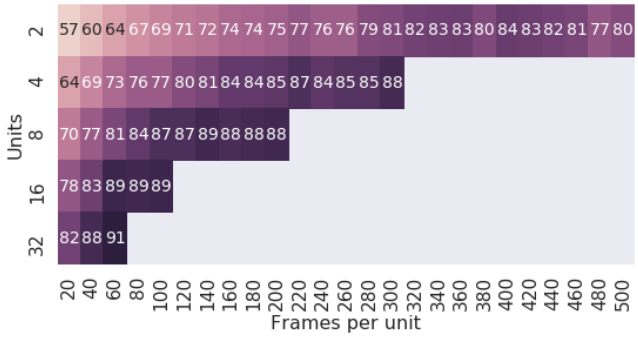


Figure 4. Comparison of the genre-classification accuracy of the L-fixed method with the LSTM-based model with regard to different combinations of the number of frames per unit and the number of units. Blank indicates that combined size of frames exceeds the length of video.

window-level feature vector into the 10 dance genres. For the LSTM-based model, we use a bi-directional recurrent neural network (RNN) with one layer of LSTM [39] cell. The network outputs a 10-dimensional one-hot vector representing dance genres. A rectified linear unit activation function is applied to the output of the LSTM. Batch normalization is applied to the output layer of the dense layers. We use cross entropy as the loss function and a batch size of 10. We train this model with a learning rate of $5e-4$ through 100 epochs and record the trained model at the minimum validation loss. This model is implemented in PyTorch [57]. For the SVM-based model, we first obtain 200-dimensional vectors by using principal component analysis to reduce the dimension of the training data, then train the SVM model. Finally, we estimate the dance genre of every window-level feature vector in a video by using these two models.

5. RESULTS

To answer RQ1 in Section 4, we investigated the accuracy of dance-motion genre classification using three-fold cross-validation (each fold used a different dancer for the test set). We first calculated the ratio of the correct estimation for every dance genre, then averaged over genres to obtain the genre-classification accuracy. The best genre-classification accuracy was 91.4% when we used the L-fixed method with the LSTM-based model where the number of frames was 60 and the number of units was 32. In this dataset, we found that dance genres can be estimated with relatively high accuracy. In the case of the L-fixed method with the SVM-based model, however, the best accuracy was dropped to 84.0%.

To answer RQ2 in Section 4, we analyzed the genre-classification accuracy when the number of frames per unit and the number of units were changed as shown in Figure 4. We found that dance-motion genre classification can be executed with an accuracy of 56.6% by using only 0.67 sec corresponding to 40 frames (20 frames per unit $\times$ 2 units) of a video. This was much shorter than we expected.

To answer RQ3 in Section 4, we conducted an analysis by creating a confusion matrix for the L-fixed method with the LSTM-based model and found that krump is relatively easy to estimate and house is relatively difficult to estimate. We also found that the estimation performance depends on the number of frames per unit and the number of units to calculate the input to the classifiers. With a small number of frames and units, street jazz and ballet jazz were easily confused by the classifier and the estimation accuracy of house dropped. This can be understood from the fact that street jazz and ballet jazz contain similar poses and house contains many movements that are commonly found in other dance genres, such as simple lateral movements.

To answer RQ4 in Section 4, we confirmed that the highest accuracy of the adaptive method using musical beats was 83.4% and that of the L-fixed method was 91.4%, both with the LSTM-based model. In the case of the adaptive method with the SVM-based model, the best accuracy was further dropped to 80.7%. In this way, the preliminary answer to RQ4 was not positive. Since there would be much room for improvement when using beat positions, we leave this for future research.

6. DISCUSSION

6.1 Dance information processing

As shown in Figure 5, dance information processing can be classified into four categories: (a) dance-motion analysis, (b) dance-motion generation, (c) dance-music analysis, and (d) dance-music generation. The goal of (a) dance-motion analysis is to automatically analyze every aspect of dance motions including dance-motion genre classification, dancer identification, dance-technique estimation, structural analysis of choreographies, and dance-mood analysis. A typical research topic of (b) dance-motion generation is to automatically generate motions for dance robots and computer-graphics dancers so that their motions can be natural and indistinguishable from human motions. As described at the beginning of this paper, research related to dance music, including (c) dance-music analysis and (d) dance-music generation, has been popular in the MIR community. There is still room for improvements regarding analyzing and classifying dance music in more depth and generating dance music in various styles and for various purposes.

Furthermore, our research community could investigate various interactions between those categories as well as advance each of the four categories. For example, we could combine (a) dance-motion analysis and (c) dance-music analysis. Analyzed dance motions would be helpful for structural analysis of musical pieces in dance music videos. Analyzed music structure could be used to analyze dance motions in a context-dependent manner. Another interesting topic of research is to find musical pieces suitable for dance motions, which will be useful for developing automatic DJ systems that recommend musical pieces suitable for various dancers on dance floors and at dance events. Analyzing the relationships between dance motions and music in existing dance videos is also important to develop systems for assisting people in creating and editing more attractive music-synchronized videos. We could also combine (a) dance-motion analysis and (b) dance-motion generation. If three-dimensional dance motions can be accurately extracted from a large collection of existing dance videos, they could be useful for automatic dance-motion-generation systems based on machine learning. If dance styles of dancers can be analyzed and modeled, it could become possible to transfer those styles to artificial computer-graphic dancers or robot dancers.

Dance information processing will naturally use multimodal dance information for different research topics. There have been many related studies such as on dance-practice support [3, 8, 11, 15, 16, 24, 36, 49, 68, 71], choreography-creation support [25, 59, 81], dance-performance augmentation [6, 9, 10, 13, 26, 27, 31, 38, 45, 56, 76, 78], dance-group support and analysis [35, 50, 54, 69, 77], dance archive [44, 58, 64, 83], dance performance alignment [21, 34], dance video editing [5, 41, 74], and dance-style transfer [4]. We look forward to advances in this emerging research field of dance information processing.

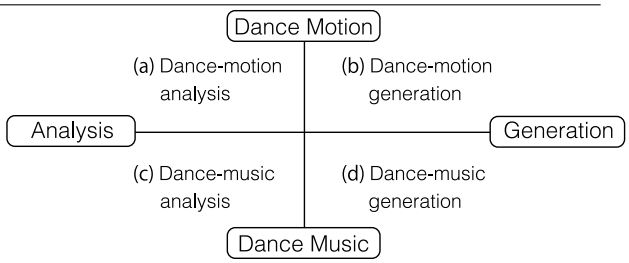


Figure 5. Overview of dance information processing.

6.2 AIST Dance Video Database

We believe that the AIST Dance DB will contribute to the advancement of dance information processing as other research databases have also contributed to the advancement of their related research. In particular, this database will advance research on dance-motion analysis (Figure 5). In addition to the dance-motion genre-classification task we discussed in this paper, it is possible to develop systems that can classify dance videos into advanced dance, basic dance, moving camera, and group dance, as shown in Figure 2. Since various dancers dance to the same musical pieces in this database, their individual differences can be analyzed in depth, and such analyzed results could be useful in developing dancer-identification systems. The basic dance videos could be useful for analyzing subtle individual differences of motions since all the dancers have the same basic motions. By using videos recorded from different directions, systems that can recognize dance motions from any direction could also be developed. As we illustrated in Section 4.2, using image-processing technologies, such as OpenPose, makes it possible to extract dance motions from dance videos for use in machine learning for various purposes including automatic dance-motion generation.

From the viewpoint of the MIR community, it is essential for the AIST Dance DB to include 60 pieces of dance music in synchronization with dance motions. This will lead to various research topics such as dance-music classification with or without using dance motions, dance-motion classification with or without using dance music, and detailed multimodal analysis of the correlation between dance motion and music. Furthermore, since this database is publicly available, it can be used for designing benchmarks of evaluating technologies.

7. CONCLUSION

The main contributions of this work are threefold: 1) we built the first large-scale shared database containing original street dance videos with copyright-cleared dance music, 2) we proposed and discussed a new research area *dance information processing*, and 3) we proposed a dance-motion genre-classification task and developed four baseline methods. We hope that the AIST Dance DB will help researchers develop various types of dance-information-processing technologies to give academic, cultural, and social impact.

8. ACKNOWLEDGMENTS

This work was supported in part by JST ACCEL Grant Number JPMJAC1602, Japan.

9. REFERENCES

[1] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[2] O. Alemi, J. Françoise, and P. Pasquier. Groovenet: Real-time music-driven dance movement generation using artificial neural networks. In *Workshop on Machine Learning for Creativity, 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining, SIGKDD 2017*, volume 8, page 26, 2017.

[3] D. S. Alexiadis, P. Kelly, P. Daras, N. E. O'Connor, T. Boubekeur, and M. B. Moussa. Evaluating a dancer's performance using kinect-based skeleton tracking. In *Proceedings of the 19th ACM International Conference on Multimedia, MM 2011*, pages 659–662, 2011.

[4] A. Aristidou, Q. Zeng, E. Stavrakis, K. Yin, D. Cohen-Or, Y. Chrysanthou, and B. Chen. Emotion control of unstructured dance movements. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation, SCA 2017*, pages 9:1–9:10, 2017.

[5] E. Bakala, Y. Zhang, and P. Pasquier. MAVi: A movement aesthetic visualization tool for dance video making and prototyping. In *Proceedings of the 5th International Conference on Movement and Computing, MOCO 2018*, pages 11:1–11:5, 2018.

[6] Louise Barkhuus, Arvid Engström, and Goranka Zoric. Watching the footwork: Second screen interaction at a dance and music performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2014*, pages 1305–1314, 2014.

[7] P. Beauguitte, B. Duggan, and J. D. Kelleher. A corpus of annotated irish traditional dance music recordings: Design and benchmark evaluations. In *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016*, pages 53–59, 2016.

[8] V. Boer, J. Jansen, A. T. Pauw, and F. Nack. Interactive dance choreography assistance. In *Proceedings of the 14th International Conference on Advances in Computer Entertainment Technology, ACE 2017*, pages 637–652, 2017.

[9] C. Brown. Machine tango: An interactive tango dance performance. In *Proceedings of the Thirteenth International Conference on Tangible, Embedded, and Embodied Interaction, TEI 2019*, pages 565–569, 2019.

[10] A. Camurri. Interactive dance/music systems. In *Proceedings of the 21th International Computer Music Conference, ICMC 1995*, pages 245–252, 1995.

[11] A. Camurri, R. K. El, O. Even-Zohar, Y. Ioannidis, A. Markatzi, J. Matos, E. Morley-Fletcher, P. Palacio, M. Romero, A. Sarti, S. D. Pietro, V. Viro, and S. Whatley. WhoLoDancE: Towards a methodology for selecting motion capture data across different dance learning practice. In *Proceedings of the 3rd International Symposium on Movement and Computing, MOCO 2016*, pages 43:1–43:2, 2016.

[12] Z. Cao, T. Simon, S. E. Wei, and Y. Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 7291–7299, 2017.

[13] F. Caputo, V. Mc Gowen, J. Geigel, S. Cerqueira, Q. Williams, M. Schweppe, Z. Fa, A. Pembrook, and H. Roffe. Farewell to dawn: A mixed reality dance performance in a virtual space. In *Proceedings of the ACM SIGGRAPH 2016 Posters, SIGGRAPH 2016*, pages 49:1–49:2, 2016.

[14] D. Castro, S. Hickson, P. Sangkloy, B. Mittal, S. Dai, J. Hays, and I. A. Essa. Let's dance: Learning from online dance videos. *arXiv preprint arXiv:1801.07388*, 2018.

[15] J. C. P. Chan, H. Leung, J. K. T. Tang, and T. Komura. A virtual reality dance training system using motion capture technology. *Journal of IEEE Transactions on Learning Technologies*, 4(2):187–195, 2011.

[16] E. Charbonneau, A. Miller, and J. J. LaViola. Teach me to dance: Exploring player experience and performance in full body dance games. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology, ACE 2011*, pages 43:1–43:8, 2011.

[17] W. Chu and S. Tsai. Rhythm of motion extraction and rhythm-based cross-media alignment for dance videos. *Journal of IEEE Transactions on Multimedia*, 14(1):129–141, 2012.

[18] W. T. Chu and S. Y. Tsai. Rhythm of motion extraction and rhythm-based cross-media alignment for dance videos. *Journal of IEEE Transactions on Multimedia*, 14(1):129–141, 2012.

[19] N. Collins. Influence in early electronic dance music: An audio content analysis investigation. In *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012*, pages 1–6, 2012.

[20] S. Dixon, E. Pampalk, and G. Widmer. Classification of dance music by periodicity patterns. In *Proceedings of the 4th International Conference on Music Information Retrieval,ISMIR 2003*, pages 159–165, 2003.

[21] A. Dremeau and S. Essid. Probabilistic dance performance alignment by fusion of multimodal features. In *Proceedings of the IEEE International Conference on*

Acoustics, Speech and Signal Processing, ICASSP 2013, pages 3642–3646, 2013.

[22] B. Duggan, B. O'Shea, M. Gainza, and P. Cunningham. Machine annotation of sets of traditional irish dance tunes. In *Proceedings of the 9th International Society for Music Information Retrieval Conference, ISMIR 2016*, pages 401–406, 2008.

[23] S. Essid, X. Lin, M. Gowing, G. Kordelas, A. Aksay, P. Kelly, T. Fillon, Q. Zhang, A. Dielmann, V. Kitanovski, R. Tournemenne, A. Masurelle, E. Izquierdo N. E. O'Connor, P. Daras, and G. Richard. A multimodal dance corpus for research into interaction between humans in virtual environments. *Journal on Multimodal User Interfaces*, 7(1–2):157–170, 2013.

[24] A. Z. M. Faridee, S. R. Ramamurthy, H. M. S. Hossain, and N. Roy. Happyfeet: Recognizing and assessing dance on the floor. In *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications, HotMobile 2018*, pages 49–54, 2018.

[25] M. C. Felice, S. F. Alaoui, and W. E. Mackay. Knotation: Exploring and documenting choreographic processes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018*, pages 448:1–448:12, 2018.

[26] M. Fujimoto, N. Fujita, Y. Takegawa, T. Terada, and M. Tsukamoto. A motion recognition method for a wearable dancing musical instrument. In *Proceedings of the International Symposium on Wearable Computers, ISWC 2009*, pages 11–18, 2009.

[27] M. Fujimoto, N. Fujita, T. Terada, and M. Tsukamoto. Lighting choreographer: An LED control system for dance performances. In *Proceedings of the 13th ACM International Conference on Ubiquitous Computing, UbiComp 2011*, pages 613–614, 2011.

[28] S. Fukayama and M. Goto. Music content driven automated choreography with beat-wise motion connectivity constraints. In *Proceedings of the 12th Sound and Music Computing Conference, SMC 2015*, pages 177–183, 2015.

[29] A. Gkiokas and V. Katsouros. Convolutional neural networks for real-time beat tracking: A dancing robot application. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017*, pages 286–293, 2017.

[30] R. I. Godøy and A. R. Jensenius. Body movement in music information retrieval. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009*, pages 45–50, 2009.

[31] D. P. Golz and A. Shaw. Augmenting live performance dance through mobile technology. In *Proceedings of the 28th International BCS Human Computer Interaction Conference on HCI 2014 - Sand, Sea and Sky - Holiday HCI, BCS-HCI 2014*, pages 311–316, 2014.

[32] M. Goto and Y. Muraoka. A beat tracking system for acoustic signals of music. In *Proceedings of the ACM International Conference on Multimedia, MM 1994*, pages 365–372, 1994.

[33] F. Gouyon and S. Dixon. Dance music classification: A tempo-based approach. In *Proceedings of the 5th International Conference on Music Information Retrieval, ISMIR 2004*, 2004.

[34] M. Gowing, P. Kell, N. E. O'Connor, C. Concolato, S. Essid, J. Lefeuvre, R. Tournemenne, E. Izquierdo, V. Kitanovski, X. Lin, and Q. Zhang. Enhanced visualisation of dance performance from automatically synchronised multimodal recordings. In *Proceedings of the 19th ACM International Conference on Multimedia, MM 2011*, pages 667–670, 2011.

[35] C. F. Griggio and M. Romero. Canvas dance: An interactive dance visualization for large-group interaction. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems, CHI EA 2015*, pages 379–382, 2015.

[36] T. Großhauser, B. Bläsing, C. Spieth, and T. Hermann. Wearable sensor-based real-time sonification of motion and foot pressure in dance teaching and training. *Journal of the Audio Engineering Society*, 60(7/8):580–589, 2012.

[37] C. Guedes. Controlling musical tempo from dance movement in real-time: A possible approach. In *Proceedings of the 29th International Computer Music Conference, ICMC 2003*, 2003.

[38] N. Hieda. Mobile brain-computer interface for dance and somatic practice. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, UIST 2017*, pages 25–26, 2017.

[39] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Journal of Neural Computation*, 9(8):1735–1780, 1997.

[40] A. Holzapfel and E. Benetos. The sousta corpus: Beat-informed automatic transcription of traditional dance tunes. In *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016*, pages 531–537, 2016.

[41] T. Jehan, M. Lew, and C. Vaucelle. Cati dance: Self-edited, self-synchronized music video. In *Proceedings of the ACM SIGGRAPH 2003 Sketches &Amp; Applications, SIGGRAPH 2003*, pages 1–1, 2003.

[42] T. Kell and G. Tzanetakis. Empirical analysis of track selection and ordering in electronic dance music using audio feature extraction. In *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013*, pages 505–510, 2013.

[43] D. Kim, D. H. Kim, and K. C. Kwak. Classification of K-Pop dance movements based on skeleton information obtained by a kinect sensor. *Sensors*, 17(6):1261, 2017.

[44] E. S. Kim. Choreosave: A digital dance preservation system prototype. In *Proceedings of the American Society for Information Science and Technology*, pages 48:1–48:10, 2011.

[45] H. Kim and J. A. Landay. Aeroquake: Drone augmented dance. In *Proceedings of the ACM Designing Interactive Systems Conference, DIS 2018*, pages 691–701, 2018.

[46] P. Knees, Á. Faraldo, P. Herrera, R. Vogl, S. Böck, F. Hörschläger, and M. L. Goff. Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015*, pages 364–370, 2015.

[47] B. Li, A. Maezawa, and Z. Duan. Skeleton plays piano: Online generation of pianist body movements from MIDI performance. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, pages 218–224, 2018.

[48] J. MacRitchie, B. Buck, and N. J. Bailey. Visualising musical structure through performance gesture. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009*, pages 237–242, 2009.

[49] C. Mousas. Performance-driven dance motion control of a virtual partner character. In *Proceedings of the IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2018*, pages 57–64, 2018.

[50] A. Nakamura, S. Tabata, T. Ueda, S. Kiyofuji, and Y. Kuno. Multimodal presentation method for a dance training system. In *Proceedings of the SIGCHI Conference Extended Abstracts on Human Factors in Computing Systems, CHI EA 2005*, pages 1685–1688, 2005.

[51] K. Nymoen, R. I. Godøy, A. R. Jensenius, and J. Tørresen. Analyzing correspondence between sound objects and body motion. *Journal of ACM Transactions on Applied Perception*, 10(2):9:1–9:22, 2013.

[52] F. Ofli, E. Erzin, Y. Yemez, and A. M. Tekalp. Multimodal analysis of dance performances for music-driven choreography synthesis. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing, ICASSP 2010*, pages 2466–2469, 2010.

[53] F. Ofli, E. Erzin, Y. Yemez, and A. M. Tekalp. Learn2Dance: Learning statistical music-to-dance mappings for choreography synthesis. *Journal of IEEE Transactions on Multimedia*, 14(3-2):747–759, 2012.

[54] K. Özcimder, B. Dey, R. J. Lazier, D. Trueman, and N. E. Leonard. Investigating group behavior in dance: an evolutionary dynamics approach. In *Proceedings of the American Control Conference, ACC 2016*, pages 6465–6470, 2016.

[55] M. Panteli, N. Bogaards, and A. K. Honingh. Modeling rhythm similarity for electronic dance music. In *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014*, pages 537–542, 2014.

[56] J. A. Paradiso, K. Y. Hsiao, and E. Hu. Interactive music for instrumented dancing shoes. In *Proceedings of the 25th International Computer Music Conference, ICMC 1999*, pages 453–456, 1999.

[57] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *Proceedings of the 31st Conference on Neural Information Processing Systems, NIPS 2017*, 2017.

[58] K. E. Raheb and Y. Ioannidis. Modeling abstractions for dance digital libraries. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL 2014*, pages 431–432, 2014.

[59] K. E. Raheb, G. Tsampounaris, A. Katifori, and Y. Ioannidis. Choreomorphy: A whole-body interaction experience for dance improvisation and visual experimentation. In *Proceedings of the 2018 International Conference on Advanced Visual Interfaces, AVI 2018*, pages 27:1–27:9, 2018.

[60] M. Raptis, D. Kirovski, and H. Hoppe. Real-time classification of dance gestures from skeleton animation. In *Proceedings of the 10th ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2011*, pages 147–156, 2011.

[61] L. Risk, L. Mok, A. Hankinson, and J. Cumming. Melodic similarity in traditional french-canadian instrumental dance tunes. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015*, pages 93–99, 2015.

[62] A. Rizzo, K. E. Raheb, S. Whatley, R. M. Cisneros, M. Zanoni, A. Camurri, V. Viro, J.-M. Matos, S. Piana, and M. Buccoli. Wholodance: Whole-body interaction learning for dance education. In *Proceedings of the Workshop on Cultural Informatics, co-located with the EUROMED International Conference on Digital Heritage, EUROMED 2018*, pages 41–50, 2018.

[63] M. Saini, S. P. Venkatagiri, W. T. Ooi, and M. C. Chan. The jiku mobile video dataset. In *Proceedings of the 4th ACM Multimedia Systems Conference, MMSys 2013*, pages 108–113, 2013.

[64] M. Sakashita, K. Suzuki, K. Kawahara, K. Takazawa, and Y. Ochiai. Materialization of motions: Tangible representation of dance movements for learning and archiving. In *Proceedings of the ACM SIGGRAPH 2017 Studio, SIGGRAPH 2017*, pages 7:1–7:2, 2017.

[65] A. Schindler and A. Rauber. An audio-visual approach to music genre classification through affective color

features. In *Proceedings of the European Conference on Information Retrieval, ECIR 2015*, pages 61–67, 2015.

[66] H. Schreiber and M. Müller. A crowdsourced experiment for tempo estimation of electronic dance music. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, pages 409–415, 2018.

[67] R. A. Seger, M. M. Wanderley, and A. L. Koerich. Automatic detection of musicians' ancillary gestures based on video analysis. *Journal of Expert Systems with Applications*, 41(4):2098–2106, 2014.

[68] A. Soga, Y. Yazaki, B. Umino, and M. Hirayama. Body-part motion synthesis system for contemporary dance creation. In *Proceedings of the ACM SIGGRAPH 2016 Posters, SIGGRAPH 2016*, pages 29:1–29:2, 2016.

[69] A. Soga and I. Yoshida. Interactive control of dance groups using kinect. In *Proceedings of the 10th International Conference on Computer Graphics Theory and Applications, GRAPP 2015*, pages 362–365, 2015.

[70] E. Stavrakis, A. Aristidou, M. Savva, S. L. Himona, and Y. Chrysanthou. Digitization of cypriot folk dances. In *Proceedings of the 4th International Conference on Progress in Cultural Heritage Preservation, EuroMed 2012*, pages 404–413, 2012.

[71] J. K. T. Tang, J. C. P. Chan, and H. Leung. Interactive dancing game with real-time recognition of continuous dance moves from 3D human motion capture. In *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication, ICUIMC 2011*, pages 50:1–50:9, 2011.

[72] T. Tang, J. Jia, and M. Hanyang. Dance with melody: An LSTM-autoencoder approach to music-oriented dance synthesis. In *Proceedings of the 26th ACM International Conference on Multimedia, MM 2018*, pages 1598–1606, 2018.

[73] T. Tang, J. Jia, and H. Mao. Dance with melody: An LSTM-autoencoder approach to music-oriented dance synthesis. In *Proceedings of the 26th ACM International Conference on Multimedia, MM 2018*, pages 1598–1606, 2018.

[74] S. Tsuchida, S. Fukayama, and M. Goto. Automatic system for editing dance videos recorded using multiple cameras. In *Proceedings of the 14th International Conference on Advances in Computer Entertainment Technology, ACE 2017*, pages 671–688, 2017.

[75] S. Tsuchida, S. Fukayama, and M. Goto. Query-by-dancing: A dance music retrieval system based on body-motion similarity. In *Proceedings of the 24th International Conference on MultiMedia Modeling, MMM 2019*, pages 251–263, 2019.

[76] S. Tsuchida, T. Takemori, T. Terada, and M. Tsukamoto. Mimebot: spherical robot visually imitating a rolling sphere. *International Journal of Pervasive Computing and Communications*, 13(1):92–111, 2017.

[77] S. Tsuchida, T. Terada, and M. Tsukamoto. A system for practicing formations in dance performance supported by self-propelled screen. In *Proceedings of the 4th Augmented Human International Conference, AH 2013*, pages 178–185, 2013.

[78] S. Tsuchida, T. Terada, and M. Tsukamoto. A dance performance environment in which performers dance with multiple robotic balls. In *Proceedings of the 7th Augmented Human International Conference, AH 2016*, pages 12:1–12:8, 2016.

[79] P. Wang and J. M. Rehg. A modular approach to the analysis and evaluation of particle filters for figure tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2006*, pages 790–797, 2006.

[80] K. Yadati, M. Larson, C. C. S. Liem, and A. Hanjalic. Detecting drops in electronic dance music: Content based approaches to a socially significant music event. In *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014*, pages 143–148, 2014.

[81] T. Yamaguchi and H. Kadone. Bodily expression support for creative dance education by grasping-type musical interface with embedded motion and grasp sensors. *Sensors*, 17(5):1171, 2017.

[82] W. Zhang, Z. Liu, L. Zhou, H. Leung, and A. B. Chan. Martial arts, dancing and sports dataset. *Journal of Image Vision Computing*, 61(C):22–39, 2017.

[83] X. Zhang, T. Dekel, T. Xue, A. Owens, Q. He, J. Wu, S. Mueller, and W. T. Freeman. MoSculp: Interactive visualization of shape and time. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, UIST 2018*, pages 275–285, 2018.

MICROTIMING ANALYSIS IN TRADITIONAL SHETLAND FIDDLE MUSIC

Estefanía Cano
Fraunhofer IDMT
cano@idmt.fraunhofer.de

Scott Beveridge
Songquito
scott.beveridge@songquito.com

ABSTRACT

This work aims to characterize microtiming variations in traditional Shetland fiddle music. These microtiming variations dictate the rhythmic flow of a performed melody, and contribute, among other things, to the suitability of this music as an accompaniment to dancing. In the context of Shetland fiddle music, these microtiming variations are often referred to as *lilt*. Using a corpus of 27 traditional fiddle tunes from the Shetland Isles, we examine inter-beat timing deviations, as well as inter-onset timing deviations of eighth note sequences. Results show a number of distinct inter-beat and inter-onset rhythmic patterns that may characterize lilt, as well as idiosyncratic patterns for each performer. This paper presents a first step towards the use of Music Information Retrieval (MIR) techniques for modelling lilt in traditional Scottish fiddle music, and highlights its implications in the field of ethnomusicology.

1. INTRODUCTION

The Shetland Isles is an archipelago situated 100 miles north of the Scottish mainland (Figure 1). The distinctive music-culture of the Isles reflects a wide range of influences from around the North-Atlantic, including mainland Scottish, Scandinavian, and North-American, in particular. The earliest evidence of a performance tradition on violin, or fiddle, in the Shetland Isles dates from the eighteenth century, but there is also evidence of a pre-violin string-instrument tradition that dates to a much earlier time. However, it was over the course of the nineteenth century and into the twentieth century that the modern instrument established itself as an important component of ceremonies and rituals where dancing often played an important role [3]. The fact that solo fiddle music was used for dancing had a great influence on performance practice, with commentators often referring to the 'lilt' of a musician's performance when describing its suitability as an accompaniment to dancing. Even though a strict definition of lilt does not exist, the term often refers to the rhythmic flow imparted to the music by the performers. The term lilt bears a resemblance to the concept of swing in Jazz performance, where

Figure 1: The Shetland Isles

notated consecutive eighth notes are not played with equal duration, often approximating a tied triplet notation with a 2:1 ratio [4]. Given that an exact definition of lilt does not exist, and that many rhythmic elements could potentially play a role in imparting this dance feel to the music, this paper focuses on understanding general trends in microtiming variations in Shetland fiddle music. With this work we hope to bring a better understanding of what "*playing with lilt*" might entail.

2. RELATED WORK

The first systematic analysis of lilt in Shetland fiddle music was carried out by Peter Cooke in 1986 [3]. In this work, an electrokymograph, an early device used to measure variations in pressure, was used to capture the pitch contour and envelope of sound from a live fiddle performance. This was then used to manually calculate note onsets and approximate note durations. Anecdotal observations made by Cooke show eighth note durations that are rarely equal, with ratios of consecutive eighth notes in the range of 4:3 to 2:1. This *long-short* duration pattern is reminiscent of

Performer	Region	Title	Tr. ID
Willie Hunter	Mainland	Aandowin at da Bow	81352
		Caald Nights o Winter	81349
		Come Again Ye'r Welcome	89928
		Da Aald Hill Grind	81386
		Da Forfeit o da Ship	81351
		Fram Upon Him	81353
		Garster's Dream	81354
Bobby Peterson	Mainland	The Flowers of Edinburgh	80293
		Clean Pease Strae	36653
		Glen Grant	36656
		Kebister Head	80295
		MacDonald's Reel	80294
		The Fishers Hornpipe	36646
		The Millers Hornpipe	36647
		Willafjord	36644
John Jamieson Irvine	Whalsay	Auld Reel o Whalsay	91259
Robert Bairnson	Mainland	Da Burn o'Finnigirth	50890
		Da Hill o'Finnigirth	50890
		Mangaster Voe	50888
		Oliver Jack	50889
		The Headlands	50888
		Willafjord	50889
Andrew Poleson	Whalsay	Greigs Pipes	96869
		Lady Mary Ramsay	96896
		Supple Sandy	100412
		Unknown Reel	96861
		Up da Stroods da Sailor Goes	100411

Table 1: The Shetland fiddle corpus. All the recordings were made between 1955 - 1977 and are available in the Tobar an Dulchais collection.

swing in jazz performances [4, 6]. The seminal work carried out by Cooke in the 80s has seen a recent resurgence of interest, and has been the focus of recent analysis using state-of-the art computational analysis methods [1].

Swing research in fiddle music has also been carried out in the context of Irish traditional music [9]. In this work, a simple pitch detection and autocorrelation approach was used to estimate note durations and calculate 'swing percentages'. This method was proposed to detect the presence of swing in both synthesized and audio recordings.

Given its central role in the Jazz repertoire, swing has received considerable attention in this domain [2, 4, 6]. In [6], Friberg and Sundström examine the relationship between swing ratio and tempo in a number of popular jazz recordings. Using spectrograms obtained from the cymbal, the swing ratio was calculated based on the timing of successive eighth notes. The reported swing ratio ranged between 3.5:1 at low tempi to 1:1 at high tempi. In a later reexamination of this work, [4] applied automatic techniques to estimate swing ratios from a larger corpus of jazz recordings. The results support the initial findings of Friberg and Sundström with the added benefit that they can be applied at scale. Microtiming variations play an important role in a number of other music traditions including Ragtime syncopation [10], Samba rhythms [7], and Jembe music from Mali [8].

3. METHOD

This work represents an initial step towards the computational analysis of microtiming variations in Shetland fiddle music. As a starting point, it was necessary to compile a relevant fiddle data set. Once the recordings had been selected, all the tunes were transcribed to traditional music notation by an expert musicologist (Sect. 3.1). To be able to extract accurate timing information, each transcription was aligned in time with its corresponding audio track. For the alignment, the transcriptions were exported in MIDI format, and a MIDI-to-audio synchronization algorithm was used (Sect. 3.2). Finally, the time-aligned transcriptions were analyzed to extract timing information in the symbolic domain (Sect. 3.3). In the following sections, we provide a detailed description of all the steps in our analysis.

3.1 Fiddle Corpus

The fiddle corpus compiled in this work comprises 27 recordings of solo fiddle tunes available in the Tobar an Dualchais website.[1] Tobar an Dualchais is a collaborative project aimed at the digitization and preservation of Gaelic and Scottish recordings. The recordings in the corpus are field recordings made on a Nagra III reel-to-reel tape recorder with Sennheiser microphones. These were captured as monophonic wave files at 22.1kHz, 16 bit resolution. Table 1 provides a description of the corpus. Five performers are included, three from Mainland Shetland and two from the island of Whalsay (see Fig.1 for a map of the Shetland Isles). Track IDs from the Tobar and Dualchais collection are also provided to allow future research on the same corpus. For more information on the corpus see our accompanying website.[2]

The entire corpus was transcribed by a musicologist from the University of Aberdeen in Scotland (with expertise on Shetland fiddle traditions), providing us with transcriptions in conventional music notation for each of the 27 tunes.

3.2 MIDI-to-audio Synchronization

To calculate microtiming variations, the 27 transcriptions were exported in MIDI format and synchronized to their corresponding audio tracks. Synchronized MIDI transcriptions allow us to find the exact location in time of note onsets. For the MIDI-to-audio alignment, the method proposed in [5] was used. This method represents both the audio and MIDI streams as a combination of chroma and chroma onset features. The optimal alignment between audio and MIDI is obtained through dynamic time warping (DTW) using dynamic programming. Two difficulties were observed with the automatic synchronization. First, synchronization of the initial bars was inaccurate for some tunes, resulting in noticeably asynchronous audio and MIDI notes. Second, given that the tuning of certain files (or segments of the files) often lays between two consecutive semitones, the algorithm struggled to deal with these pitch variations and resulted in inaccurate alignments. To account for these inaccuracies in the automatic synchronization, the annotations of the complete corpus

[1] http://www.tobarandualchais.co.uk/

[2] https://github.com/ecanoc/ShetlandFiddles

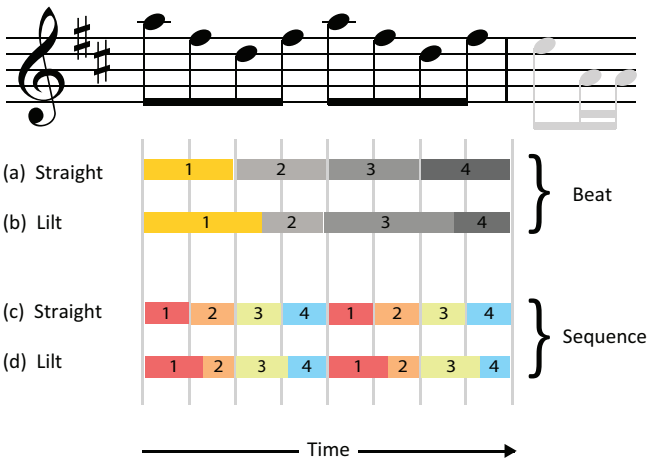


Figure 2: Schematic depiction of our onset-based analysis on a segment of the tune *Clean Pease Strae*. **Beat-level analysis:** (a) Beats of equal length (straight version) (b) Beats with timing deviations (lilt version). **Sequence analysis:** (c) Eighth notes of equal length (straight version) (d) Eighth notes with timing deviations (lilt version).

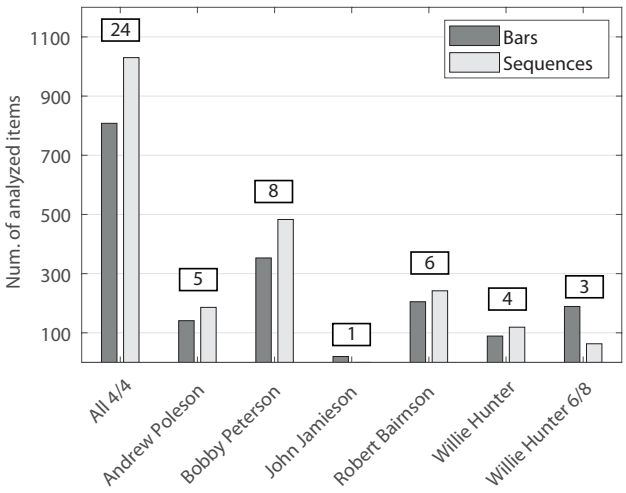


Figure 3: Total number of bars, note sequences and tunes (shown in boxed numbers above) per performer in the Shetland fiddle corpus.

were then manually fine-tuned by two professional musicians using Sonic Visualiser [3] .

3.3 Onset-based Analysis

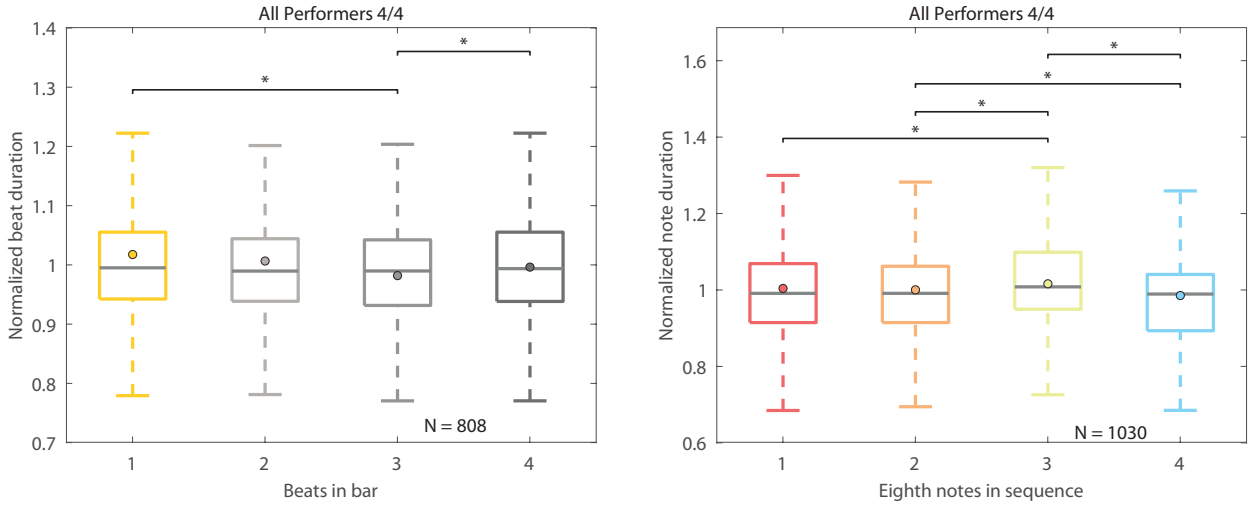
The synchronized MIDI transcriptions are used to analyze microtiming variation in our corpus. Two different analyses were conducted: (a1) Microtiming variations on the beat level, and (a2) Microtiming variations in eighth note sequences. Given that accurate annotations of the note offsets are not available or easily extracted, we focus on measuring time intervals between onsets in this study. For the beat-level analysis (a1), we calculate time intervals between consecutive beats in the bar. We refer to these intervals as inter-beat-intervals (IBI). For the eighth-note-sequence analysis (a2), we calculate time intervals between consecutive eighth notes in the sequence. We refer to these intervals as inter-onset-interval (IOI).

Our analysis can be better understood by looking at Fig. 2, where a segment of the tune *Clean Pease Strae* is displayed. For the beat level analysis, Fig. 2a shows inter-beat-intervals (IBI) of equal length. This is referred to as the *straight* version of a performance. Our study seeks to understand whether deviations from the straight version exist, and whether performers slightly shorten or stretch the beats when playing with lilt. This scenario is depicted in Fig. 2b, where beats 1 and 3 are slightly longer than beats 2 and 4. Similarly, we analyze sequences of eighth notes in all tunes, and focus on measuring inter-onset-intervals (IOI) for sequences of four eighth notes (for $\frac{4}{4}$ meter), and sequences of three eighth notes (for $\frac{6}{8}$ meter). This can be seen in Figs. 2c and 2d, where the straight and lilt versions of the two eighth note sequences in the bar are shown, respectively. As with the beat-level analysis, we seek to understand whether performers slightly shorten or stretch the eighth notes when playing with lilt. In Fig 2d, the eighth notes 1 and 3 in the sequences are slightly longer than notes 2 and 4.

3.3.1 Practical Considerations

To facilitate the analysis of our corpus, we implemented an algorithm for pattern detection in MIDI files. The algorithm includes methods to detect relevant information such as downbeats, rhythm sequences (eighth notes, quarters, etc), double-stops, pitch sequences, etc. Supported by our pattern detection algorithm, the following considerations were taken: (1) Given that our analysis uses note onsets to detect inter-beat-intervals (IBI), only those bars where an onset was present (a note was played) on each of the beats in the bar were considered. While this means that parts of the performance were discarded in our analysis, it also completely removes the need to perform beat annotations, and relies entirely on the performers' onsets to calculate timing. (2) For the eighth-note-sequence analysis, only complete sequences of four eighth notes in $\frac{4}{4}$ bars, and three eighth notes in $\frac{6}{8}$ bars were considered. We restrict our analysis to sequences whose first eighth note coincides with an on-beat. Here again, we completely rely on the performer's onsets to calculate inter-onset-intervals (IOI) and timing variations. (3) A song-wise mean normalization of IBI and IOI was performed for the beat-level analysis and for the sequence analysis, respectively. This normalization allows us to remove differences in beat and note duration dictated by the different tempi of the tunes, and allows us to make direct comparisons between tunes. (4) In our corpus, only three tunes out of the 27 are written in $\frac{6}{8}$. These three tunes are all performed by Willie Hunter, and hence, conclusions with respect to the usage of timing in $\frac{6}{8}$ bars are restricted to the ones revealed by Willie Hunter's performances. (5) In all tunes where double-stops

[3] https://www.sonicvisualiser.org/



(a) Corpus beat-level results

(b) Corpus note-sequence results

Figure 4: Results for all performers and all tunes in $\frac{4}{4}$ bars in the corpus. (a) Beat-level analysis, and (b) Note-sequence analysis. Significant differences are highlighted (* p-value < 0.05: Wilcoxon signed-rank test for pairwise comparisons)

were detected, only the onset of the lower note was used in our calculations.

Figure 3 shows a summary of the number of tunes, complete bars found and complete sequences found per performers in the corpus using our pattern detection algorithm.

4. RESULTS

4.1 Annotation reliability

To assess the reliability of the annotations, five of the tracks in the corpus were annotated by the two expert annotators. As a measure of annotator agreement, we use Chronbach's alpha coefficient α. For simplicity, we refer to the annotators as a1 and a2 in the following. The agreement between annotators was extremely high with $\alpha(a1, a2) = 0.9995$. It is important to consider, that the starting point for both annotators was the MIDI synchronization obtained with the method proposed in [5]. Hence, we also calculate the Chronbach's coefficient between the automatic MIDI synchronization M, $a1$, and $a2$, resulting in $\alpha(a1, M) = 0.9999$, and $\alpha(a2, M) = 0.9995$. These results validate both the reliability of our final annotations, and the high accuracy obtained by the automatic synchronization in the first place. Only minor modifications were required in order to obtain the final annotations from the automatically synchronized ones. These results also validate the usage of the method in [5] as an efficient approach for MIDI-to-audio synchronization of fiddle recordings, and opens the possibility to further enlarge our corpus in the future.

4.2 Onset-based analysis

In the following sections, we present our results in a top-down manner, starting with trends observed over the entire corpus, followed by a performer-wise analysis, and concluding with a comparative analysis of the tune *Willafjord* (the only tune performed by two perfomers in our corpus). To summarize the results, boxplots are presented both for the beat-level and sequence analyses. In all the figures, the boxplots represent the minimum and maximum values (whiskers), and the first, second (median) and third quartiles (box). In addition, the mean values for each distribution are displayed with colored circles. We present normalized note and beat durations in our plots (see Sect. 3.3.1).Values smaller than 1 show IOIs and IBIs shorter than that of the mean. Similarly, values larger than 1 show IOIs and IBIs longer than that of the mean.

4.2.1 Corpus analysis

Pairwise differences of the mean IOIs and mean IOBs of the entire corpus are calculated using the Wilcoxon signed-rank test at a 5% significance level (p-value < 0.05). Results for the corpus analysis are presented in Fig. 4 where the contribution of all performers and all tunes in $\frac{4}{4}$ meter are displayed.

Figure 4a shows the results of the beat-level analysis for all performers and tunes in $\frac{4}{4}$ bars. The pairwise comparisons revealed the normalized beat duration of beat 3 to be significantly shorter than 1 and 4. The slight shortening of beat 3 across all performers might indicate an attempt to delineate the on-beats in the $\frac{4}{4}$ bar.

Figure 4b shows the results of the sequence analysis. Note 3 in the sequence is significantly longer than all other eighth notes in a sequence. In addition, we see a significant difference between note 2 and note 4, with note 4 slightly shorter than note 2. In contrast, no significant difference was found between notes 1 and 2 in the sequence, appearing to be performed mostly in a *straight* manner .

4.2.2 Performer-wise analysis

In this section, we analyze possible performer idiosyncrasies both on the beat level and on the note sequence

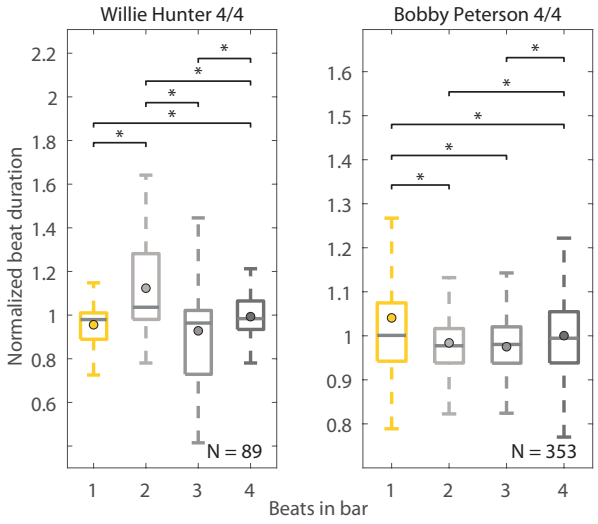


Figure 5: Performer-wise beat-level results. Significant differences are highlighted (* p-value < 0.05: Wilcoxon signed-rank test for pairwise comparisons)

level.

On the beat level, significant differences are evident in only two of the five performers, Willie Hunter and Bobby Peterson, with the other performers mostly showing beats of equal length. As seen in Fig. 5, the general trend in both Hunter and Peterson is to slightly shorten the third beat. These results support the trend observed in the corpus analysis of beats in Fig. 4a. In Willie Hunter, beat 3 is significantly shorter than beats 2 and 4, but beats 1 and 3 appear to be mostly played with the same duration. In contrast, beat 3 in Bobby Peterson is significant shorter than beats 1 and 4, but not than beat 2. Beat 1 in Bobby Peterson appears to be slightly prolonged, showing significant difference with all other beats.

On the note-sequence level, significant differences are evident in five performers as displayed in Fig. 6. John Jamieson only performs one tune in our corpus, and no eighth note sequences were found in it. In this analysis, we also include the results for the three tunes in $\frac{6}{8}$ performed by Willie Hunter. For the $\frac{6}{8}$ tunes, we consider sequences of three consecutive eighth notes. In Willie Hunter's $\frac{6}{8}$ sequences, we see a tendency to slightly shorten note 2. These differences are significant with respect to note 1 and 3. In Hunter's $\frac{4}{4}$ sequences, note 3 is significantly longer than all the others, a pattern that we also observe in Bobby Peterson. However, for Bobby Peterson, we additionally see a slightly shortened note 4 (significant with respect to note 2 and 3), and a tendency to lengthen note 2 with respect to 1. In contrast, Robert Bairnson appears to play mostly straight sequences, with a slightly shorter note 3 with respect to 1. As mentioned by Cooke in [3], the shortening of the 3rd note may be related to bowing: if notes 2, 3, and 4 are played with one bow, the middle note if often shortened and accented. The influence of bowing on lilt falls beyond the scope of this study, but calls for further future investigation. Andrew Poleson on the other hand, shows a consistent shortening of note 2, with significant differences with respect to all other notes.

It appears that a tendency to play pairs of eighth notes (either 1-2 or 3-4) in a long-short (LS) pattern might exist, with Willie Hunter ($\frac{4}{4}$) and Bobby Peterson showing a LS pattern in notes 3-4, and Andrew Poleson in notes 1-2. These results also go in hand with observations made by Cooke in [3].

4.2.3 Comparative analysis of the tune Willafjord

To analyze differences between performers, we compare the two versions of the tune *Willafjord* in our corpus; one performed by Bobby Peterson, the other by Robert Bairnson. Here we focus specifically on the differences at the note sequence level, and perform pairwise comparisons of the four notes sequence. Given that there are differences between the two performances, a direct correspondence between the sequences that we compare cannot be established, and hence, we treat them as unmatched pairs. To do so, the Wilcoxon sum-rank test is performed at a 5% significance level (p-value < 0.05). The tests reveal significant differences in IOIs for all of the eighth note positions, suggesting completely different performing practices by the two fiddlers for the tune *Willafjord*.

Our comparative analysis of the tune *Willafjord* gives a tantalizing insight into the source of such performer idiosyncrasies. These can be further elucidated in the biographies of Bobby Peterson and Robert Bairnson in Peter Cooke's original ethnographic account. Although both men lived on the main island of Shetland (approximately 23 miles apart), Bobby Peterson spent a great deal of his young life as sailor on a whaling ship. There, he was subject to a great many influences including hearing and learning tunes from other sailors from Scotland and Scandinavia. He learned the fiddle mostly be ear, in fact the first tune he ever learned was the focus of our analysis, the tune *Willafjord*. In contrast, Robert Bairnson, although only a short distance away, was fairly isolated as infrastructure on the Shetland Isles in the early 20th century made travelling very difficult. Bairnson also lived in a district where religious attitudes deemed music and dancing sinful. He was taught formally by a local minister, learning notation at the same time. Therefore he lacked the 'indigenous repertory' of other Shetlanders [3, p. 20]. These biographies go some way to explaining why such clear difference can be observed between performers, and in particular, the straight playing of Bairnson versus the more lilting performance of Peterson.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented microtiming analysis of a corpus of traditional Shetland fiddle music. The corpus, created specifically for this study, was first annotated by expert annotators using automatic MIDI-to-audio synchronization as a starting point. We examined inter-beat-interval timing deviations, as well as inter-onset-interval timing deviations of eighth note sequences to find the elusive qualities of *lilt*. We first analyzed the corpus as a whole, moving to

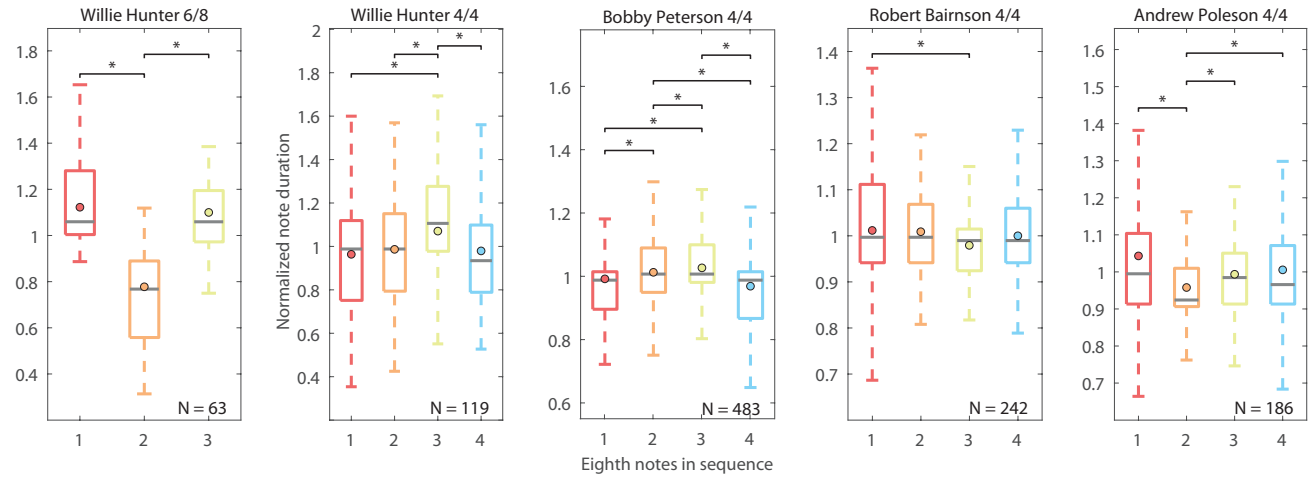


Figure 6: Performer note-sequence results. Significant differences are highlighted (* p-value < 0.05: Wilcoxon signed-rank test for pairwise comparisons)

performer-based analysis, and finalized with a comparative study of a tune played by two performers.

In general, the beat-level analysis indicates a tendency to slight shorten beat 3 in the bar. The note-sequence analysis, suggests the use of long-short patterns in pairs of eighth notes (either 1-2 or 3-4). Even though these emerging patterns in the data may suggest what playing with lilt entails, it is clear that lilt also encompasses idiosyncratic aspect of performance practice. Further investigation is required in order to understand how these patterns relate to other aspects of music performance such as bowing, accentuation, repertoire-specific performing styles, tempo, etc.

With this work, we have reached three important milestones in our research: (1) We have produced high-quality annotations for our corpus that will serve as the foundation and ground-truth for more powerful computational models, (2) we have validated the use of a state-of-the-art MIDI-to-audio synchronization algorithm for the task at hand, and (3) we have revealed microtiming variation patterns in fiddle music from the Shetland Isles.

6. ACKNOWLEDGMENTS

The authors would like to thank the panel of annotators Juan Fernando Alzate and Elissa Sayampanathan, as well as Mr. Ronnie Gibson and Dr. Frances Wilkins at the University of Aberdeen, and Dr. Cathlin Macaulay at the Scottish Studies Archive, School of Literature, Languages and Cultures Celtic & Scottish Studies, University of Edinburgh.

7. REFERENCES

[1] Scott Beveridge, Ronnie Gibson, and Estefanía Cano. Performer profiling as a method of examining the transmission of Scottish traditional music. In *Proc of the 4th International Workshop on Folk Music Analysis (FMA)*, Istanbul, Turkey, 2014.

[2] Matthew W. Butterfield. Why do jazz musicians swing their eighth notes? *Music Theory Spectrum*, 33(1):3–26, 2011.

[3] Peter Cooke. *The Fiddle Tradition of the Shetland Isles*. Cambridge University Press, 1986.

[4] Christian Dittmar, Martin Pfleiderer, and Meinard Müller. Automated estimation of ride cymbal swing ratios in jazz recordings. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 271–277, Malaga, Spain, 2015.

[5] Sebastian Ewert, Meinard Müller, and Peter Grosche. High resolution audio synchronization using chroma onset features. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1869–1872, Taipei, Taiwan, 2009.

[6] Anders Friberg and Andreas Sundströöm. Swing ratios and ensemble timing in jazz performance: Evidence for a common rhythmic pattern. *Music Perception: An Interdisciplinary Journal*, 19(3):333–349, 2002.

[7] Fabien Gouyon. Microtiming in "samba de roda"—preliminary experiments with polyphonic audio. In *Proc. of the XII Simpósio da Sociedade Brasileira de Computação Musical São Paulo*, pages 197–203, Sao Paulo, Brazil, 2007.

[8] Rainer Polak. Rhythmic feel as meter: Non-isochronous beat subdivision in Jembe music from Mali. *Music Theory Online*, 16(4), 2010.

[9] Vincent Rosinach and Caroline Traube. Measuring swing in irish traditional fiddle music. In *Proc. of the International Conference on Music Perception and Cognition*, pages 1168–1171, Bologna, Italy, 2006.

[10] Anja Volk and Bas de Haas. A corpus-based study on ragtime syncopation. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 163–168, Curitiba, Brazil, 2013.

SUPRA: DIGITIZING THE STANFORD UNIVERSITY PIANO ROLL ARCHIVE

Zhengshan Shi[1] **Craig Stuart Sapp**[2] **Kumaran Arul**[3] **Jerry McBride**[4] **Julius O. Smith III**[1]
[1] Center for Computer Research in Music and Acoustics, Stanford University
[2] Center for Computer Assisted Research in the Humanities, Stanford University
[3] Department of Music, Stanford University
[4] Music Library and Archive of Recorded Sound, Stanford University
kittyshi@ccrma.stanford.edu, craig@ccrma.stanford.edu

ABSTRACT

This paper describes the digitization process of a large collection of historical piano roll recordings held in the Stanford University Piano Roll Archive (SUPRA), which has resulted in an initial dataset of 478 performances of pianists from the early twentieth century transcribed to MIDI format. The process includes scanning paper rolls, digitizing the hole punches, and translating the pneumatic expression codings into MIDI format to create expressive performance files. We offer derivative files from each step of this process, including a high resolution image of the roll, a "raw" MIDI file of hole data, an "expressive" MIDI file that translates hole data into dynamics, and an audio file rendering of the expressive MIDI file on a digital piano sample. This provides digital access to the rolls for researchers in a flexible, searchable online database. We currently offer an initial dataset, "SUPRA-RW" from a selection of "red Welte"-type rolls in the SUPRA. This dataset provides roll scans and MIDI transcriptions of important historical piano performances, many being made available widely for the first time.

1. INTRODUCTION

1.1 Background

Piano rolls are among the most important historical music storage formats, utilizing holes on a scrolling paper roll to activate keys automatically by a pneumatic mechanism built into a piano. There are many types of piano rolls with varying degrees of autonomous playback prescribed by the roll system. We focus here on the "reproducing roll", one that reproduces automatically notes, timings, and expressive details of a live performance.

By late 1904 the German company Michael Welte and Söhne developed the first "reproducing player piano", the Welte-Mignon [7]. This was a high-end, fully autonomous self-playing instrument capable of reproducing all features of an original performance of a pianist, including details of timing, dynamics and pedaling. The process involved capturing and coding expressive details into the hole punches on the edges of rolls which then cued the pneumatic mechanisms in the piano to alter hammer velocities instantaneously. These codings were created by skilled technicians who made judgements about the expressive effects they could generate using the pneumatic parameters available for manipulation in each system. The resultant playback of a reproducing roll sounds like a recreation of an individual musical performance.

The reproducing player piano quickly became a commercial success that attracted the most important artists of the day to make recordings on the medium. Composers recorded playing their own works on roll include Claude Debussy, Maurice Ravel, Gustav Mahler, Sergie Prokofiev, Sergei Rachmaninoff, Edvard Grieg, Alexander Scriabin, Enrique Granados, Igor Stravinsky, George Gershwin, and Scott Joplin, among others.

A variety of proprietary reproducing roll systems were produced by over a dozen companies , each one incompatible with the others. The most important of these include Welte-Mignon, Hupfeld, Ampico, Duo-Art, and Phillips Duca. Some companies developed multiple formats. Welte maufactured three formats: Welte T-100 (red Welte), Welte T-98 (green Welte), and Welte-Licensee.

1.2 SUPRA

Stanford University's Player Piano Project [13] is a multi-departmental effort begun in 2014 by the Archive of Recorded Sound to address the obstacles faced by researchers who wish to study piano rolls and pneumatic instruments. Stanford now holds one of the largest roll collections in the world, with more than 16,000 rolls and a dozen pneumatic roll playing instruments. The collection includes a variety of roll types, including thousands of reproducing rolls. A dedicated roll scanner was built by the project to generate image scans of rolls for preservation and digitization. The Stanford University Piano Roll Archive (SUPRA) is the online database of roll images generated by the scanning effort. It aims to provide a virtual experience of piano rolls including a high resolution

color image of the roll and a digitally synthesized audio rendition of each roll. In order to create accurate audio files of reproducing rolls, algorithms that model the pneumatic expression systems of each format must be created. This process of "emulation" (i.e. emulating the pneumatic system through a digital means) is important if the expressive content of reproducing rolls are to be accurately transcribed. This paper outlines the end-to-end process undertaken to scan, digitize, and create emulation algorithms for a subset of 478 Welte T-100 reproducing rolls that will be made accessible through the online SUPRA database.

2. PRIOR WORK

Efforts at scanning and digitizing piano rolls have been undertaken by hobbyists and enthusiasts with private collections, but with inconsistent results and incomplete documentation [1, 15, 18]. More recently a number of institutional projects have been initiated [2, 3, 8–10]. Much of this work has focused on non-reproducing rolls, those rolls lacking expression coding [8, 12]. Most institutional efforts do not provide audio file transfers of the rolls at this point [2, 3, 9]. In some cases only documentation of the rolls is offered but not complete roll scans [3].

A number of dedicated roll scanners have been constructed [1,9,12,15,18]. However, some projects utilize flat bed scanners which require images to be stitched together leading to potential errors [8]. Peter Phillips's unique design of a "pneumatic roll reader" offers some advantages, but does not allow for an archival image [14]. Anthony Robinson has done impressive work with scanner design and is able to produce quality roll scans [15]. These are, however, in gray scale and at modest resolution.

There is not much published work on the topic of emulation although there have been some well publicized efforts at utilizing emulations to playback reproducing rolls. Colmnares et al. [6] provide some theory behind the topic and relies on the work of Wayne Stahnke, one of the pioneers in modeling pneumatic roll systems. Stahnke produced two commercially successful emulated transfers of Rachmaninoff's playing on Ampico reproducing rolls that garnered critical acclaim [17], however the algorithms and detailed procedures of his work remain unpublished. Peter Phillips's recent doctoral thesis provides comprehensive documentation and experimental justification for his emulation procedures and is the most thorough discussion published thus far [14]. Our work here follows closely on his effort and also on our prior work to create an emulation for the Welte Licensee format [16].

3. THE SUPRA-RW DATASET

We chose to begin the SUPRA database with Welte T-100 rolls as these were the first reproducing piano rolls made. These important recordings have received very little attention from roll scanning efforts because they require a scanner capable of managing the significantly greater width of these rolls. Expression emulation of Welte T-100 rolls is also among the most complex of reproducing roll formats.

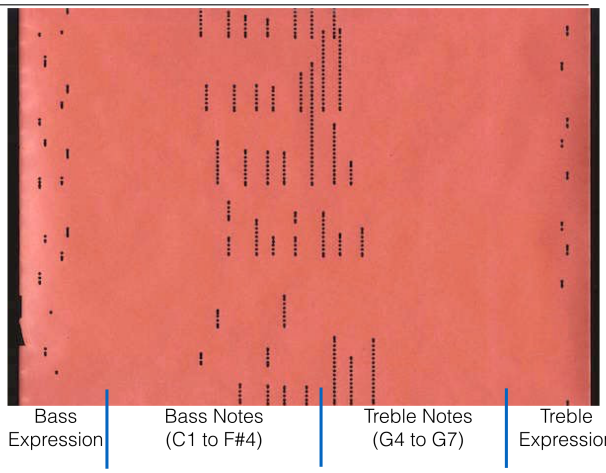


Figure 1: A sample from a red Welte roll. It is divided into four sections: bass expression, bass notes, treble notes, and treble expression.

Welte T-100 rolls are often referred to as "red Welte rolls" because they were generally punched on red paper. They are 12.9 inches (32.8 cm) wide, and have 100 perforation tracks (holes), eight per inch across, evenly spaced. A sample from a red Welte roll is shown in Figure 1. The first and last ten perforation tracks are used for expression, and the middle 80 tracks are used for notes from C1 to G7.

The SUPRA-RW dataset is the result of the digitization of 478 Welte T-100 rolls in Stanford's collection. It consists of about 52 hours of piano roll performances, with an average length of 6 minutes and 32 seconds. The database allows users to search, view images, download and listen to audio emulations of the rolls.

Our digitization procedure results in the following files and derivatives in sequence:

(a) Archival TIFF image at 300 DPI (dots per inch) and 24-bit color.
(b) JPEG files derived from the archival TIFF.
(c) Uncompressed grayscale TIFF file (that utilizes the green color channel of the original scan) allowing efficient access to high resolution detail of the holes.
(d) Raw MIDI file that captures all hole data extracted from the grayscale TIFF file.
(e) Expressive MIDI file that merges multiple holes into single musical notes, applies emulation algorithms to control for individual note dynamics, and adds pedaling. Metadata such as title and composer is also added to this file.
(f) Audio files (WAV and M4A) rendered by running the expressive MIDI file through the Ivory Keys II software synthesizer.

The SUPRA-RW dataset as well as the software are available online [1] at a Creative Commons Attribution-Non-Commercial-ShareAlike 4.0 International license (CC BY-NC-SA 4.0) [2].

[1] https://supra.stanford.edu
[2] https://creativecommons.org/licenses/by-nc-sa/4.0/

Figure 2: Stanford's piano roll scanner built by Swope Design Solutions of San Francisco, California. A Welte T-100 piano roll is shown on the scanner.

4. ROLL DIGITIZATION

4.1 Scanning

Piano rolls are digitized using a dedicated roll scanner built in conjunction with Swope Design Solutions and based on designs by Anthony Robinson [3]. The scanner (Figure 2) uses a line-scan camera (DALSA Spyder3 Color 4k) triggered by a rotary encoder on a glass cylinder that tracks the movement of the paper roll that passes over it. The target resolution of the images is 300 DPI, with the experimentally measured DPI being 301.50 $\pm$ 0.25 across the width of a roll and 300.25 $\pm$ 0.25 along the length of the roll.

The scanner can image a roll up to 5$\times$ playback speed, although the typical acquisition speed is 2–3$\times$. An original uncompressed color image is typically 1–4 GB in size. The line-scan camera takes pictures that are 2 pixels high and 4096 pixels wide. One row of pixels is used to measure the green color channel, and the second row contains interleaved red and blue color pixels. The adjacent images from the camera are 50% overlapped such that a red/blue row from one image is aligned with the green row of the next image. Since the green channel has twice the resolution of the red or blue channels, the green channel is used for hole data extraction from the images.

[3] For more details and a video of the scanner in action, see https://library.stanford.edu/blogs/stanford-libraries-blog/2018/10/piano-roll-scanner-update.

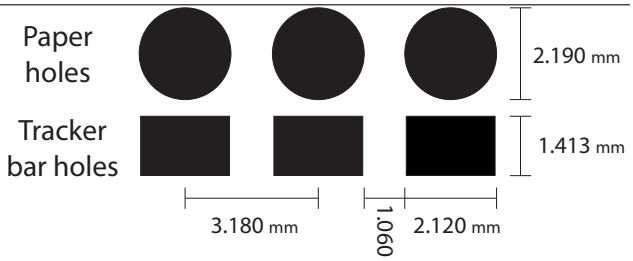


Figure 3: Red Welte paper and tracker bar hole physical dimensions and horizontal spacings (to scale).

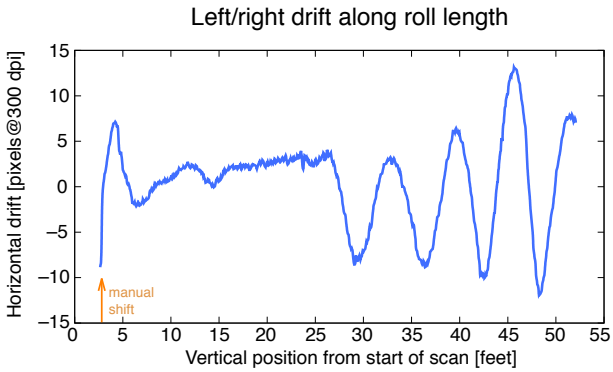


Figure 4: Drift analysis for a sample roll. The total drift range is 25px from left to right. The arrow near the start of the roll indicates a manual shift by the scanner operator.

4.2 Drift Correction and Hole Detection

The paper of an original piano roll can warp due to age and poor storage conditions. In such cases, paper holes may drift and misalign with the corresponding tracker bar hole and trigger the incorrect note. Figure 3 shows the spacing between the rectangular tracker bar holes and the circular holes of the piano roll paper. When the paper shifts by more than one mm (about 13 pixels), a paper hole will start to bleed the vacuum in an adjacent tracker bar hole. If the overlap is large enough, an incorrect note will be triggered on the player piano. Thus, a first step in extracting the musical holes from a roll image involves identifying and cancelling this drift.

Figure 4 plots the drift of a sample roll throughout its length. There is an initial sudden shift of about 15 pixels at the start of the roll caused by the scanning operator moving an adjustment bar. For the next 25 feet the drift is minimal, at only a few pixels. Then starting at around 27 feet, an oscillation with an amplitude of 20 pixels begins to be observed. This oscillation has a frequency of about 5–6 feet and is common in many Welte T-100 scans. We determined this to be due to inconsistencies in the paper, as well as mechanical properties of the scanner relating to the tension and alignment of the supply and take-up spools.

For a sample of 60 red rolls, the average total drift range was 20.8 $\pm$ 10.8 pixels, where the spacing between note columns was 37.75 px. Five of the rolls had a total drift that would cause them to produce incorrect notes if played without correction on a player piano, and 25 had a drift that may cause problems. The largest total drift throughout the

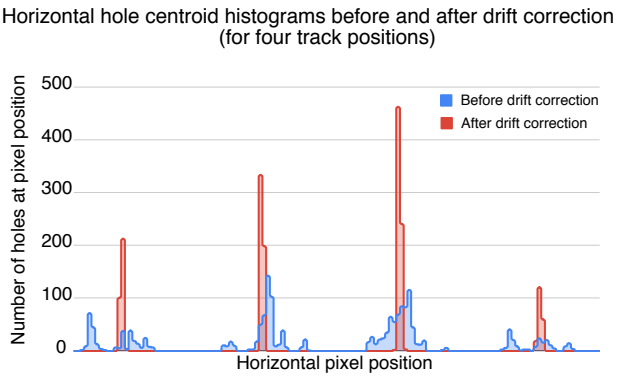


Figure 5: Horizontal positions of hole centroids for four tracks before (blue) and after (red) drift correction.

scan of a roll was 55.7 px, and the smallest was 2.7 px over the length of a roll.

To calculate the drift correction along the length of a piano roll, the edges of the paper are identified, and these edges are low-pass filtered to remove defects such as edge tears from the edge analysis. When the left and right edges of the paper move suddenly in parallel, this indicates that the scanner operator moved the adjustment bar during the scanning process.

This left-right drift is cancelled out by adjusting the horizontal position by the negated drift value at that position on the roll. The centroid (center-of-mass) for each musical hole is identified in the image, and then assigned a corrected position by subtracting the drift offset at the centroid position. Figure 5 shows the results of this drift correction for a single tracker bar position. After adjusting for drift, the hole centroids for a track are clustered within a standard deviation of less than one pixel ($1/300$th of an inch), compared to a range of 18 pixels before the adjustment.

Finally, a Discrete Fourier Transform (DFT) is applied to the drift-corrected hole centroid histogram (window size 4096, zero padding by a factor of 16). The spacing of the hole tracks is obtained by searching for the peak harmonic in the expected region of the DFT. The offset of the spacings is locked to the track position with the most hole punches.

4.3 Defect Analysis

The drift adjustment also allows for increased accuracy in identifying holes that are not quite aligned with any intended tracker hole. Such errors can be removed in this step. Figure 6 illustrates such a problematic hole. In this example, the green-highlighted hole at the bottom left corner of the figure is caused by a tear in the original paper. This hole is not aligned on any of the expected hole centers that are indicated by purple vertical lines. The blue holes indicate alignment as expected.

Other statistical measures of hole shapes are also used to detect aberrant holes. For example, the angle of the major axis of the hole is expected to align with the length of a roll, except for circular single-punch holes. Holes less than $1/6$ of the expected area of a single punch are automatically excluded. These are typically small defects in the

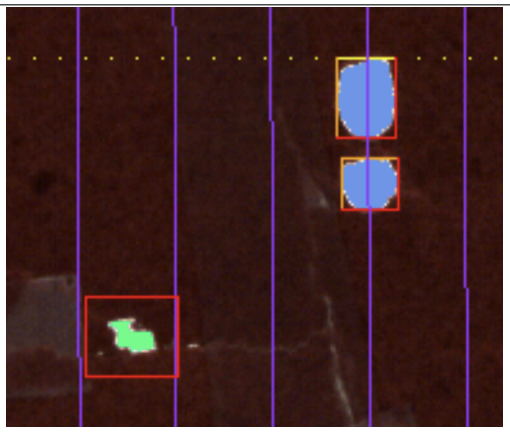


Figure 6: Unintentional hole in paper (bottom left) and intentional holes (upper right).

paper, such as dirt or large pieces of cellulose in the paper that have fallen out over time. Holes wider than the expected spacing between holes are also flagged as potential problems.

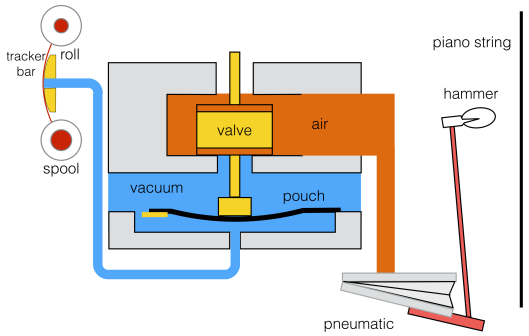
4.4 Bridge Removal

Long holes on piano rolls are often split into several smaller holes to avoid weakening the paper, however they sound as a single longer note when played back on a pneumatic instrument. This process is called “bridging”. Our raw MIDI files retain the bridging information, thus making them suitable for punching new paper copies of the rolls (what are called “recuts") that retain these separations. This detail is also important for master-roll reconstruction analysis. Our expressive MIDI files however remove this bridging to produce the resultant sound of the roll. Adjacent holes are taken to merge into single notes when the spacing between holes is less than $1.37\times$ the diameter of the punches. Figure 6 illustrates such a case where the two blue holes in the upper right corner represent a single note and would be merged.

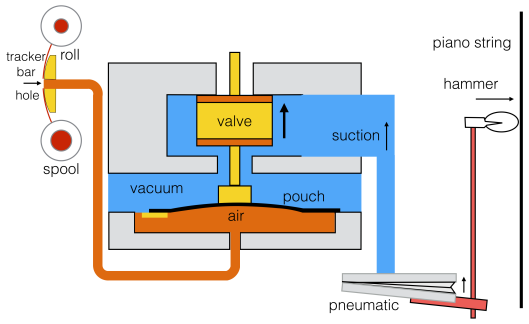
5. ROLL EMULATION

5.1 The Pneumatic System

A player piano is powered by suction, utilizing pneumatic valves that regulate vacuum pressure created by an electric motor. As shown in Figure 7, when a roll passes over the tracker bar, a hole on the paper allows air to leak into the lower section of the valve box, causing the air below to change to atmospheric air pressure. As it moves the valve pouch upwards, the vacuum thus travels to the pneumatic and collapses it, causing the pneumatic to move, which strikes a piano key. However, on a reproducing piano, a further series of pneumatic controls, the “expression box”, regulates the precise rate of change in the suction level of the “dynamic” pneumatic. There is one expression box for each half of the keyboard, thus allowing dynamic changes to only occur on one half at a time, however the changes



(a)



(b)

Figure 7: Basic components in a player piano. (a) When not reading a hole. (b) When reading a hole.

can occur very rapidly over time [4].

Thus, the dynamic pneumatic can do one of the following:

(a) Remain stationary.

(b) Open or close slowly, producing the effect of a slow crescendo or decrescendo (suction level increases or decreases slowly).

(c) Open or close quickly, producing the effect of a fast crescendo or decrescendo (suction level increases or decreases quickly).

5.2 Expression Emulation of Welte T-100 Rolls

The process of emulation involves understanding how the expression box (or equivalent component in other formats) affects suction pressure on the dynamic pneumatics for each key over time. The expression system for Welte T-100 rolls involves ten parameters (including pedals) that can affect independently or in combination the dynamic result at any given time. Table 1 shows the possible parameters and their locations as expression hole tracks. There are parallel holes for each expression regulation on each side of the piano because, as mentioned above, the pneumatic expression mechanism is split in half across the keyboard. The damper (sustain) pedal plays a role in the dynamics because it affects the collective sustain and decay of notes played.

[4] The original technology used by reproducing roll companies to capture dynamic information from the live performer remains unclear. One theory suggests that the dynamics were translated into perforations from expression lines drawn by styli attached to pneumatics that were connected to expression regulators. Others have suggested a combination of electro-pneumatic valves and a dynamic rotor. A more detailed investigation of this question is beyond the scope of this paper but further information can be found in other sources [11, 14].

Bass Section	Treble Section
14: Bass Mezzoforte off	113: Treble Mezzoforte off
15: Bass Mezzoforte on	112: Treble Mezzoforte on
16: Bass Crescendo piano	111: Treble Crescendo piano
17: Bass Crescendo forte	110: Treble Crescendo forte
18: Bass Forzando piano	109: Treble Forzando piano
19: Bass Forzando forte	108: Treble Forzando forte
20: Soft-pedal off	107: Sustain-pedal off
21: Soft-pedal on	106: Sustain-pedal on
22: Motor off	105: Electric cutoff
23: Motor on	104: Rewind
24-66: Notes C1 to F#4	67-103: Notes G4 to G7

Table 1: MIDI note number and corresponding perforation track information (encoded in the raw MIDI files).

We begin by encoding each hole track as a MIDI note number in the raw MIDI file. The function of each hole track on red Welte rolls is summarized in Table 1. Holes of the bass expression tracks (MIDI Note Number 14—19) control the dynamics of notes below F#4 and those of the treble expression tracks (108—113) control the dynamics of notes above G4. Holes for pedal movements (20—21, and 106—107) apply to all notes. Notes (pitches) on the keyboard are holes 24–103.

The "Mezzoforte on" and "Mezzoforte off" hole tracks control a pneumatic hook (which is often referred to as the "mezzoforte hook") that prevents the expression pneumatic from fully opening or closing. The damper (sustain) pedal and una corda (soft) pedal are controlled by lock-and-cancel valves: holes in one track turn on the valve, and holes in an adjacent track turn it off, allowing the hole to have a continuing effect once triggered.

The "Crescendo forte" and "Crescendo piano" hole tracks produce slow crescendos (increasing dynamics). They are also controlled by lock-and-cancel valves. This means once a perforation of "Crescendo forte" is triggered, the dynamic will increase until it is cancelled by "Crescendo piano". The "Forzando forte" and "Forzando piano" produce fast crescendos and decrescendos. They are controlled by a single continuous roll perforation, unlike lock-and-cancel valves, so the pneumatic is powered for the length of the perforation. When the crescendo and forzando tracks are not activated, a steady slow decrescendo results. In fact, a slow decrescendo is effectively constantly activated due to its connection to ambient air pressure.

To calculate the rate at which the crescendo or decrescendo operates, we examined original Welte T-100 test rolls, manuals and other sources [5, 14]. A test roll contains a number of specific note and dynamic tests that allow a technician to adjust the fine regulation of the player piano. The slow crescendo, slow decrescendo, fast crescendo, and fast decrescendo rates are regulated by test 3 through 6 in the T-100 test-roll manual. Test 3 regulates the slow crescendo rate to the mezzoforte hook. Test 4 regulates the fast crescendo and decrescendo. Test 5 regulates the re-

Crescendo type	Travel between	Time (ms)
Slow Crescendo	min. to mezzoforte	1190
Slow Decrescendo	mezzoforte to min.	2380
Fast Crescendo	min. to mezzoforte	200 to 300
Fast Decrescendo	max. to min.	70 to 210

Table 2: Crescendo type and the time it takes to travel between two dynamic levels.

lease from fortissimo touch to piano touch. Test 6 regulates the combination of crescendo and decrescendo. These tests would have been used in conjunction with the judgement of a trained technician. Even as specified, they allow for a small range of results to the tests.

Based on this research, we chose rates for these parameters as shown in Table 2. We have chosen to model pneumatic pressure changes during execution of the slow crescendo and decrescendo as a non-linear function (one-pole filter), whereas that of the fast crescendo and decrescendo is modelled as a linear function. We apply the following constraints to map pneumatic pressures to MIDI velocity levels: the mininum velocity (softest level) is 35, mezzoforte (medium level) is 65, and the maximum (loudest level) is 90. In addition, the overall velocity of bass notes are assigned to be about 5 velocity levels lower than the treble notes.

5.3 Tempo Configuration

Most piano rolls formats are played with a speed written at the start of a roll by the manufacturer; however, Welte T-100 rolls do not have this indication. Test rolls and manuals do suggest that these rolls are to be played at single set speed, however Welte player instruments curiously come with a speed-adjustment lever. It has been theorized that this lever may have allowed for small adjustments needed by poorly regulated instruments. We deduce the general set tempo from an examination of test rolls and sources to be approximately 9.46 ft./min., but accept a potential range of $\pm$ 0.5 ft./min.

Timings of notes in image-extracted MIDI files are expressed in delta-time ticks (pulses) that represent one pixel row in the original image. This allows notes in the MIDI file to be linked to their source locations in the scanned rolls. To translate roll speed into MIDI playback speed, we first set the initial MIDI tempo to 60 BPM. Then we calculate ticks (or pulses) per quarter note (TPQ, or PPQ, in the file header) by multiplying the roll speed with a factor that converts ft./min. into pixels/sec.:

$$\text{factor} = \frac{\text{feet}}{\text{min}} \times \frac{300\text{pixel}}{\text{inch}} \times \frac{12\text{inch}}{\text{feet}} \times \frac{1\text{min}}{60\text{second}} = 60$$

So a speed of 9.46 ft./min. is used to set the TPQ value in the MIDI header to 568.

5.4 Paper Acceleration

The take-up spool on a player piano rotates at a constant speed, while the diameter of the spool increases as paper is wound around it. This causes an acceleration of the paper

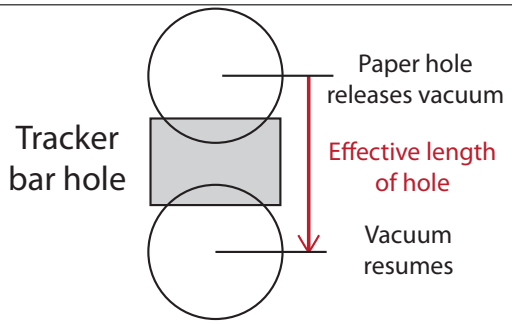


Figure 8: Schematic of paper hole moving across trackbar (not to scale).

over time. Since the MIDI file time unit represents spatial distance on the roll, tempo changes are given in the MIDI file to emulate this acceleration. Based on our research and sources, we set this to be a 0.22% acceleration rate per foot [4]. The result is a more accurate tempo across the length of the roll.

5.5 Hole Extension

Another correction we apply concerns the effective length of paper holes over the tracker bar. The pneumatic valve is actually triggered before the paper hole completely lines up with the tracker bar hole, as enough of the ambient air pressure is generated by a portion of the hole. Thus the effective length of the hole is actually longer than it is on the paper, as illustrated in Figure 8. Compensating for this extension is especially important for the forzando piano and forte expression tracks because they are fast-acting and sensitive to small variations in time. We approximate the extra time the valve is open by measuring the length of a Welte-Mignon tracker bar hole and multiplying it by 0.75.

5.6 Audio Rendition

We provide high-quality audio transfers of the expressive MIDI files in SUPRA-RW by rendering them through Synthogy Ivory II Pianos in Steinberg Cubase. The resulting wav and m4a files should generate a result that compares favorably to a Welte T-100 roll played on an original Welte-Mignon pneumatic instrument.

6. CONCLUSION

In this paper, we describe the creation of the SUPRA-RW dataset which includes scanning, symbolic music extraction, pneumatic system modeling, and performance rendering via emulation. These procedures will apply with some variation (especially in emulation algorithms) to other reproducing roll formats and should be useful for those working to digitize large collections.

There is much work to do with reproducing (and non-reproducing) rolls. Our project will continue to study pneumatic expression mechanisms and we plan to create emulations for other roll formats. This will support online access to the thousands of rolls in Stanford's collection and will grow the SUPRA database.

7. ACKNOWLEDGEMENT

We thank Anthony Robinson for contributing his roll scanner design and expertise to the project. Brett Swope of Swope Design Solutions built the scanner for this project. We thank Peter Phillips and Wayne Stahnke for their generous advice and consultation. The Stanford University Library staff provided expertise in many areas for which we would like to thank Tony Calavano, Doris Cheung, Frank Ferko, Nathan Coy, Jill Sison, Kristen St. John, Peter Crandall and Andree Clair Bonnepart. This project is funded partially through the generosity of an anonymous donor.

8. REFERENCES

[1] International association of mechanical music preservationists (IAMMP). http://www.iammp.org/. Accessed: 06 May 2019.

[2] The laboratory of the italian mechanical music association (AMMI). https://sites.google.com/view/ammi-lab/home. Accessed: 06 May 2019.

[3] Notenrollensammlung des deutschen museums. https://digital.deutsches-museum.de/projekte/notenrollen/. Accessed: 06 May 2019.

[4] American piano company research laboratory documents. Stanford University, Howe Collection of Musical Instrument Literature ARS 0167, 1929.

[5] Description of the various functions of the welte mignon test roll 100. De Luxe Reproducing Roll Corporation, 1963.

[6] Gustavo Colmenares, René Escalante, Juan F. Sans, and Rina Surós. Computational modeling of reproducing-piano rolls. *Computer Music Journal*, 35(1):58–75, 2011.

[7] Gerhard Dangel and Hans Schmitz. Welte-mignon piano rolls: Complete library of the european recordings 1904-1932 for the welte-mignon reproducing piano. Stuttgart, 2006.

[8] Biblioteca Nacional de España. Los rollos de pianola de la bne suenan a través de su web. http://www.bne.es/es/AreaPrensa/noticias2016/0503-Rollos-de-pianola.html. Accessed: 06 May 2019.

[9] Musikinstrumentenmuseum der Universität Leipzig. Tasten. http://mfm.uni-leipzig.de/dt/Forschung/Tastenprojekt.php. Accessed: 06 May 2019.

[10] Christoph E. Hänggi and Kai Köpp. *"Recording the soul of music": Welte-Künstlerrollen für Orgel und Klavier als authentische Interpretationsdokumente?: Symposium Seewen 2013*. Bern: HKB, Hochschule der Künste Bern; Seewen: Museum für Musikautomaten Seewen SO, Sammlung Dr. h.c. Heinrich Weiss-Stauffacher, 2017.

[11] The Pianola Institute. History of the pianola - an overview. http://pianola.org/history/history.cfm. Accessed: 06 May 2019.

[12] Matteo Malosio, Flavio Pedrazzini, and Perego Niccolò. The sisar project. *The Music Box*, 26(6):221–223, 2014.

[13] Department of Music and Stanford University Archive of Recorded Sound. The player piano project. http://library.stanford.edu/projects/player-piano-project. Accessed: 06 May 2019.

[14] Peter Phillips. *Piano rolls and contemporary player pianos: The catalogues, technologies, archiving and accessibility*. PhD thesis, University of Sydney, 7 2017.

[15] Anthony Robinson. Anthony's roll scanning. http://semitone440.co.uk/scanner/. Accessed: 06 May 2019.

[16] Zhengshan Shi, Kumaran Arul, and Julius O Smith. Modeling and digitizing reproducing piano rolls. In *Proc. International Society for Music Information Retrieval*, pages 197–203, 2017.

[17] Wayne Stahnke. A window into time: Rachmaninoff performs his solo piano works. Telarc CD-80489, 1998.

[18] Warren Trachtman. Roll expression emulation software. http://www.trachtman.org/rollscans. Accessed: 06 May 2019.

FAST AND FLEXIBLE NEURAL AUDIO SYNTHESIS

Lamtharn Hantrakul
Google Brain
hanoih@google.com

Jesse Engel
Google Brain
jesseengel@

Adam Roberts
Google Brain
adarob@

Chenjie Gu
Google DeepMind
gcj@

ABSTRACT

Autoregressive neural networks, such as WaveNet, have opened up new avenues for expressive audio synthesis. High-quality speech synthesis utilizes detailed linguistic features for conditioning, but comparable levels of control have yet to be realized for neural synthesis of musical instruments. Here, we demonstrate an autoregressive model capable of synthesizing realistic audio that closely follows fine-scale temporal conditioning for loudness and fundamental frequency. We find the appropriate choice of conditioning features and architectures improves both the quantitative accuracy of audio resynthesis and qualitative responsiveness to creative manipulation of conditioning. While large autoregressive models generate audio much slower than real-time, we achieve these results with a more efficient WaveRNN model, opening the door for exploring real-time interactive audio synthesis with neural networks.

1. INTRODUCTION

Expressive musical instruments, whether digital or acoustic, enable players to use relatively low-dimensional gestures to control perceptual qualities of audio such as pitch, dynamics, and timbre in real time [5, 10]. Progress in *Neural Audio Synthesis*, directly rendering audio with deep neural networks, has revolutionized the field of speech synthesis [13, 15, 18, 19] by replacing hand-designed functions with data-driven design, and is similarly poised to create a brand new class of expressive musical instruments [1, 3, 4, 6, 12].

Much of this progress is due to deep autoregressive models, such as WaveNet [18] and Tacotron [15, 19]. While these models can generate a wide range of realistic audio, using them as expressive musical instruments is not straightforward as they require fine-grained domain-specific conditioning information, such as phonemes [18] or mel-spectrograms [15] for speech, and are prohibitively slow at generating audio.

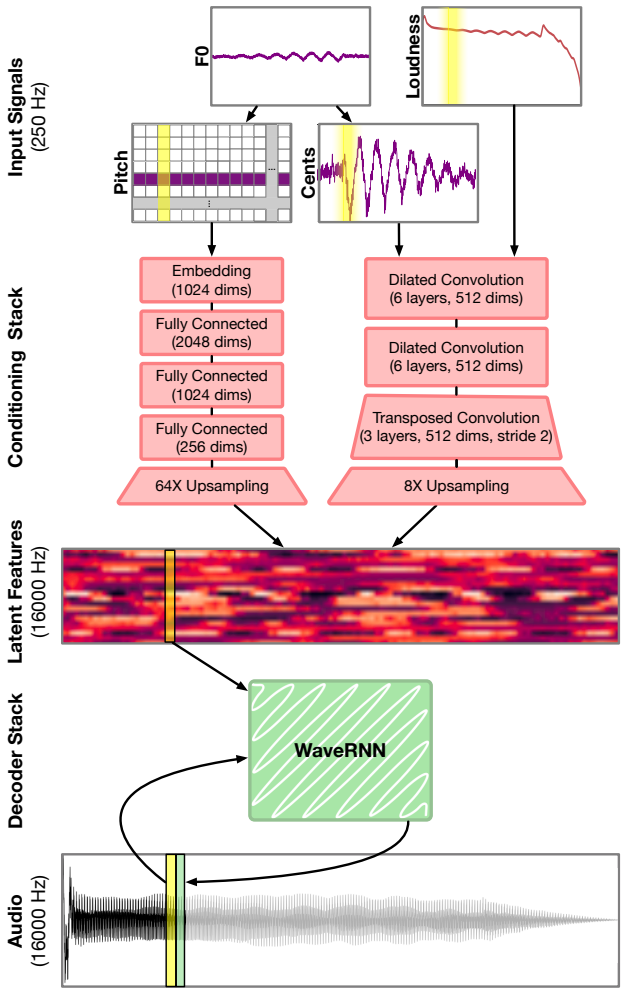


Figure 1. Real-time neural synthesizer architecture. Fundamental frequency (F0) and loudness features are fed through a conditioning stack and upsampled to audio rate. Using our Pitch/Cents representation, F0 is split into pitch one-hot embeddings and continuous deviations in cents. The latent features are concatenated and fed into a WaveRNN which generates audio autoregressively. Audio samples can be heard in the online supplement [1].

In this paper, we make progress in overcoming these challenges by using domain-specific conditioning features to drive a simpler and more efficient WaveRNN [7] model to generate audio of musical instruments. We explore a variety of conditioning features and architectures and demonstrate that a WaveRNN-based model driven by time-distributed and fine-scale musical features is capable of synthesizing realistic audio faster than real time. Our key

findings include:

- *Fine-grain control over loudness.* By conditioning on extracted loudness features, WaveRNNs can resynthesize audio that closely matches the original loudness, from long attacks and decays to fast changes like a tremolo.
- *Fine-grain control over pitch.* By conditioning on extracted fundamental frequency features, WaveRNNs can resynthesize audio that closely matches the original frequencies, from a constant pitch, to subtle vibratos and large glissandos.
- *Feature selection.* Conditioning on perceptual loudness consistently outperforms using amplitude energy, while conditioning on discrete pitches and continuous cents outperforms using a single continuous frequency feature.
- *Real-time generation*. Since the models are built around WaveRNN, even unoptimized kernels synthesize audio faster than real-time in batch mode. We show our results hold for casaul conditioning stacks, opening the door for low-latency interactivity with a properly optimized implementation.

Audio for all examples shown in this paper can be found in the online supplement [1] .

2. RELATED WORK

2.1 Musical Conditioning

WaveNet autoencoders can infer a latent conditioning signal from raw audio [4]. This enables unique opportunities for timbre transfer and morphing [2,12], but the latent code is relatively high dimensional and unintuitive to manipulate due to entanglement of perceptual attributes.

Discrete pianorolls can serve as an intuitive intermediate representation to control generation of realistic polyphonic audio in focused domains such as solo piano performance [6,9]. However, many instruments have dynamic pitch and volume that cannot be captured with discrete pianorolls. In contrast, this work examines using continuous pitch and loudness conditioning to create a synthesizer capable of such dynamic control.

2.2 Fast Synthesis

A successful approach to speeding up generation is rendering audio in parallel. Parallel Wavenet [17] distills a teacher WaveNet model into a student network that generates audio in parallel using inverse autoregressive flows. This dramatically reduces inference latency but requires training several networks and carefully tuning several heuristic losses. Other flow-based models such as WaveGLOW [14] can train audio generation flows directly, but so far have only been successful at inverting spectrograms. Generative adversarial networks provide another approach to parallel generation and GANSynth [3] recently proved four orders of magnitude faster than WaveNet baselines by generating magnitudes and phases in the spectral domain. However, these models are not capable of handling fine-scale conditioning or variable-length sequences considered here.

The streaming nature of autoregressive models makes them uniquely suited for real-time performance. WaveRNN and LPCNet [7,16] are single-layer recurrent neural networks that reduce complexity to generate 24kHz 16-bit speech at speeds up to 4×real time, even on mobile device CPUs. These models can also run inference in a streaming fashion; a critical requirement for interactive and live applications.

WaveRNN in particular, achieves its performance through a set of architectural and engineering innovations, including 1) the representation of 16-bit audio as a tuple of two 8-bit integers, which enables efficient parameterization of the neural network using a dual-softmax layer, 2) special GRU cells for the two 8-bit integers which achieves high quality synthesized audio, and 3) custom kernels on GPUs and CPUs for dense and sparse models respectively.

3. EXPERIMENTAL DETAILS

3.1 Dataset

We focus our work on a smaller subset of the NSynth dataset [4] identical to GANSynth [3]. These total 70,379 examples, comprising mostly of strings, brass, woodwinds and mallets with pitch labels within MIDI range 24-84 (F0 of ∼32-1000 Hz). Each sample is 4 seconds long and sampled at 16KHz, resulting in 64,000 dimensions.

3.2 WaveRNN

Our WaveRNN model consists of a 1280-unit GRU, two 640×512 fully connected layers (projection) and two 512×256 fully connected layers (logits). The output has two size-256 softmax layers, each predicting 8 bits of the audio (16 bits in total). The model size is on par with WaveRNN for speech synthesis [7]. Compared to other fast audio synthesis models (Parallel WaveNet [17], GANSynth [3], and WaveGlow [14]), WaveRNN has a simpler training setup: the training loss is simply negative log-likelihood and the set of hyper-parameters to tune is small.

3.3 Conditioning

Our work explores conditioning with fine-grain pitch and amplitude control. Details of the conditioning architecture are shown in Figure 1. Below we motivate the choice of representation, followed by the choice of conditioning architecture.

3.3.1 Amplitude Representation

We experiment with two representations for amplitude: Root Mean Square (RMS) *Energy* and an A-weighted Log Amplitude *Loudness*.

[1] `http://bit.ly/2GcCPNV`

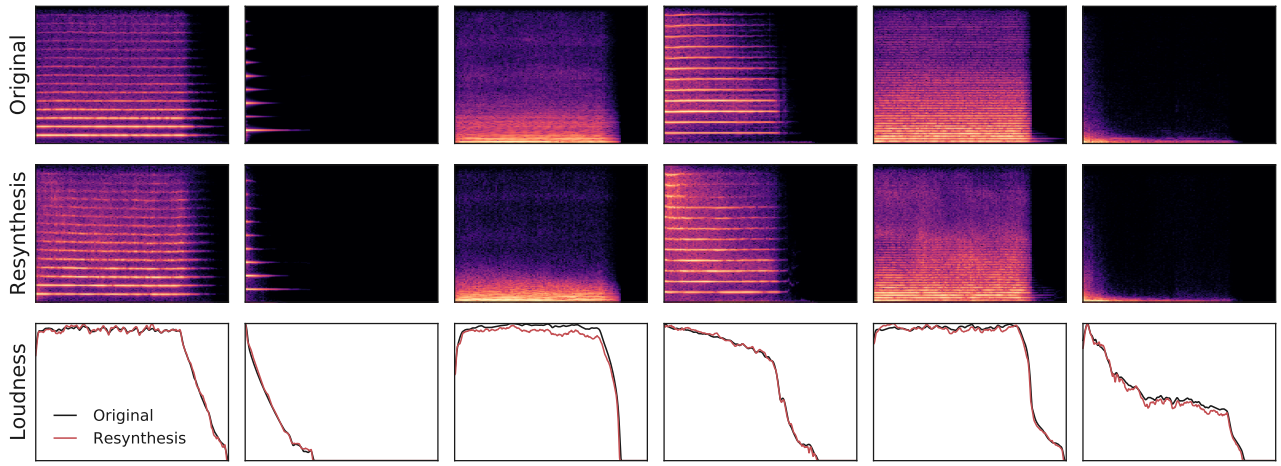


Figure 2. Audio resynthesized from extracted features. Spectrograms of examples from the NSynth dataset and and audio resynthesized from audio features (loudness and fundamental frequency) extracted from the original audio. The bottom row shows the loudness features of the original audio and resynthesized audio. Spectral and loudness contours, and fundamental frequency contours (not shown), are largely reproduced by the resynthesized audio. These samples, and those in the other Figures, were produced with the best performing Loudness + Pitch/Cents conditioned model.

Energy: Energy is computed from the STFT of the waveform with `hop_length=64` and `n_fft=2048`, yielding energy vectors of length 1000 for 4 seconds of audio (250 Hz). For training, we normalize these values across the entire dataset.

Loudness: There are many detailed psychometric models of perceived loudness [11]. For clarity, we opt for a simple A-Weighting of the power spectrum, which places greater emphasis on higher frequencies, followed by log-scaling. The loudness vector is centered and has equal length to energy. The exact computational steps are included in the Appendix.

3.3.2 Amplitude Conditioning

Wave2Midi2Wave [6] successfully conditioned a WaveNet to generate realistic piano audio based on a convolutional stack encoding a pianoroll. We adopt a similar architecture for encoding amplitude. The conditioning network consists of a stack of 12 dilated convolution layers (with dilations 1, 2, 4, 8, 16, 32, 1, 2, 4, 8, 16, 32), followed by three transposed convolutions with stride 2. All convolution layers have 512 filters with kernel size 3. For causal conditioning, we simply zero-pad and shift the receptive fields to not include future context.

3.3.3 Fundamental Frequency

We use CREPE [8], a recent and data-driven pitch tracking model with state-of-the-art performance. With hop size of 64, CREPE produces a vector of frequencies of length 1000. We convert these into MIDI pitch using librosa's `hz_to_midi()` function. From this point, we explore different representations of pitch.

Normalized Frequency: Dividing the vector of MIDI values by 127.0 yields a normalized vector representing pitch on a linear scale.

Octaves/Notes/Cents: We experiment with splitting the F0 vector into an octave one-hot $\mathbb{R}^5$, note one-hot $\mathbb{R}^{12}$ and a continuous vector of floats for cents. Each component is a vector of length 1000.

Pitches/Cents: Our most effective representation consists of a pitch one-hot $\mathbb{R}^{60}$ and a continuous vector of floats for cents. Each component is a vector of length 1000.

3.3.4 Fundamental Frequency Conditioning

For normalized frequency, we use the same convolutional stack as for amplitude features.

For Octave/Notes/Cents and Pitches/Cents, we use separate conditioning stacks for the one-hot component and continuous component (shown in Figure 1). The vector of continuous cents is encoded using the same convolutional stack as amplitude features.

For one-hot features, we use a mulitilayer perceptron (MLP) encoder network. The one-hots first pass through an embedding layer and then projected via a series of fully connected (FC) layers with RELU non-linearity into a final target dimensionality. For Octave/Notes/Cents, the two branches of encoders for octave and notes each use embedding units 1024 and FC units 2048, 1024, 512. The final octave and note activations are concatenated to produce a tensor of shape (1024, 1000). For Pitches/Cents, a single pitch encoder is used with embedding units 1024 and FC units of 2048, 1024 and 256. The final activation shape is (256, 1000). Unlike the convolutional stacks, the one-hot MLP encoding stacks are causal by definition.

3.4 Conditioning the WaveRNN

All conditioning features are time-series with lower sampling frequency (250Hz) than audio. To be used as local conditioning vectors by WaveRNN they are upsampled via

replication to 16 KHz, concatenated and added as an additional bias term in the GRU gate equations.

3.5 Training

During training, the model is tasked with resynthesizing the high dimensional audio as accurately as possible, combining the low dimensional conditioning inputs and teacher-forced previous "outputs" to autoregressively make next-step predictions. We minimize the negative log-likelihood loss of the WaveRNN's coarse and fine logits, which is computed via softmax cross-entropy.

The system was implemented in TensorFlow and each model trained on 1 Million steps. We use the ADAM optimizer with epsilon of 1e-8, momentum of 0.9, max gradient norm of 1.0 and an exponential moving average rate of 0.9999. Learning rate decay is 0.7 for every 100k steps. We fine-tuned models over batch sizes of 128, 256 and 512 and learning rates of either 9e-5 or 1e-4, and report best performing models.

4. METRICS

Resynthesis: Since log-likelihood is not a direct measure of sample quality, we quantitatively evaluate our models through the task of resynthesis. Audio features are extracted from the source audio and used to synthesize a new sound with the same features. Some examples of resynthesis are shown in Figure 2. We take the L_1 distance of extracted features between the orignal and resynthesized audio to be the fine-scale perceptual error in resynthesis. Aditionally, since the NSynth dataset has labels for pitch and instrument family (a proxy for timbre), we use these labels to train a classifer which we use to evaluate the global similarity of the resynthesis. The classifier has identical structure to the ones implemented in [4] and [3], and we provide complete details in the Appendix. We calculate each metric on resynthesized audio from the full test set of 17,600 samples.

Loudness L_1 distance: The loudness vector is extracted from the synthesized audio and L_1 distance computed against the input's conditioning loudness vector (ground truth). A better model will produce lower L_1 distances, indicating input and generated loudness vectors closely match. Note this distance is *not* back-propagated through the network as a training objective.

F0 L_1 distance: Pitch tracking using CREPE, like with any pitch tracker, is not completely reliable. Instabilities in pitch tracking, such as sudden octave jumps at low volumes, can result errors not due to model performance and need to be accounted.

CREPE outputs a useful confidence in its prediction of F0 for every frame. By examining the accuracy of pitch tracking on ground truth audio, we found that applying a confidence threshold of 0.85 filtered out areas of unreliable pitch tracking and yielded the best trade-off against false positives. Only F0 from time frames above this threshold are considered in our analysis. For models representing pitch using Octaves/Notes/Cents and Pitches/Cents, we recompute the equivalent normalized frequency. The F0 L_1 distance is reported in MIDI space for easier interpretation; an average F0 L_1 of 1.0 corresponds to a semitone difference.

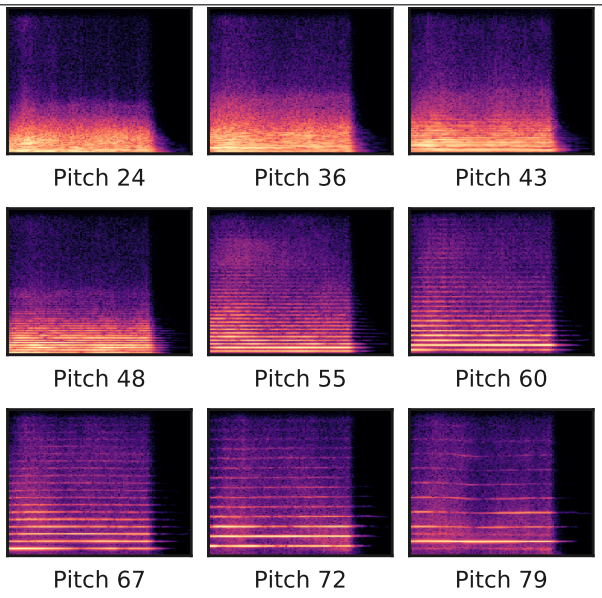


Figure 3. Synthesizing audio at different pitches with the same loudness envelope.

F0 Outliers: Audio examples with F0 confidence below 0.85 for the full length of the example indicate a failure of pitch tracking and are considered "outliers" of the measurement. 398 *ground truth* audio samples in the test set are categorized as outliers, producing a baseline for this metric of $398/17600 = 0.02$. For *generated* audio, examples completely below the 0.85 confidence threshold, are similarly removed. By inspection, we also found L_1 distances above an octave to correspond to pitch tracking failures and remove the samples as outliers. Better performing models have lower values close to the 0.02 baseline.

Pitch Error: The classifier pitch prediction gives a more global measurement of pitch correspondence during resynthesis. On ground truth samples, the classifier has a 0.06 error rate, which is the best that generated samples can hope to achieve.

Instrument Family Error: Instrument family labels are a rough measure of timbral similarity. We assume that if the original and resynthesized audio have similar timbre then they should be classified as the same instrument family. Although the classifier is not perfect, with an error rate of 0.22, it is state-of-the-art for the task and dataset, providing a rough measure of relative performance between models.

5. RESULTS AND DISCUSSION

5.1 Resynthesis

Table 1 shows the quantitative metrics for models with different conditioning on the resynthesis task. Models with causal conditioning are given in parentheses next to their non-causal equivalents.

	Signal Processing	CREPE		NSynth Classifier	
	Loudness (L_1)	**F0 (L_1)**	**F0 Outliers**	**Pitch Error**	**Family Error**
Loudness	0.14 (0.33)	6.20 (6.89)	0.57 (0.66)	0.98 (0.99)	0.75 (0.85)
Loudness + Normalized Frequencies	0.16 (0.24)	1.42 (4.17)	0.07 (0.15)	0.35 (0.81)	0.60 (0.82)
Loudness + Octave/Note/Cents	0.11 (0.14)	1.21 (1.81)	0.07 (0.08)	0.33 (0.25)	0.61 (0.78)
Loudness + Pitch/Cents	**0.10** (**0.10**)	**0.94** (**1.00**)	**0.06** (**0.07**)	**0.16** (**0.12**)	**0.53** (**0.72**)
Ground Truth	-	-	**0.02**	**0.06**	**0.22**

Table 1. Ablation study showing detailed conditioning improves resynthesis accuracy. Loudness is extracted directly from the audio as described in Section 3.3.1. CREPE is used for tracking fundamental frequency (F0), and a classifier pretrained on the NSynth dataset is used to predict pitch and instrument family (see Appendix). Models with causal convolutions in their conditioning stacks have their numbers in parentheses, while the rest use non-causal convolutions. Conditioning on F0 in Pitch/Cents tuples outperforms both Octave/Note/Cents tuples and frequencies as normalized floats, and the trend holds for both causal and non-causal conditioning.

F0 representations: A clear trend is present across all metrics: performance improves as F0 conditioning moves from Normalized Frequency, to Octave/Note/Cents, to Pitch/Cents representations. Interestingly, this improvement is not only present in the frequency-based metrics (*F0 L_1, F0 Outliers, Pitch Error*), but in the metrics for volume and timbre as well (*Loudness L_1, Family Error*).

Qualitatively, we found the models to have different failure modes. Models trained with Octaves/Notes/Cents conditioning held correct fundamental frequencies more reliably than Normalized Frequency, but would on rare occasions be completely offset by a large and erroneous interval for the length of the note. This ambiguity seems to arise from discontinuities at octave boundaries, such as between MIDI notes B2 and C3 that are close in absolute frequency. While Normalized Frequency does not suffer from this effect, we found many models trained on this representation produced audio that would dip "flat" in frequency. This could be due to the reduced effective range of input, which needs to cover the entire range of frequencies on a normalized continuous scale. Unlike speech, slight deviations in frequency are perceived as notes being out of tune.

Non-causal vs Causal: The trends in F0 conditioning are reinforced by the fact they are shared between both causal and non-causal variants of the models. Non-causal conditioning allows incorporating future information into current predictions which helps performance in almost cases. Despite this, the best performing models see less of a performance drop, which is promising for future low latency applications. Increasing the capacity of the causal encoder stack may achieve parity in performance.

Energy vs Loudness: Table 1 in the Appendix justifies the preference for loudness over energy as a conditioning signal. We compare variants of the best performing model (Loudness + Pitch/Cents vs. Energy + Pitch/Cents) and find the loudness-conditioned model outperforms the energy-conditioned model on all metrics. It is worth emphasizing that Loudness L_1 is a fair evaluation metric in this case because it is not used as a loss during training and is more aligned to human perception.

Missing Conditioning: Finally, Table 1 in the Appendix also compares a model conditioned only with Loudness to one conditioned only with Pitch/Cents. Predictably, each does better than the other on their respective metrics, and fails on the other complementary metrics, demonstrating that both levels of conditioning are required. Interestingly, the Instrument Family Error is lower for the loudness-conditioned model, indicating aspects of timbre are likely more highly correlated with loudness contours than pitch contours.

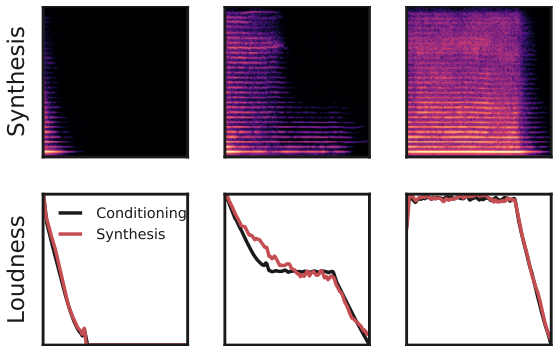


Figure 4. Interpolating in loudness conditioning.

Timbre: In the absence of timbre conditioning, the models learns to correlate timbre with pitch and loudness contours in the NSynth dataset. For example, the combination of a short decay and high F0 vector is a mallet-like sound whereas a long decay and low F0 vector is a cello-like sound. This is evidenced by the reduced Instrument Family Error in tandem with lower Loudness and F0 L_1 distances. Naively adding instrument family conditioning and spectrogram conditioning did little to improve the metrics or control of generated timbre. We believe exploring new methods of timbre control is a rich area for future research and would enable applications like interactive instrument morphing.

5.2 Creative Conditioning

We perform qualitative studies with modified out-of-dataset conditioning vectors. All examples are generated with the Loudness + Pitch/Cents model, and audio for all examples can be found in the online supplement [1] .

Interpolation of Loudness vectors: On the bottom row of of Figure 4, we show three loudness vectors. The left and right vectors were selected from the test set to

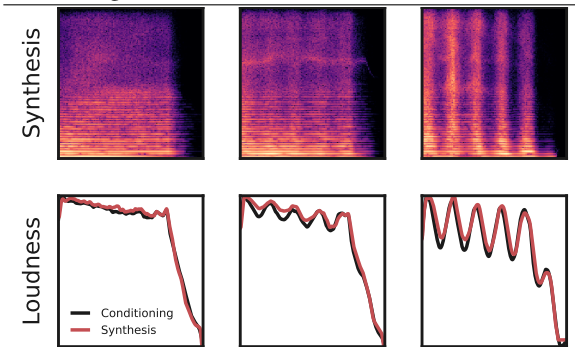


Figure 5. Applying tremolo to loudness conditioning.

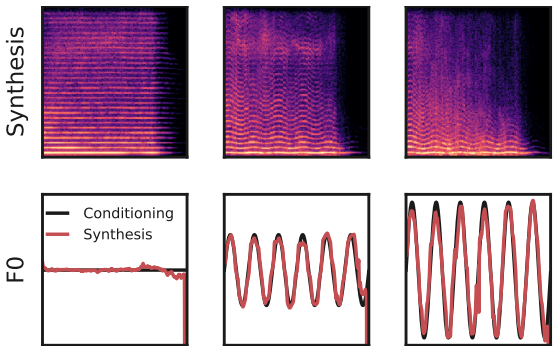


Figure 6. Applying vibrato to frequency conditioning. F0 extracted with CREPE.

demonstrate two extremes: fast decay (left) and long sustain (right). The middle is a synthetic linear interpolation of the two vectors. The top row shows spectrograms for audio synthesized by our model conditioned on each of these loudness vectors with the same F0 vector held at constant pitch. The model is able to closely adhere to the loudness curves even for the interpolated example. More importantly, the spectrogram reveals how loudness conditioning functions more than a naive "amplitude envelope", since the harmonic content changes non-linearly with the loudness signal. This rich behavior draws an analogy to the behavior of real acoustic instruments, where varying excitation introduces rich non-linear changes to the harmonic spectra based on the characteristics of the instrument.

Tremolo: Figure 5 shows the spectrogram (top) and extracted loudness vector (bottom) for an example with increasing an increasing intensity of tremolo (left-to-right) added to the original loudness vector. The generated audio closely tracks the loudness contours through diverse modulation of harmonic content. Note how the reduction in power is uneven across the frequency spectrum, dampening higher harmonics more than fundamental frequencies. A naive multiplicative tremolo would reduce power equally across all frequency bands.

Vibrato: Figure 6 shows audio synthesized from a baseline F0 vector with constant pitch. The middle vector adds a tremolo of a semitone while the rightmost vector adds a tremolo of about 2 semitones. The loudness vector is held constant in the synthesized audio. As seen by oscillations in frequency of the corresponding spectrograms, the model can generate audio reflective of increasing intensities of vibrato.

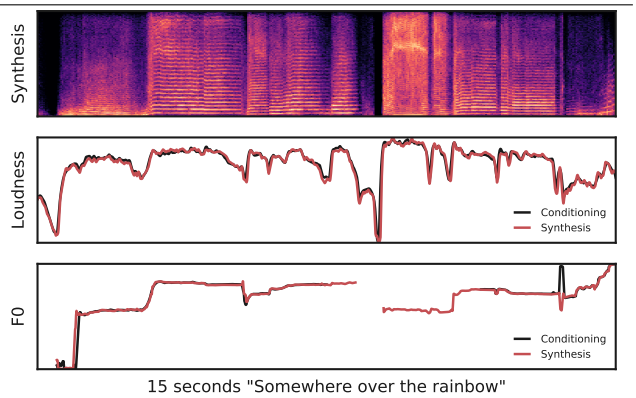


Figure 7. Re-synthesis of out-of-domain input contours extracted from a live vocalist singing "Somewhere over the rainbow". Break in F0 corresponds to silence.

Out-of-domain inputs: Figure 7 shows audio re-synthesized from conditioning signals extracted from a vocalist singing "Somewhere Over the Rainbow". The model was never trained on sequences longer than 4 seconds, nor samples with fast-moving amplitude and pitch variations such as the jump in "Some-*where*". Nonetheless, the model is able to generalize and synthesize audio tightly following these modulations. This opens the door for a variety of interactive applications. The user can provide input contours extracted from live singing, guitar playing or generate these directly from a MIDI Polyphonic Expression (MPE) controller or touchscreen interface.

5.3 Generation Speed

For this work, we draw the distinction between *"real-time throughput"* (producing x seconds of audio in wall time less than or equal to x seconds), and *"low-latency generation"* (producing audio with little to no delay from a conditioning input).

The original WaveRNN paper [7] achieved both through systems optimizations of the underlying kernels. These optimizations motivate our use of WaveRNN for future applications, but are not yet implemented in this paper. Despite this, even with unoptimized kernels, we see dramatic speedups over traditional WaveNet models and are able to achieve faster than real-time throughput speeds for batches of audio on commonly available hardware. For example, our best performing model generates 82 seconds of audio (batch size 21) in 60 seconds on an NVIDIA GTX 1080 GPU ($\sim 1.4\times$ real time).

6. CONCLUSION

In this work, we demonstrate state-of-the-art synthesis of musical instrument sounds with fine-grain temporal control over loudness and pitch. The model learns tight correlations between loudness and pitch, being able to introduce non-linear spectral modulations beyond a naive tremolo or vibrato. The comparable performance between non-causal and causal models points towards streaming applications such as a low-latency re-synthesis guitar pedal or live vocal effect.

7. REFERENCES

[1] Alexandre Défossez, Neil Zeghidour, Nicolas Usunier, Léon Bottou, and Francis Bach. SING: symbol-to-instrument neural generator. *CoRR*, abs/1810.09785, 2018.

[2] Jesse Engel. Hands on, with nsynth super. `https://magenta.tensorflow.org/nsynth-super`. Accessed: 2019-03-01.

[3] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis. *CoRR*, abs/1902.08710, 2019.

[4] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders. *CoRR*, abs/1704.01279, 2017.

[5] Neville H Fletcher and Thomas D Rossing. *The physics of musical instruments*. Springer Science & Business Media, 2012.

[6] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. *CoRR*, abs/1810.12247, 2018.

[7] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aäron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis. *CoRR*, abs/1802.08435, 2018.

[8] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello. Crepe: A convolutional representation for pitch estimation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 161–165. IEEE, 2018.

[9] Rachel Manzelli, Vijay Thakkar, Ali Siahkamari, and Brian Kulis. Conditioning deep generative raw audio models for structured automatic music. In *ISMIR*, 2018.

[10] Eduardo Reck Miranda and Marcelo M Wanderley. *New digital musical instruments: control and interaction beyond the keyboard*, volume 21. AR Editions, Inc., 2006.

[11] Brian CJ Moore, Brian R Glasberg, and Thomas Baer. A model for the prediction of thresholds, loudness, and partial loudness. *Journal of the Audio Engineering Society*, 45(4):224–240, 1997.

[12] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. A universal music translation network. *CoRR*, abs/1805.07848, 2018.

[13] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan Ömer Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller. Deep voice 3: 2000-speaker neural text-to-speech. *CoRR*, abs/1710.07654, 2017.

[14] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. *CoRR*, abs/1811.00002, 2018.

[15] R. J. Skerry-Ryan, Eric Battenberg, Ying Xiao, Yuxuan Wang, Daisy Stanton, Joel Shor, Ron J. Weiss, Robert Clark, and Rif A. Saurous. Towards end-to-end prosody transfer for expressive speech synthesis with tacotron. In *ICML*, 2018.

[16] Jean-Marc Valin and Jan Skoglund. Lpcnet: Improving neural speech synthesis through linear prediction. *CoRR*, abs/1810.11846, 2018.

[17] Aäron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C. Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. Parallel wavenet: Fast high-fidelity speech synthesis. *CoRR*, abs/1711.10433, 2017.

[18] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *Arxiv*, 2016.

[19] Yuxuan Wang, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc V. Le, Yannis Agiomyrgiannakis, Robert Clark, and Rif A. Saurous. Tacotron: Towards end-to-end speech synthesis. In *INTERSPEECH*, 2017.

Session E

DEEPSRGM - SEQUENCE CLASSIFICATION AND RANKING IN INDIAN CLASSICAL MUSIC WITH DEEP LEARNING

Sathwik Tejaswi Madhusudhan
University of Illinois, Urbana Champaign
stm4@illinois.edu

Girish Chowdhary
University of Illinois, Urbana Champaign
girishc@illinois.edu

ABSTRACT

A vital aspect of Indian Classical Music (ICM) is *Raga*, which serves as a melodic framework for compositions and improvisations alike. Raga Recognition is an important music information retrieval task in ICM as it can aid numerous downstream applications ranging from music recommendations to organizing huge music collections. In this work, we propose a deep learning based approach to Raga recognition. Our approach employs efficient prepossessing and learns temporal sequences in music data using Long Short Term Memory based Recurrent Neural Networks (LSTM-RNN). We train and test the network on smaller sequences sampled from the original audio while the final inference is performed on the audio as a whole. Our method achieves an accuracy of 88.1% and 97 % during inference on the *Comp Music Carnatic dataset* and its 10 Raga subset respectively making it the state-of-the-art for the Raga recognition task. Our approach also enables sequence ranking which aids us in retrieving melodic patterns from a given music data base that are closely related to the presented query sequence.

1. INTRODUCTION

Carnatic and Hindustani music are the two main branches of Indian Classical Music (ICM), which underlies most of the music emanating from the Indian subcontinent. Owing to the contemplative and spiritual nature of these art forms and varied cultural influences, a lot of emphases is placed on melodic development. Raga, which governs various melodic aspects of ICM, serves as a framework for compositions and improvisations. Krishna et al. [20] define a Raga as the "collective melodic expression that consists of phraseology which is part of the identifiable macro-melodic movement". A major portion of contemporary Indian Music including film, folk and other forms of music heavily draw inspiration from ICM [10, 12].

Numerous melodic attributes make Ragas distinctive in nature, such as the *svara* (roughly, a musical note), the *gamaka* (for example oscillatory movement about a given note, slide from one note to another etc [17]), *arohana* and *avarohana* (upward and downward melodic movement) and melodic phrases/ motifs [12, 20]. Emotion is another attribute that makes Ragas distinctive. Balkwill et al. [2] conducted studies on the perception of Ragas and observed that even an inexperienced listener is able to identify emotions portrayed by various Ragas. Given the importance of Raga in ICM, Machine Learning (ML) based automatic raga recognition can aid in organizing large audio libraries, labeling recording in the same, etc.

We believe automatic raga recognition has tremendous potential in empowering content-based recommendation systems pertaining to ICM and contemporary Indian music. However, Raga recognition is not a trivial problem. There are numerous examples where two or more Ragas have the same or a similar set of notes but are worlds apart in the musical effect they produce due factors like the gamaka, temporal sequencing (which has to abide by the constraints presented in the arohana and avarohana), as well as places of svara emphasis and rest. Raga identification is an acquired skill that requires significant training and practice. As such, automatic Raga classification methods have been widely studied. However, existing methods based on Pitch Class Distributions (PCD) disregard the temporal information and are highly error-prone, while other methods (reviewed in detail in Section 2) are highly dependent on preprocessing and hand made features, which limit the performance of such methods.

In this work, we introduce a deep learning based solution to Raga recognition and Raga content-based retrieval. First, we reformulate the problem of Raga recognition as a sequence classification task performed using an LSTM RNN based architecture with attention. We then show that the same network, with minimal fine-tuning based on the triplet margin loss [27], can be used for sequence ranking. We introduce sequence ranking as a new sub-task of automatic Raga recognition. In this configuration, the model can be used to perform Raga content-based retrieval. A user provides the model with a query sequence and the model returns sequences that are very closely related to the presented query. This can aid in downstream applications like recommendation systems and so on.

Deep learning has proven to be an indispensable asset due to its flexibility, scalability, learning abilities, end to end training, and most importantly, unprecedented success in modeling unstructured spatial and temporal data over the past few years. Our work draws inspiration from the

successes in [26, 27] introduce a convolutional neural network based recommendation system for music which overcomes and outperforms various drawbacks of traditional recommendation systems like collaborative filtering, a bag of words method, etc. It also leverages LSTM-RNN and attention based models which have proven to be extremely effective in tasks like sequence classification, sequence to sequence learning, neural machine translation, image captioning task, etc., all of which directly deal with sequences.

To summarize, the main contributions of our work are as follows :

- We present a new approach to Raga recognition using deep learning using LSTM based RNN architecture to address problem of Raga recognition.
- With our approach we obtain 97.1 % on the 10 Raga classification task and 88.1 % accuracy on the 40 Raga classification task on the Comp Music Carnatic Music Dataset (CMD), hence improving the state-of-the-art on the same.
- We introduce sequence ranking as a new sub task of Raga recognition, which can be used in creating Raga content based recommendation systems.

Note that we refer to our Raga recognition (i.e sequence classification) model as SRGM1 and the sequence ranking model as SRGM2 throughout the paper.

2. RELATED WORKS

Broadly speaking, ML methods for Raga classification can either be PCD based or sequence based. Many of the previous works have focused on PCD based methods for Raga recognition [6, 7, 18] with Chordia et al. [4], currently holding the best performing results for PCD based methods with an accuracy of 91%. [17] presents an in-depth study of various distribution based methods for Raga classification. Although an intuitive and an effective approach, the major shortcoming of PCD based methods is that it disregards temporal information and hence is error prone. To overcome this, previous works have used Hidden Markov Models based approach [19], convolutional neural networks based approach [21], arohana avarohana based approach [23] among various other methods.

[9, 24] employ Raga phrase-based approach for Raga recognition. [11] use an interesting phrase-based approach inspired by the way in which seasoned listeners identify a Raga. This method is currently the state-of-the-art on a 10 Raga subset of CMD. [12] introduces Time Delayed Melodic Surface(TDMS) which is a feature based on delay co-ordinates that is capable of describing both tonal and temporal characteristics of the melody given a song. With TDMS, the authors achieve state-of-art performance on the CMD.

While the PCD based methods ignore the temporal information, many works like assume that the audio is monophonic. Works like [19,23] attempt to take advantage of the temporal information by extracting arohana and avarohana.

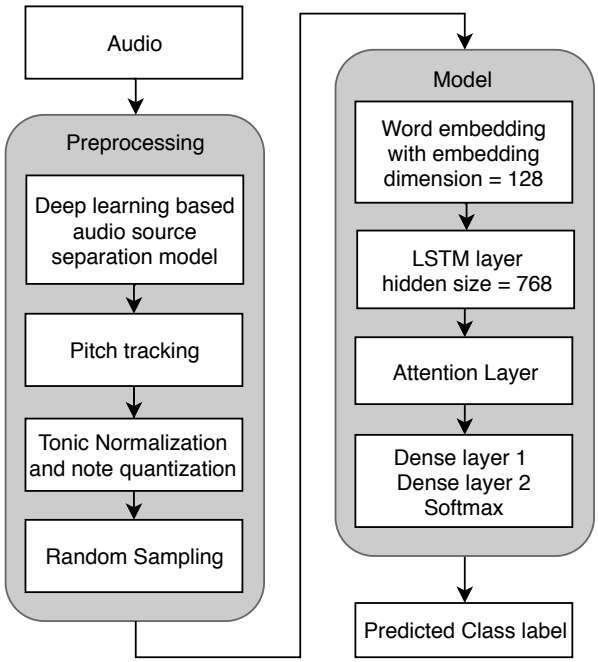


Figure 1. Figure shows various preprocessing steps and model architecture for SRGM1 (refer Section 3)

Although arohana and avarohana are critical components to identifying a Raga, they do not contain as much information as the audio excerpt itself. Most previous works heavily rely on prepossessing and feature extraction, most of which are handcrafted, which can prove to be a limitation as it only gets harder to devise ways to extract complex and rich features.

3. RAGA RECOGNITION AS A SEQUENCE CLASSIFICATION PROBLEM (SRGM1)

Sequence classification is a predictive modeling problem where an input, a sequence over space or time, is to be classified as one among several categories. In modern NLP literature, sentiment classification is one such example, where models predict sentiments of sentences. We argue that the Raga recognition is closely related to the task of sequence classification, since any audio excerpt in ICM involving vocals/melody instruments, can be broken down as a sequence of notes spread over time. To reformulate Raga recognition as a sequence classification task, we treat the notes obtained via predominant melody estimation as words, the set of all words form the vocabulary and each Raga as a class. We detail our approach to raga recognition as below.

3.1 Preprocessing

3.1.1 Deep Learning based audio source separation

Audio recordings available as part of CMD accurately represent live ICM concerts with vocalists accompanied by various string, wind, and percussion instruments. How-

ever, for the purposes of raga recognition, analyzing the vocals is sufficient. We believe that the presence of other elements in the audio interferes with the performance of the classifier and hence we include audio source separation as a preprocessing step. We use Mad-TwinNet [8], a deep neural network based approach capable of recovering vocals from a single channel track comprised of vocals and instruments. It uses a Masker Denoiser architecture combined with twin networks (a technique used to regularize recurrent generative networks). We use the pre-trained model made publicly available by the authors.

3.1.2 Pitch tracking

Our model analyzes the melody content of audio excerpts to perform Raga recognition, hence pitch tracking is an essential preprocessing step. To perform pitch tracking, we use the python API [15] for Praat [3] which is an open source software for the analysis of speech and sound.

3.1.3 Tonic Normalization and note quantization

ICM utilizes a relative scale that is based on the Tonic note of the performer. Since the Tonic can vary in different renditions of the same raga, recognizing and accounting for the tonic note is extremely important. [11] have made numerous features, including Tonic of audio excerpts, available as part of CMD. We normalize the pitch tracked array of every audio in CMD by using the following:

$$f_n = 1200 * log_2(f/T) \tag{1}$$

where f is the frequency after pitch tracking (in Hz), T is the frequency of the Tonic (in Hz) and f_n is the tonic normalized frequency. The above expression results in 100 cents in a half step, for instance, E and F. Empirically we observe that it is sufficient to have just 5 levels in a half step. However, we leave this as a hyperparameter on the model. Hence the expression for obtaining a tonic normalized note is given by:

$$f_p = \text{round}(1200 * log_2(f/T) * (k/100)) \tag{2}$$

where k is number of desired levels in a half step. For instance, $k = 5$ when number of desired levels between two consecutive notes are 5.

3.1.4 Random sampling

The pitch tracked sequences of audio excerpts from CMD have a length of at least $5 * 10^5$ steps. It is impractical to train a Recurrent neural network on sequences with such length as it increases training time considerably. Also, the network would struggle to retain information over such large sequences. Hence, we train the network on *subsequences*, which are smaller sequences randomly sampled from the pitch tracked audio excerpt. It is random in that the start index is randomly selected. We sample N_r number of times from every audio excerpt. Length of the subsequence is chosen carefully as it impacts the models training considerably. We observe that smaller sequences tend to confuse the model. An ideal value for the length of the subsequence is 4000-5000 steps.

3.1.5 Choosing an appropriate value for N_r

Eq (3), determined empirically, gives the appropriate value for N_r, based on the length of the subsequence L_r and maximum of the set of lengths of all pitch tracked audio excerpts L_{max}.

$$N_r = \lceil 2.2 * L_{max}/L_r \rceil \tag{3}$$

3.2 Model Architecture

At the core of our model (refer figure 1) is the LSTM [13, 22] based recurrent neural network, which is a popular choice for sequence classification and sequence to sequence learning tasks. We experiment with various configurations for the LSTM block and emperically find that a single LSTM layer with a hidden size of 768 works the best. The word embedding layer, which appears prior to the LSTM layer, converts each note that appears in the subsequence into a 128-dimensional vector. The LSTM block is followed by an attention layer. Our implementation closely follows soft alignment as described in [1]. The attention layer is followed by two densely connected layers, the first of which has 384 hidden units and the second one has hidden units equal to the number of classes represented in the training dataset. This is followed by a softmax layer, which converts the output of the final dense layer into a probability distribution over the given number of classes i.e, a vector of length equal to the number of classes and sums to one.

3.3 Model training and Optimization

We train the model for over 50 epochs, with a batch size of 40 (i.e, 40 subsequences are fed to model at each training step) and an initial learning rate of 0.0001. We employ dropout [25] in the dense layer and batch normalization [14] after the LSTM layer to reduce over fitting.

3.3.1 Loss Function

Since we are dealing with a classification problem with number of classes greater than 2, the categorical cross entropy is the best suited loss function. The categorical cross entropy loss (CCE) is computed as follows:

$$\text{CCE} = -\sum_{i=1}^{N} y_{i,j} * log(p_i) \tag{4}$$

Where N is the number of classes, $y_{i,j}$ is an indicator function which is 1 only when $i = j$, j is the training label for the given subsequence, p_i is the i^{th} entry of the probability vector (which is the output of the model).

3.3.2 Optimization

Random sampling dramatically increases the size of the data set (the size would be 480*N_r, where 480 is the number of recordings in CMD). This makes it hard to fit all these samples on a single computer. Hence we use the Distributed asynchronous stochastic gradient descent training algorithm [5]. We use the Adam [16] optimizer to update the gradients during training.

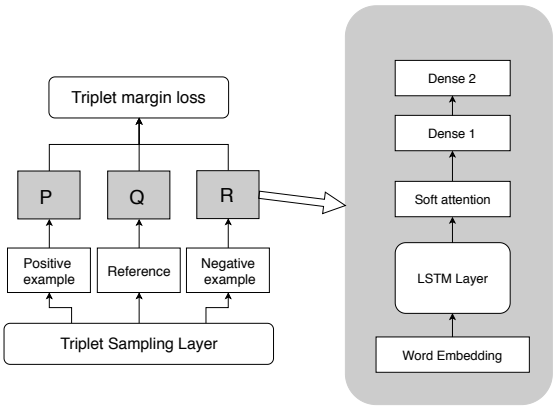


Figure 2. Schematic diagram for the sequence ranking algorithm. P, Q and R are the copies of the same model and hence have the same architecture.

4. SEQUENCE RANKING AND RETRIEVAL (SRGM2)

A fully trained sequence ranking model will be able to create feature embeddings such that the L2 distance between the feature embeddings of a pair of sequences is directly proportional to how similar the sequences are. In this work we fine tune a pre-trained Raga recognition model (SRGM1, from the previous section) using the triplet margin loss and evaluate its ability in retrieving subsequences similar to query subsequence. The demonstrated ability to retrieve similar sounding musical pieces is a unique and highly useful feature of our method.

4.1 Preprocessing

The preprocessing steps used in the training of SRGM1 apply to sequence ranking. We start with audio source separation followed by pitch tracking, tonic normalization and finally random sampling.

4.2 Model architecture

The network architecture of the sequence ranking model is shown in 2. P, Q, and R are modified copies of pre-trained SRGM1 network (Note: the diagram shows P, Q, and R as different blocks. However, in practice, the triplets are passed through the same network one after the other). The modifications being, 1) absence of softmax activation at the end of the network 2) Dense 2 is altered to have 600 hidden units in place of 40 (or 10 depending on the dataset it was trained on).

4.3 Triplet sampling

The triplet sampling layer supplies the model with triplets such that one pair is similar to each other and the other pair is dissimilar. We consider two sequences to be similar if they are from the same raga and are dissimilar if they are from different ragas. The procedure for sampling triplets is as follows:

- From the collection of all subsequences, sample a subsequence P_{ref} randomly (uniform random).
- Let P_{ref} belong to raga R_i
- Select another phrase P_+ from the same raga R_i randomly (uniform random) such that $P_+ != P_{ref}$
- Randomly (uniform random) choose another Raga R_j such that $R_j != R_i$ from the set of all Ragas R
- Sample P_- from Raga R_j randomly (uniform random).

4.4 Model training and Optimization

Similar to the training of SRGM1, we employ dropout on the dense layers of the model. Since we are fine-tuning SRGM1, we do not have to train the word embeddings and the LSTM layers. Triplet sampling can be offline (before training) or online (during training). Although it reduces training time, offline sampling is extremely inefficient as it requires a lot of memory (RAM). Hence we sample triplets as and when required. The sequence ranking model is optimized based on the triplet margin loss, given by:

$$\mathscr{L} = min(D(\mathscr{P}_+, \mathscr{P}_{ref}) - D(\mathscr{P}_-, \mathscr{P}_{ref}) + M, 0) \quad (5)$$

where P is the feature embedding obtained for any input P using the model. D(.) is a distance function (the most common choice is the Euclidean distance) and M is the margin (common choice 1). Similar to SRGM1, we use distributed asynchronous stochastic gradient descent for training the model with Adam optimizer.

5. DATASET, INFERENCE AND EVALUATION METHODOLOGY

5.1 Dataset

In this work, we use the Carnatic Music Dataset [11] made available by the Comp Music group. The Comp Music data set consists of high-quality recordings of live ICM concerts. This dataset comprises of 124 hours of 480 commercially available recordings that are stored in the mp3 format.

The dataset contains 12 full-length recordings per Raga and features 40 different Ragas (hence 480 recordings). These recordings feature numerous artists, a wide variety of compositions and a diverse set of Ragas. The ragas are diverse in terms of their melodic attributes (refer to Section 1). Therefore, we believe that training and evaluating the model on this dataset would allow us to assess how well the model would perform in near real-world scenarios.

5.2 Comparison with prior works

We compare our results with three prior works, [4, 11, 12] of which [12] is the current state of the art for the Raga recognition task on the CMD dataset and [11] is the current state of the art for Raga recognition on the 10 Raga subset of CMD dataset. Gulati et al. [12] provides a summary of

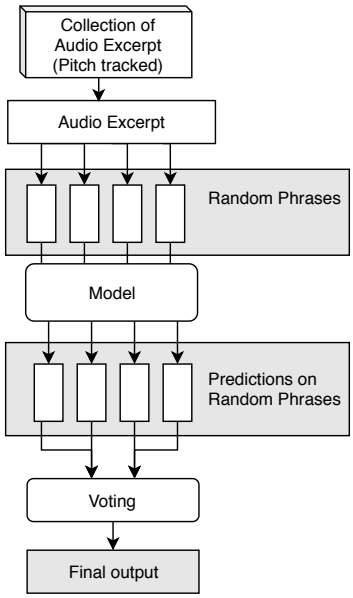


Figure 3. Figure gives an overview of the inference process for SRGM1 as described in Section 5.3.1

the performance of all of these the three methods on the CMD dataset. To ensure a fair comparison, we follow the evaluation strategy as provided in [12] (described in 3.3.1).

5.3 Inference

5.3.1 SRGM1

Although the model is trained on subsequences, during inference, we are interested in determining the Raga of the audio as a whole. Thus, follow the procedure as shown in figure 3. First, we split the audio excerpt into numerous subsequences and obtain the predictions for each of these. We then perform voting to determine the majority class. Note, that if the majority class has less than 60% of the votes we label the classification as incorrect.

5.3.2 SRGM2

For SRGM2, we are interested mostly in the sequences the model retrieves rather than inference on the audio as a whole. However, if one desires to make inference on the audio as a whole, we can adopt the procedure described in section 3.3.1.

5.4 Evaluation strategy

5.4.1 Evaluation strategy for SRGM1

CMD consists of 12 recordings per Raga. Since we train the model on subsequences, we have a total of 12*N_r subsequences in every class. Therefore, this is a balanced classification problem and accuracy is a suitable metric for evaluation. Authors in [12] use leave-one-out cross-validation strategy for evaluation where one of the 12 recordings for every Raga is used as test data and the remaining 11 are used as the training data. We adopt the same approach to evaluate our model as it enables us to make fair comparison of our results with previous works. We also present a confusion matrix for the observed results to further investigate the performance of SRGM1.

Gulati et al. [11] use stratified 12 fold cross-validation. To be able to compare our results to theirs on the 10 Raga subset of CMD, we adopt the same strategy. As an alternative evaluation strategy, we hold out 5 of the 12 recordings from every class as the test set and train the model on the rest.

Method	**CMD-10 Ragas**	**CMD-40 Ragas**
SRGM1	95.6%	84.6%
SRGM1 Ensemble	**97.1**%	**88.1**%
$\mathscr{M}_F$	-	81.5 %
$\mathscr{M}_{KL}$	-	86.7 %
$\mathscr{M}_B$	-	86.7 %
$\mathscr{E}_{VSM}$	91.7 %	68.1 %
$\mathscr{E}_{PCD}$	82.2 %	73.1 %

Table 1. Table summarizes performances of various models on the CMD dataset and its 10 Raga subset. Refer to Section 6.1

5.4.2 Evaluation strategy for SRGM2

A suitable evaluation metric for SRGM2, a sequence ranking model, is the "precision at top-k" measure which is popular score used to evaluate information retrieval systems. Precision at top k is defined as the proportion of retrieved items in the top-k set that are relevant to the query. We consider a retrieved subsequence to be relevant when it belongs to the same Raga as the query subsequence. The expression for precision at top-k hence becomes :

$$P_k = \frac{\sum_{i=1}^{k} I_{y_i,c}}{k} \tag{6}$$

$$P_{avg} = \frac{\sum_{i=1}^{n} P_{k_i}}{n} \tag{7}$$

Where P_k is precision at top k evaluated on one query sample, I_{y_i} is an indicator function which is 1 only when y_i which is the Raga of the i^{th} retrieved sample is same as c which is the Raga of the query sample. P_{avg} is the top-k precision averaged over the complete dataset.

6. RESULTS AND DISCUSSION

We present a summary of results in table 1 where we compare the performance of our model with [12], [11] and [4]. $\mathscr{M}_F$, $\mathscr{M}_{KL}$ and $\mathscr{M}_B$ represent 3 variations of TDMS [12] which uses euclidean distance, KL divergence and Bhattacharya distance respectively. $\mathscr{E}_{VSM}$ represents vector space model [11] for Raga recognition while $\mathscr{E}_{PCD}$ represents pitch class distribution based method presented in [4].

We observe that our method performs better than the current state-of-the-art on both 10 Raga subset of CMD and the CMD as a whole. We obtain an improvement of 6 % on the 10 Raga subset of CMD and improvement of

Figure 4. Figure shows the confusion matrix for the predictions obtained using SRGM1.

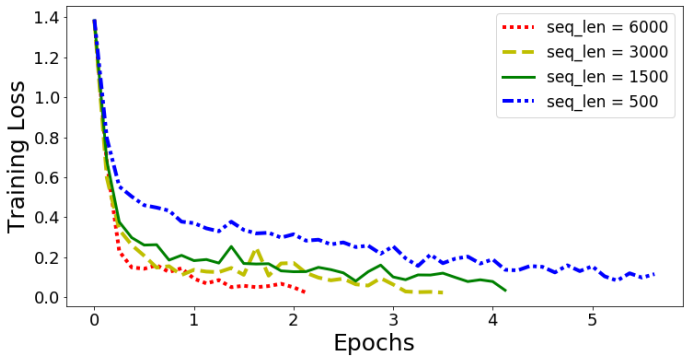


Figure 5. The above graph depicts variation in the "training loss vs epochs" plot with changing subsequence length.

Subsequence length	Num epochs to converge	Time per epoch in sec (Wall Time)	Hold out test set accuracy
500	12	32	88.86
1500	6	58.4	95.5
3000	5	120.1	95.63
6000	3	241.5	97.34

Table 2. Summary of our findings for the study on variation of model training and performance based on subsequence length

Metric	top-30 precision	top-10 precision
Score	81.83 %	81.68 %

Table 3. Summary of performance of sequence ranking on the top-10 and top-30 precision metrics

2% on CMD using the SRGM1-Ensemble model. The Ensemble model was created using 4 sets of weights for the same model architecture. For each test sample, we obtain output from each of the 4 models and then combine them by summing the log of the outputs of the model to obtain the final prediction vector for that sample.

The confusion matrix for SRGM1 trained on CMD (40 Ragas) is as shown in figure 4. Note that the confusion matrix is presented for the results obtained on a subsequence level and not on the audio excerpt level i.e, these are results before performing inference (refer to Section 4.3.1). There are no patterns apparent from the confusion matrix. However, on closely observing the samples on which the model performs poorly, we see that these samples have very little information, which can be attributed to events like to long pauses during the performance, sustaining a particular note for a prolonged interval, etc.

To further investigate the effect of subsequence length on the training of a model, we devise an experiment by using a 4 Raga subset of CMD. We use 36 recordings as the training set and 12 recordings as the test set. We train the network until the categorical cross entropy loss reduces to 0.02. Figure 5 shows a plot of the training loss vs the number of epochs. It is clearly visible that a model that is trained on subsequences of length 6000 converges in lesser number of epochs and has a smooth descent while the model using subsequences of length 500 makes the training very noisy and the network takes as long as 12 epochs to attain the same value of the loss. A summary of our findings has been tabulated in table 2.

We evaluate the sequence ranking model on the metrics described in section 5.4.2, namely top-10 precision and top-30 precision. Our findings have been summarized in table 3. SRGM2 obtains a top-30 precision of 81.83 % and top-10 precision of 81.68 %. We also conduct a qualitative analysis of the retrieved samples by inspecting various aspects like the range of svaras (notes) observed in the query to that in the retrieved sample, checking for similar note progressions, etc. We observe that the sequences retrieved by the model are similar to the ones in the query. We attach several samples of query subsequences and retrieved sequences as part of the supplementary material.

7. CONCLUSION AND FUTURE WORK

In this work, we proposed a novel method to address the problem of Raga recognition. Through various experiments and validation, we are able to demonstrate the capability of our approach in tackling the problem of raga recognition. We also introduce sequence ranking as a new sub-task of raga recognition. We present numerous query - retrieved subsequence pairs (as part of supplementary material) which demonstrates the effectiveness of this approach to mine databases for similar sequences.

We believe that deep learning based approaches have tremendous scope in MIR pertinent to ICM. As part of future work, we would like to further explore the idea of sequence ranking as we feel it can be used for tasks like joint recognition of tonic and Raga. An exciting future research direction would be to explore the possibility of using deep learning for generative modeling tasks in ICM. It would definitely be interesting to see if deep learning models can replicate the intricate improvisation aspects of ICM.

8. REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Laura-Lee Balkwill and William Forde Thompson. A cross-cultural investigation of the perception of emotion in music: Psychophysical and cultural cues. *Music perception: an interdisciplinary journal*, 17(1):43–64, 1999.

[3] Paul Boersma et al. Praat, a system for doing phonetics by computer. *Glot international*, 5, 2002.

[4] Parag Chordia and Sertan Şentürk. Joint recognition of raag and tonic in north indian music. *Computer Music Journal*, 37(3):82–98, 2013.

[5] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231, 2012.

[6] Pranay Dighe, Parul Agrawal, Harish Karnick, Siddartha Thota, and Bhiksha Raj. Scale independent raga identification using chromagram patterns and swara based features. In *International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 1–4. IEEE, 2013.

[7] Pranay Dighe, Harish Karnick, and Bhiksha Raj. Swara histogram based structural analysis and identification of indian classical ragas. In *International Society for Music Information Retrieval Conference*, pages 35–40, 2013.

[8] Konstantinos Drossos, Stylianos Ioannis Mimilakis, Dmitriy Serdyuk, Gerald Schuller, Tuomas Virtanen, and Yoshua Bengio. Mad twinnet: Masker-denoiser architecture with twin networks for monaural sound source separation. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[9] Shrey Dutta and Hema A Murthy. Raga verification in carnatic music using longest common segment set. In *International Society for Music Information Retrieval Conference*, volume 1, pages 605–611, 2015.

[10] Tejaswini Ganti. *Bollywood: A guidebook to popular Hindi cinema*. Routledge, 2013.

[11] Sankalp Gulati, Joan Serra, Vignesh Ishwar, Sertan Sentürk, and Xavier Serra. Phrase-based rāga recognition using vector space modeling. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 66–70. IEEE, 2016.

[12] Sankalp Gulati, Joan Serrà Julià, Kaustuv Kanti Ganguli, Sertan Sentürk, and Xavier Serra. Time-delayed melody surfaces for rāga recognition. In *International Society for Music Information Retrieval Conference*, 2016.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[15] Yannick Jadoul, Bill Thompson, and Bart De Boer. Introducing parselmouth: A python interface to praat. *Journal of Phonetics*, 71:1–15, 2018.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Gopala Krishna Koduri, Sankalp Gulati, Preeti Rao, and Xavier Serra. Rāga recognition based on pitch distribution methods. *Journal of New Music Research*, 41(4):337–350, 2012.

[18] Gopala Krishna Koduri, Vignesh Ishwar, Joan Serrà, and Xavier Serra. Intonation analysis of rāgas in carnatic music. *Journal of New Music Research*, 43(1):72–93, 2014.

[19] A Srinath Krishna, PV Rajkumar, KP Saishankar, and Mala John. Identification of carnatic raagas using hidden markov models. In *International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 107–110. IEEE, 2011.

[20] TM Krishna and Vignesh Ishwar. Carnatic music: Svara, gamaka, motif and raga identity. In *Proceedings of the 2nd CompMusic Workshop*. Universitat Pompeu Fabra, 2012.

[21] Sathwik Tejaswi Madhusdhan and Girish Chowdhary. Tonic independent raag classification in indian classical music. 2018.

[22] Christopher Olah. Understanding lstm networks, 2015.

[23] Surendra Shetty and KK Achary. Raga mining of indian music by extracting arohana-avarohana pattern. *International Journal of Recent Trends in Engineering*, 1(1):362, 2009.

[24] Rajeswari Sridhar and TV Geetha. Raga identification of carnatic music for music information retrieval. *International Journal of recent trends in Engineering*, 1(1):571, 2009.

[25] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[26] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013.

[27] Xinxi Wang and Ye Wang. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 627–636. ACM, 2014.

MODELING MUSIC MODALITY WITH A KEY-CLASS INVARIANT PITCH CHROMA CNN

Anders Elowsson
KTH Royal Institute of Technology
anderselowsson@gmail.com

Anders Friberg
KTH Royal Institute of Technology
afriberg@kth.se

ABSTRACT

This paper presents a convolutional neural network (CNN) that uses input from a polyphonic pitch estimation system to predict perceived minor/major modality in music audio. The pitch activation input is structured to allow the first CNN layer to compute two pitch chromas focused on different octaves. The following layers perform harmony analysis across chroma and time scales. Through max pooling across pitch, the CNN becomes invariant with regards to the key class (i.e., key disregarding mode) of the music. A multilayer perceptron combines the modality activation output with spectral features for the final prediction. The study uses a dataset of 203 excerpts rated by around 20 listeners each, a small challenging data size requiring a carefully designed parameter sharing. With an R^2 of about 0.71, the system clearly outperforms previous systems as well as individual human listeners. A final ablation study highlights the importance of using pitch activations processed across longer time scales, and using pooling to facilitate invariance with regards to the key class.

1. INTRODUCTION

1.1 Modality

Minor and major modality is a function of scale, harmony and tonality and is perceptible even to very young children [20]. However, the rich variability of music harmony renders many compositions hard to classify into a minor or major mode. Researchers have therefore investigated modality as a continuous variable in listening tests, producing more or less uniformly distributed averages with high internal consistency. Such a continuous variable, ranging from minor to major, has interchangeably been referred to as *modality* [12-14, 28, 34], *mode* [1], *key mode* [33], *mode majorness* [33], and *majorness* [2, 28, 33]. We will mainly use the term "modality" or "minor/major modality".[1] This paper aims to improve on previous methodologies for predicting perceived modality, designing a CNN that is able to model associated intricacies of musical harmony.

In a listener study [14], rated modality had a significant correlation (0.3-0.6) with rated *speed*, *articulation*, *pitch* (low/high) and *timbre*/*brightness* – happy tunes in major mode are likely more often performed with a higher articulation (staccato). This means that a system can be designed to predict perceived modality simply by picking up aspects in the audio not directly associated with harmony. Music information retrieval (MIR) systems relying on such confounding factors of variation have been challenged by Sturm [37]. The CNN architecture proposed in this study tries to minimize these interactions by specifically targeting properties directly linked to modality, as expanded upon, e.g., in Section 2.4.

1.2 Previous Studies Predicting Modality

Two previous studies have attempted to predict perceived modality from music audio. The first study [14] used partial least squares regression applied to audio features from the MIR toolbox [28, 29]. Two models were tried, the first using dedicated modality features and the second also including other spectral features. They were evaluated on the same two datasets used in the present study (Section 5.1), reaching an R^2 of 0.43 (0.38) and 0.47 (0.53) respectively (results for the second model in parenthesis).

A second study [1] have instead used the Inception v3 architecture [38] applied to a mel-frequency spectrogram. Results on a dataset of 5000 15 seconds (s) excerpts with lower ground truth consistency was $R^2 = 0.23$ (based on the Pearson's correlation coefficient of 0.48 reported in an additional/supplemental paper [2]). The model was developed to handle numerous perceptual features and may not be ideal for modality; the pooling operations applied across mel-frequency obfuscates tonal interrelationships at ranges larger than the pooling kernels. Since the filters span the time dimension, the model may to some extent instead make predictions from other aspects of the audio that covaries with modality, as outlined in Section 1.1.

1.3 Pitch Chroma and Deep Layered Learning

Chroma features, imposing spectral energies across a wide frequency range onto the twelve pitch classes of a musical octave, have a long tradition in MIR [15, 17, 35]. Pitch chromas have also been derived both from MIDI data and estimated through the autocorrelation function in the past [39]. A problem with the chroma is that it often is affected by interferences and becomes noisy [23]. Researchers have used various techniques to mitigate these issues, including harmonic percussive source separation [40] and cepstral whitening [31]. A multilayer perceptron (MLP) has also been used, with chord annotations defining ground truth pitch classes [23]. In this paper, we instead use the output from a high accuracy/resolution polyphonic pitch tracking system [8]. The pitch transcription is reshaped and fed as input to a CNN so that several pitch chromas emphasizing

[1] Since "mode" and "modality" have a wider scope, perhaps "modalite" could be a useful nomenclature for future studies.

different octaves can be learned within its first layer. The full architecture thus uses intermediate targets to restructure the learning problem according to the inherent organization of music. Such a "deep layered learning" approach [6], learning intermediate equivariant music representations, has been used for various MIR problems in recent years [e.g., 5, 19, 26].

1.4 CNNs and Ensemble Learning

The CNN proposed in this study use filter kernels operating across pitch/pitch class, pitch octaves and time scales, applying the same processing to each time frame. Previous CNNs for harmony processing have instead used filter kernels operating across time and frequency, for tasks such as key estimation [22, 25] and chord recognition [18, 24].

A CNN trained two times with randomly initialized parameters will generally produce two different predictions to the same input data. This is something that makes neural networks useful for ensemble learning [16]. The average (ensemble) prediction from several models containing more or less decorrelated errors will be better than randomly choosing one of them [32, 36]. Ensemble learning is used in this paper, as specified in Section 4.

1.5 Overview of the Paper

This paper presents a pitch chroma CNN architecture for predicting perceived modality. Section 2 describes how the pitch activation input to the CNN is computed and structured. In Section 3, the network architecture and training procedure is outlined. Section 4 describes how several CNNs were combined into an ensemble, and how a global prediction was made using additional features in an ensemble of MLPs. The two datasets and the evaluation procedure is described in Section 5, and results presented in Section 6. Section 7 presents an ablation study, testing how the design of the model affect predictive performance, and Section 8 offers conclusions.

2. PREPARING THE INPUT REPRESENTATION

2.1 Defining a Start and End Time for Each Excerpt

A start and end time were first determined for each musical excerpt (ME) so that the CNN would not have to make any predictions for silent parts in the beginning and end. A magnitude log-frequency spectrogram with 60 bins per octave was computed as described in [8] (pre-filtering). Let x be a vector representing the frequency response in time frame i. The overall magnitude of that time frame, across all frequency bins, was then defined as

$$m_i = \sqrt{\overline{x^2}}, \tag{1}$$

using the elementwise square and arithmetic mean, and forming m as a vector across time. The signal level was defined as $L_i = 20 \log_{10} m_i$, and the resulting vector filtered with a Hann window of width 61 frames (0.35 s). The *average* signal level of the ME was instead defined as $L_a = 20 \log_{10} \overline{m}$. The first time frame with a signal level within 10 dB of L_a defined the start, and the last frame within 10 dB of L_a defined the end of the ME.

2.2 Pitchogram

The input to the CNN was extracted from the *Pitchogram* representation computed with an existing machine learning system [8]. That system uses two stages to compute the Pitchogram. First, a sparse filter kernel operates across a log-frequency spectrogram to compute activations corresponding to tentative fundamental frequencies (f_0s), upsampled through parabolic interpolation to a centitone resolution. These tentative f_0s are then analyzed in a deeper network and computed activations inserted at the corresponding pitch bin in the Pitchogram. The Pitchogram thus contains f_0 activations and has a pitch resolution of 1 cent/bin and a time resolution of 5.8 ms/frame.

2.3 Extracting Semitone-spaced Pitch Vectors

The Pitchogram was down-sampled to 1 bin/semitone before processing by the modality CNN. To do this, a Hann window of height 141 bins (cents) was first applied across pitch to smooth the pitch response. Then, to adjust MEs that deviate globally from standard tuning (specifying A4 to a frequency of 440 Hz), the Pitchogram of each ME was "tuned". This was especially important for some of the MEs from the film music datasets (Section 5.1), where the orchestral performances had different tunings. The tuning was achieved by locating the maximum activation in a vector v of length 100, where each element corresponds to the sum of pitchogram activations at a specific cent value (i.e., all bins 47 cents above standard tuning were summed as the 47th entry of the vector). Only one vector was computed for each ME (i.e., global tuning). The whole Pitchogram was then shifted $\pm$50 cent based on the index of the maximum element. Finally, semitone-spaced activations were extracted between MIDI pitch 26-96, resulting in a pitch vector for each frame of height 71.

2.4 Varying Time Scales

We assume that listeners use pitch information at varying time scales to form an overall impression of the modality of a piece of music. At the shortest time scale, concurrent tones can form harmonic relationships that sound more like a major chord or a minor chord. At slightly longer time scales, tones played in succession may together imply the mode of the chord. At even longer time scales, the combination of tones and their relative activation may resemble key profiles/tonal hierarchies [27] that are more or less indicative of a major or minor tone scale. We wanted to develop a model that was agnostic to various factors that may covary with modality, such as accentuation (see Section 1.1), in the hope that our model would then generalize better to other datasets lacking these covariations. This precluded many models tracking variations in pitch activations across time (e.g., recurrent neural networks). Therefore, in order to still account for tonal relationships evolving over time, the pitch response was smoothed across time with filters of varying width, producing pitch vectors responding at varying time scales. The smoothing was done with Hann windows of width

$$w = 10 \times 3^n + 1 \tag{2}$$

where *n* varied between 1-5. The shortest Hann window therefore had a width of 31 frames (0.18 s) and the longest a width of 2431 frames (14.1 s). Since the unfiltered pitch vector was also included, the processing was applied at six different time scales. The smoothed pitch vectors were finally stacked across width, as shown in the left pane of Figure 1. This enabled the system to combine them during processing with a filter of width and stride 6 (Section 3.1).

2.5 Octave Spaced Depth (Chroma)

The proposed CNN computes a pitch chroma within its first layer. To facilitate this, the pitch vector was divided into 5 overlapping sections spaced an octave apart, each covering 23 semitones. The sections were concatenated across depth, resulting in aligned pitch classes (facilitating chroma processing within the CNN with filters extending across depth). This depth dimension is illustrated for a single time frame in the right pane of Figure 1. The CNN input of each time frame was thus prepared. It can be understood as a $23 \times 6 \times 5$ three-dimensional tensor, where height (23) represents pitch class, width (6) represents different time scales and depth (5) represents pitch octaves.

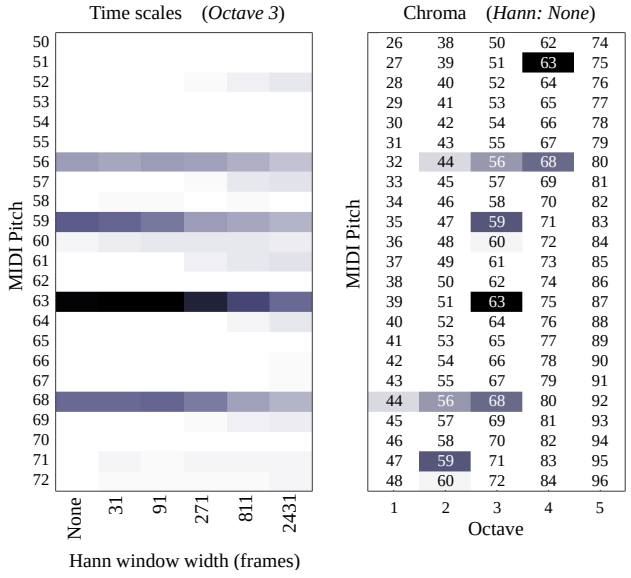


Figure 1. Two dimensions of the 3D-tensor input to the CNN for a single time frame, consisting of pitch activation from [8] restructured across pitch, time scale and pitch class. The time frame is taken 8 seconds into ME No 4 in the film clips dataset, where a G ♯ minor chord was played. The left pane consists of activations at varying time scales in the third octave. The right pane consists of the different pitch octaves, shown at the shortest time scale.

2.6 Segmentation

Each ME was divided into 6 overlapping segments of length 9 seconds (1550 time frames). Since no ME was longer than 54 seconds, these segments spanned the entire ME. Each segment of an ME was assigned the same ground truth annotation that had been established from listener ratings (Section 5.1). Since each frame (input tensor) had $23 \times 6 \times 5 = 690$ input values, each segment had slightly above a million input values (690×1550).

3. CNN ARCHITECTURE AND TRAINING

3.1 CNN Architecture

The CNN that was applied to each 9 seconds segment is shown in Figure 2. The same processing architecture was applied to each time frame of the segment. Convolutional filters always operated across unspanned dimensions, and zero-padding was never utilized, thereby shrinking the output space when applying the filters. Rectified linear units (ReLUs) were used as activation functions. Since the dataset in the study was small (203 MEs), it was important (and a challenge) to keep the number of learnable parameters small while retaining the ability to model the intricacies of musical harmony. The network had a total of 413 learnable parameters, including parameters for the batch-normalization that was applied after each ReLU layer.

As shown in the figure, the first *chroma layer* is used for converting input tensors to pitch chromas. Two filters learn weights for each octave, operating across pitch (height) and time scales (width). The two resulting pitch chromas are split into two branches, and processed with 1 and 5 filters respectively in the subsequent *harmony analysis* layer. This was done to reduce the total number of parameters while still achieving the following objectives:

- Allow the system to use two pitch chromas so that it could, for example, independently filter and account for information in the bass and higher pitches.
- Output 6 different activations for each time scale and pitch class from the *harmony analysis* layer using only $12 \times 6 + 6 = 78$ filter parameters.

Each filter used for harmony analysis spans an octave so that all pitch classes are taken into account when computing an activation (attempts at dividing the harmony analysis into two layers with shorter filters gave slightly lower results). Since the input has a height of 23, each filter is applied at 12 positions when operating across pitch, one for each pitch class. Thus, the input and filter sizes are the minimum sizes for which the same combination of all pitch classes at various keys can be subjected to the same filters. The two branches were concatenated across depth and the next filter (orange) then spanned the various *time scales*. The subsequent *max pooling* filter (purple) provides invariance with regards to the key class that the music was performed in, since it spans all 12 pitch/key classes. Based on the max pooling design, it can be expected that the previous layers of the CNN learn to either react to minor or major chords, minor or major tonal hierarchies, or various combinations thereof. The strongest indications of both major and minor modality across key are then passed to the subsequent *fully connected* (locally) layer (orange) for further processing. This layer is implemented as 7 filters that span the entire local input space (a fully connected layer would instead span the entire segment, something that is not desirable). A second filter (orange) combines the previous activations into a single *frame prediction*.

Finally, *average pooling* (purple) of the frame predictions is applied across the entire 9 seconds segment to produce a *segment prediction* (red) as a regression output.

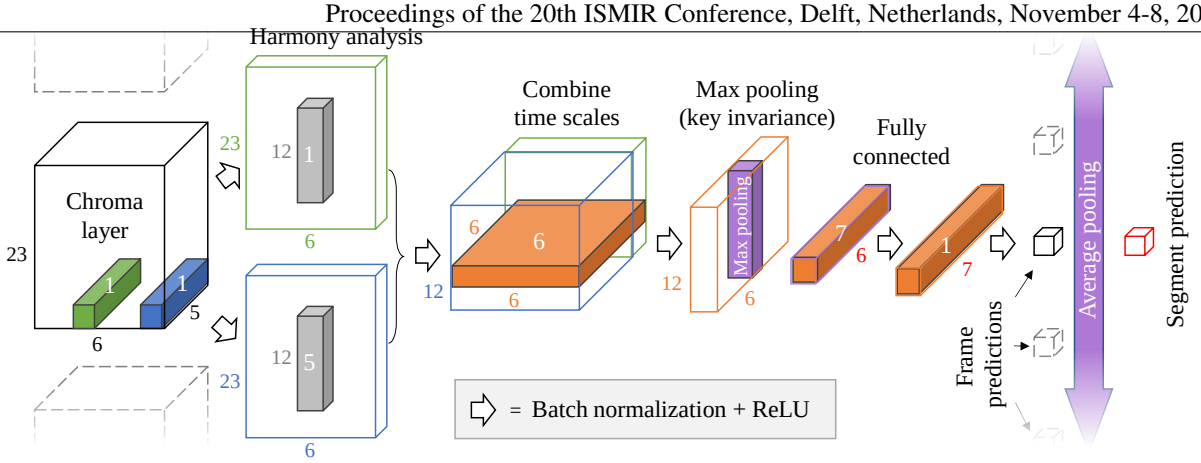


Figure 2. The CNN architecture for predicting perceived modality, shown for a single input tensor (time frame). Neighboring time frames are faded and dashed. All dimensions (except singleton) are indicated with black or colored numbers, and the number of filters is indicated with white numbers. White arrows indicate batch normalization followed by ReLUs. In the chroma layer, two filters (green and blue) compute pitch chromas from the input. The two chromas are processed to analyze the relationship between different pitch classes. Various time scales are then combined through six filters (orange), and max pooling applied (purple) to provide invariance with regards to the key class of the music. A fully connected layer is implemented through filters (orange) that span the entire feature space for a specific time frame. Finally, one filter is used to compute a frame prediction, and average pooling is applied to generate a prediction for the entire segment (red).

3.2 Training

The system was evaluated with 10-fold cross-validation, (Section 5.2). For each fold, the CNN was trained with the Adam optimizer [21], using the mean-squared-error loss function, an initial learning rate of 0.01, and a drop factor (every epoch) of 0.98. The gradient decay factor was set to 0.9 and the factor for L_2 regularization was set to 0.0001. A mini-batch size of 32 (segments) was used, shuffling the training data every epoch. The CNN was trained for 25 epochs. If the computed R^2 on the *training split* (the tracks used for training in each of the ten folds) was below 0.83 after 25 epochs, the network was reinitialized and training restarted (to avoid networks stuck in a local minimum). This happened in around 3 % of the cases. We tested the main model without the restarting condition in the ablation study (Section 7.1), which produced a minimal difference. Note that there was no validation stopping, for the same reasons as outlined in [10]: small validation sets are unreliable performance indicators, and maximizing performance for individual networks will not necessarily maximize performance of ensembled (Section 4) networks.

4. GLOBAL ESTIMATES AND ENSEMBLING

Two different global estimates were computed for each ME, one estimate from an ensemble of the CNN (ECNN), and one estimate from an ensemble of MLPs (EMLP) refining the output activations from each CNN.

4.1 CNN

The CNN modality prediction for each ME was computed as the average of all frame predictions (see Figure 2). This means that the local CNN architecture up until average pooling was applied to each frame at run-time.

We used ensemble learning to improve the accuracy of the CNN predictions (Section 1.4). Ten CNNs were trained for each fold, and the average of their predictions was used.

4.2 Additional MLP

In addition, another global estimate was computed for each ME with an EMLP, using the global CNN prediction and input features from the Pitchogram and spectrogram. The intention was to use and examine the effect of various (potentially confounding) features in the audio that the CNN was designed to not model.

Pitch activations in the tuned Pitchogram were averaged across time and summed to a vector, indexing into the vector based on each bins' distance to the closest semitone (0-50 cents). The 6 first discrete cosine transform (DCT III) components of the vector were then extracted as features. These features capture the extent and shape of micro-tuning deviations (PT) across the track (e.g., from vibrato).

We also computed both a vibrato suppressed (VS) and vibrato enhanced (VE) spectral flux (SF), using the max-filtering processing strategy first described in [9], developed to model perceived speed from onset densities in pitched instruments. The processing was applied to the whitened log-frequency signal level spectrogram computed as described in [8]. The VE SF was computed by subtracting the VS SF from the regular half-wave rectified bin-wise SF, thereby retaining energy only in the bins suppressed in the VS SF. For both versions, we computed the mean across time after half-wave rectification, producing an SF vector across frequency. The 6 first DCT components were extracted from these vectors as features. We then used the same log-frequency spectrogram, computed the mean across time, and extracted the first 6 DCT components of the resulting vector (spectral distribution, SD).

The average CNN output activation from all frames (Section 4.1) was also used as an input feature (naturally, this was the most important feature). In summary, the MLP had $1 + 6 \times 4 = 25$ input features, divided into four feature groups (PT, VS, VE and SD) and one CNN prediction.

The MLP was rather small and resembled the MLP developed for predicting performed dynamics in [10]. It had two hidden layers each consisting of 8 neurons. The network was trained for 5 epochs with the Levenberg-

Marquadt optimization [30]. Hyperbolic tangent (tanh) units were used as activation functions in all layers except for the last linear output activation. Each input feature was normalized by its minimum and maximum value to the range ± 1. Ensemble learning was used, taking the average prediction of 20 MLP models. Since the ECNN consisted of 10 CNNs, and since one EMLP was trained for each CNN, the final prediction in each fold was computed as an average of $10 \times 20 = 200$ MLP models.

5. DATASETS AND EVALUATION PROCEDURE

5.1 Datasets

The dataset for the study was assembled from two music audio datasets. The first dataset (D_1) consists of 100 audio excerpts of popular music (average length 30 s) that were produced from MIDI [14]. The second dataset (D_2) from [4] consists of 110 audio excerpts of film music (average length 15 s). As previously noted [7], the film music dataset contains duplicates. Seven duplicates were found and removed, reducing the size of D_2 down to 103 MEs. The MEs are polyphonic and use a wide range of instruments.

The overall modality had previously been rated by two groups of 19 and 21 listeners for the two datasets. Listeners were asked to rate the modality of each excerpt on a quasi-continuous scale between minor (1) and major (10), listening on high-quality loudspeakers. The ratings were averaged across listeners, producing a single ground truth rating of perceived modality for each ME. Reliability was relatively high, with a standardized Cronbach's alpha (CA) [3, 11] of 0.94 and 0.97 for the two datasets.

The datasets were pooled into a single dataset (203 MEs), which was used for training and testing.

5.2 Evaluation Procedure

The accuracy of the model was computed with the coefficient of determination, R^2, between predictions and ground truth annotations. We used the square of Pearson's correlation coefficient (including an intercept).

The models were evaluated with 10-fold cross-validation, using a stratified sampling so that each training set contained about the same number of MEs from D_1 and D_2. To improve the reliability of the results, the *complete experiment* was repeated ten times, re-partitioning the validation split each time.[2] To get 95 % confidence intervals (CIs), ten results (R^2s) were sampled with replacement and the mean computed, repeating the procedure 10^6 times. The resulting distribution of mean R^2s could then be used for extracting CIs; it indicates the reliability of the test results based on its variation over test runs.

6. RESULTS

6.1 Main Results

The final result for the ECNN and the ECNN in combination with the global EMLP is presented in Table 1.

[2]Running the main 10-fold cross-validation experiment ten times took about 5.5 days, training the ECNN with a GeForce GTX 1080 GPU (5 days; $10 \times 10 \times 10 = 1\,000$ CNNs) and the EMLP using 5 parallel i7-6700K CPU threads (0.5 days; $10 \times 20 \times 10 \times 10 = 20\,000$ MLPs).

Model	R^2	95 % CI	D_1	D_2
ECNN	0.672	0.665-0.679	0.645	0.710
ECNN+EMLP	0.716	0.710-0.722	0.710	0.745

Table 1. Squared correlation (R^2) between the ground truth ratings of perceived modality in music audio and the predictions of the two proposed models; also measured individually within the two datasets (D_1 and D_2).

The full system reached an R^2 of 0.716 for the predictions of perceived modality in the two datasets (corresponding to a correlation, r, of 0.846). The predictions from the CNN ensemble without the subsequent EMLP, minimizing contributions from confounding factors of variation, were almost as accurate, with an R^2 of 0.672.

As seen in Table 1, the second dataset (D_2) consisting of film clips was easier to predict than the first one (D_1). This difference is in line with results of the previous study on the same datasets that reported an R^2 of 0.43 (full model, 0.38) for D_1 and 0.47 (full model, 0.53) for D_2. The higher CA (Section 5.1) for this dataset indicates that listeners also had stronger agreement when rating it. Figure 3 shows predictions in relation to ground truth annotations.

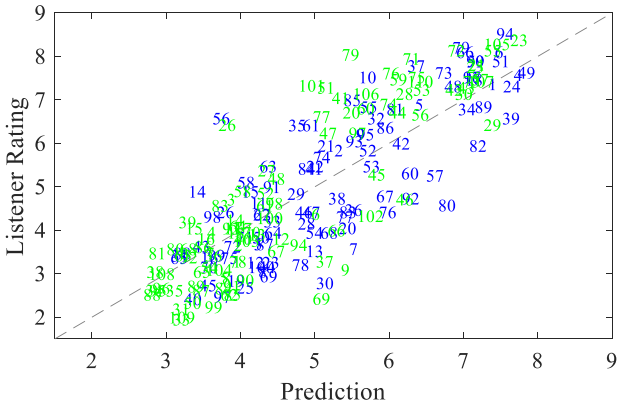


Figure 3. Predicted modality (x-axis) in relation to rated modality (y-axis) for datasets D_1 (blue) and D_2 (green) in one of the test runs ($R^2 = 0.706$) for the ECNN+EMLP. The dashed grey line indicates perfect prediction. Numbers indicate the index of each ME for future comparisons.

6.2 Comparison with Previous Systems and Humans

Figure 4 provides context regarding the prediction accuracy. The proposed system (blue bars) clearly outperforms previous systems (Section 1.2, white bars). Note that [1] was tested on a different dataset, so differences in predictive performance should be interpreted with caution. Human performance (circles) was computed from the listeners of the original listening test using a similar strategy as proposed in [10]. The performance of n listeners was derived by sampling (with replacement) n listeners and computing the R^2 between their mean rating and the mean rating from the non-sampled listeners. The procedure was repeated 10^5 times, using the 10^5 results to compute a mean and 95 % CIs. For $n = 1$, sampling is not applicable, and the 95 % CIs were defined as the listener with the second lowest and second highest R^2.

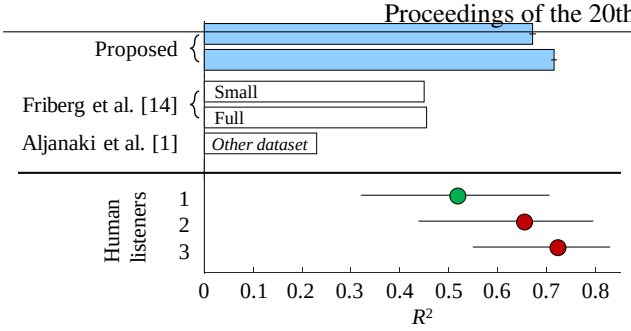


Figure 4. Modality estimation results (R^2) of the proposed system (blue), previous systems (white), individual human listeners (green circle), and ensembled human listeners (red circles). Black lines indicate 95 % CIs.

7. ABLATION STUDY

7.1 CNN

Important stages of the CNN architecture were examined by training the ECNN with various alterations of the CNNs. The different stages/properties tested were:

Time scales: *None-2431* – All different time scales (Section 2.4) were tested separately.

Key class invariance pooling (Pool): *Avg* – A CNN using average pooling instead of max pooling. *Conv* – A fully convolutional architecture that instead reduced the pitch dimension through four layers with two filters of height {5 4 3 3}, followed by a single filter of height 1.

Input: *Mag* – Using a magnitude log-frequency spectrogram as input, computed as described in [8] (pre-filtering). *dB* – Using the whitened log-frequency signal level spectrogram from [8]. For both versions, the spectrum covered the same range of 71 semitones, using overlapping triangular filters to reduce the frequency resolution.

Pitch chroma (PC): *Mean* – The mean of the five octaves was computed directly and passed as input, using 6 harmony analysis filters in the first layer

Results relative to the main CNN model are shown in Figure 5 and conclusions of the experiment provided in Section 8. The same validation split was used for all tests to increase consistency. This split was also used for the main model for computing a performance reference.

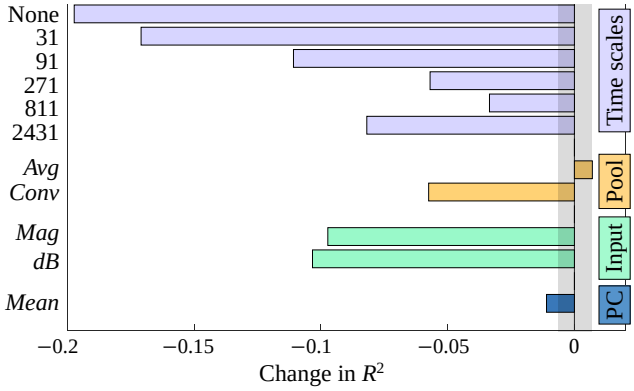


Figure 5. The variation in R^2 for different CNN settings in relation to the main CNN model. The 95 % CIs (±0.007) from the main experiment are indicated by the grey area.[3]

[3]Note that these CIs define the 95 % range within which the mean of *ten* complete 10-fold cross-validation runs varies. The ECNN ablation study used *one* complete 10-fold cross-validation run per architecture.

7.2 MLP

We also tested various combinations of EMLP input features.[4] The results shown in Figure 6 indicate that a combination of SF features with vibrato suppression (VS) and vibrato enhancement (VE) was important.

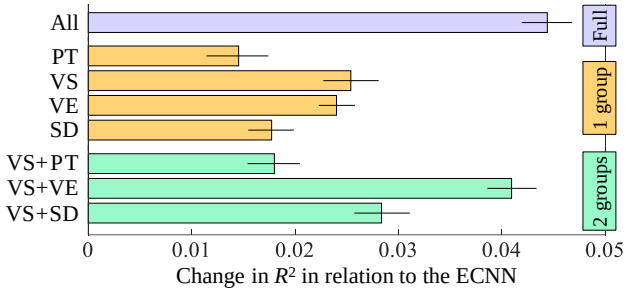


Figure 6. The change in R^2 in relation to the ECNN results, when using various EMLP feature groups (Section 4.2). Black lines indicate 95 % CIs for the relative improvement over the ECNN in each test run.

8. CONCLUSIONS

A convolutional neural network for predicting perceived modality in music was implemented. Its predictive performance was well above that of previous systems as well as the average human listener, performing better than around 95 % of the human annotators. It requires the combined ratings of 3 listeners to reach the same predictive performance as the model. The CNN used pitch activations from a pitch tracking system as input; the ablation study showed that this input representation improves performance substantially in relation to spectral input (*Mag* and *dB*). The methodology of max pooling across key classes to provide invariance seems beneficial since it improved performance in relation to a fully convolutional model. However, average pooling, in which the network instead has to rely on earlier ReLUs to discard irrelevant activations in certain key classes seems to be an equally attractive, or even better, option for achieving key class invariance.

The CNN was restricted from using filters operating across time, to reduce the influence of irrelevant confounding factors of variations, such as accentuation and spectral fluctuations. Instead, the CNN received input filtered to account for different time scales. The ablation study indicates that a time scale of around 4-5 seconds is the most relevant and that instantaneous time scales, only using harmonic information from concurrent tones, significantly reduces performance. Performance only dropped slightly when the pitch chroma layer of the CNN was discarded and the mean (across octaves) pitch chroma instead used as input (*Mean* PC, Figure 5). The small size of the dataset likely reduces the importance of this CNN layer; tracking interactions between pitches in different registers requires more learnable parameters, which requires more input data for generalization.

We hope that the results can inspire further development of CNN architectures accounting for musical invariances, including, and beyond, key class and pitch class.

[4]All feature groups were tested with the *same* global CNN activation as an additional feature, and evaluated across the full ten experimental runs.

9. REFERENCES

[1] A. Aljanaki and M. Soleymani, "A data-driven approach to mid-level perceptual musical feature modeling," in *ISMIR*, 2018, pp. 615-621.

[2] A. Aljanaki and G. Widmer, "Modeling Majorness as a Perceptual Property in Music from Listener Ratings," *arXiv preprint arXiv:1806.10570*, 2018.

[3] L. J. Cronbach, "Coefficient alpha and the internal structure of tests," *psychometrika*, vol. 16, no. 3, pp. 297-334, 1951.

[4] T. Eerola and J. K. Vuoskoski, "A comparison of the discrete and dimensional models of emotion in music," *Psychology of Music*, vol. 39, no. 1, pp. 18-49, 2011.

[5] A. Elowsson, "Beat tracking with a cepstroid invariant neural network," in *ISMIR*, 2016, pp. 351-357.

[6] A. Elowsson, "Deep Layered Learning in MIR," *arXiv preprint arXiv:1804.07297*, 2018.

[7] A. Elowsson, "Modeling Music: Studies of Music Transcription, Music Perception and Music Production," Doctoral Dissertation, KTH Royal Institute of Technology, 2018.

[8] A. Elowsson, "Polyphonic Pitch Tracking with Deep Layered Learning," *arXiv preprint arXiv:1804.02918*, 2018.

[9] A. Elowsson and A. Friberg, "Modelling Perception of Speed in Music Audio," in *Sound and Music Computing Conference (SMC)*, 2013, pp. 735-741.

[10] A. Elowsson and A. Friberg, "Predicting the perception of performed dynamics in music audio with ensemble learning," *Journal of the Acoustic Society of America (JASA)*, vol. 141, no. 3, pp. 2224-2242, 2017.

[11] C. F. Falk and V. Savalei, "The relationship between unstandardized and standardized alpha, true reliability, and the underlying measurement model," *Journal of personality assessment*, vol. 93, no. 5, pp. 445-453, 2011.

[12] A. Friberg, K. Choi, R. Schön, J. Downie, and A. Elowsson, "Cross-cultural aspects of perceptual features in K-pop: A pilot study comparing Chinese and Swedish listeners," in *43rd International Computer Music Conference (ICMC)*, 2017, pp. 291-296.

[13] A. Friberg and A. Hedblad, "A comparison of perceptual ratings and computed audio features," in *Proceedings of the 8th sound and music computing conference*, 2011, pp. 122-127.

[14] A. Friberg, E. Schoonderwaldt, A. Hedblad, M. Fabiani, and A. Elowsson, "Using listener-based perceptual features as intermediate representations in music information retrieval," *Journal of the Acoustic Society of America (JASA)*, vol. 136, no. 4, pp. 1951-1963, 2014.

[15] T. Fujishima, "Real-time chord recognition of musical sound: A system using common lisp music," *Proc. ICMC, Oct. 1999*, pp. 464-467, 1999.

[16] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 10, pp. 993-1001, 1990.

[17] C. Harte and M. Sandler, "Automatic chord identifcation using a quantised chromagram," in *Audio Engineering Society (AES) Convention 118*, 2005.

[18] E. J. Humphrey and J. P. Bello, "Rethinking automatic chord recognition with convolutional neural networks," in *Machine Learning and Applications (ICMLA)*, 2012, vol. 2, pp. 357-362.

[19] Y.-N. Hung and Y.-H. Yang, "Frame-level instrument recognition by timbre and pitch," in *ISMIR*, 2018, pp. 135-142.

[20] M. P. Kastner and R. G. Crowder, "Perception of the major/minor distinction: IV. Emotional connotations in young children," *Music Perception: An Interdisciplinary Journal*, vol. 8, no. 2, pp. 189-201, 1990.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] F. Korzeniowski and G. Widmer, "End-to-End Musical Key Estimation Using a Convolutional Neural Network," in *EUSIPCO*, pp. 966-970: IEEE.

[23] F. Korzeniowski and G. Widmer, "Feature learning for chord recognition: The deep chroma extractor," in *ISMIR*, 2016, pp. 37-43.

[24] F. Korzeniowski and G. Widmer, "A fully convolutional deep auditory model for musical chord recognition," in *MLSP*, 2016, pp. 1-6.

[25] F. Korzeniowski and G. Widmer, "Genre-Agnostic Key Classification With Convolutional Neural Networks," in *ISMIR*, 2018, pp. 264-270.

[26] F. Krebs, S. Böck, M. Dorfer, and G. Widmer, "Downbeat Tracking Using Beat Synchronous Features with Recurrent Neural Networks," in *ISMIR*, 2016, pp. 129-135.

[27] C. L. Krumhansl and F. C. Keil, "Acquisition of the hierarchy of tonal functions in music," *Memory & Cognition*, vol. 10, no. 3, pp. 243-251, 1982.

[28] O. Lartillot, "MIRtoolbox 1.5 User's Manual," ed, 2013, p. 215.

[29] O. Lartillot and P. Toiviainen, "A Matlab toolbox for musical feature extraction from audio," in *International conference on digital audio effects (DAFx)*, 2007, pp. 237-244.

[30] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431-441, 1963.

[31] M. Müller, S. Ewert, and S. Kreuzer, "Making chroma features more robust to timbre changes," in *Acoustics, Speech and Signal Processing (ICASSP)*, 2009, pp. 1877-1880.

[32] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and systems magazine*, vol. 6, no. 3, pp. 21-45, 2006.

[33] P. Saari, T. Eerola, and O. Lartillot, "Generalizability and simplicity as criteria in feature selection: Application to mood classification in music," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 6, pp. 1802-1812, 2011.

[34] R. Schön, "A cross-cultural listener-based study on perceptual features in K-pop," Master thesis, 2015.

[35] J. Serra, E. Gómez, P. Herrera, and X. Serra, "Chroma binary similarity and local alignment applied to cover song identification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 6, pp. 1138-1151, 2008.

[36] P. Sollich and A. Krogh, "Learning with ensembles: How overfitting can be useful," in *Advances in neural information processing systems*, 1996, pp. 190-196.

[37] B. L. Sturm, "A simple method to determine if a music information retrieval system is a “horse”," *IEEE Transactions on Multimedia*, vol. 16, no. 6, pp. 1636-1644, 2014.

[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818-2826.

[39] G. Tzanetakis, A. Ermolinskyi, and P. Cook, "Pitch histograms in audio and symbolic music information retrieval," *Journal of New Music Research*, vol. 32, no. 2, pp. 143-152, 2003.

[40] Y. Ueda, Y. Uchiyama, T. Nishimoto, N. Ono, and S. Sagayama, "HMM-based approach for automatic chord detection using refined acoustic features," in *Acoustics Speech and Signal Processing (ICASSP)*, 2010, pp. 5518-5521.

CONVOLUTIONAL COMPOSER CLASSIFICATION

Harsh Verma **John Thickstun**
Allen School of Computer Science and Engineering,
University of Washington, Seattle, WA, USA
harshv@cs.washington.edu, thickstn@cs.washington.edu

ABSTRACT

This paper investigates end-to-end learnable models for attributing composers to musical scores. We introduce several pooled, convolutional architectures for this task and draw connections between our approach and classical learning approaches based on global and n-gram features. We evaluate models on a corpus of 2,500 scores from the KernScores collection, authored by a variety of composers spanning the Renaissance era to the early 20th century. This corpus has substantial overlap with the corpora used in several previous, smaller studies; we compare our results on subsets of the corpus to these previous works.

1. INTRODUCTION

Models for attributing composers to musical scores have been extensively studied in the music information retrieval community. The composer classification question has been posed for a variety of corpora, from Renaissance composers [2,3], to the narrow (and challenging) case of Haydn and Mozart string quartets [5, 8, 12, 22], and to various collections of classical era composers (most of the other papers discussed in Section 2). In this work we study an expansive collection of scores, from 13th century sacred music by Guillaume Du Fay to 20th century ragtimes by Scott Joplin.

A major challenge of this task is learning from limited data. While the corpus considered here is larger than most, this is largely due to the number of composers considered (19): for specific composers, we have at most 466 scores (Bach) and as few as 22 (Japart). Small datasets are an inherent problem for composer classification: the corpus used in this work contains, for example, all of the Bach chorales and all of the Mozart string quartets. We cannot resurrect these composers and have them write us more scores to include in our corpus. This situation contrasts starkly with many learning problems, where substantial progress can be made by collecting massive datasets and exhaustively training an expressive model (usually a deep neural network) with "big data."

Further complicating this task, an individual score is itself a high dimensional object: the average score in our corpus consists of thousands of notes, each of which is encoded as a high dimensional vector to represent its pitch and value. Learning from a small number of examples in a high dimensional space is a formidable problem; thus much work on composer classification focuses on feature engineering, feature selection, dimensionality reduction, or some combination of these approaches to construct low-dimensional representations of scores to learn from.

In this paper we take a different approach: we dispense with feature engineering and explore end-to-end classifiers that operate directly on full scores. Specifically, we investigate shallow convolutional neural networks with an aggressive pooling operation. In this setting, all but the most impoverished linear classifiers achieve 100% training accuracy. We rely on implicit regularization introduced by the network structure and first-order optimization with early stopping to avoid overfitting to training data. While theoretical understanding of such an approach is in its early stages [18], we find empirically that this works quite well for composer classification.

2. RELATED WORK

The earliest works on composer classification [3, 15] analyzed highly preprocessed corpora of melodic fragments. Much of the subsequent work on classification focuses on engineering features to summarize full scores. These approaches can be broadly categorized, using the terminology of [7], into "global" summarization approaches that compute small sets of summary statistics as a feature set for each score [2, 5, 6, 10, 12, 16, 22] and local "event" featurizations that extract n-gram counts of a score as features [8, 9, 11, 24, 25]. There is also a line of work that applies compression-based dissimilarity metrics [1, 19, 20] to this task, which offers a substantially different perspective on classification problems.

The present work is most similar in spirit to [3] and [23]. Like [3], we adopt an end-to-end approach to feature learning using neural architectures. In contrast with [3], we learn on full scores with minimal preprocessing and consider a multi-class classification task over a broad variety of composers; this approach is made possible by modern hardware unavailable to researchers in 2002. We also take a more systematic approach to architecture exploration, and identify effective architectures that are simpler than

hybrid convolutional-recurrent approach taken in [3].

Like [23], we exploit structure in musical scores using convolutional models. But where [23] use a fixed Morlet or Gaussian convolution filters, the convolutional filters in this work are parameterized and learned from the data to maximize classification accuracy. We also explore multi-layer "deep" convolutional models and demonstrate improvements using such architectures versus the single layer of convolutions explored in [23].

Comparing to the substantial body of work that emphasize feature-engineering, the present work can be seen as an unified framework for learning global and event features. We will draw analogies between linear convolutional filters and n-gram features, and also demonstrate how convolutional models can express many popular global features. We will also introduce a global pooling operation that can be interpreted as an counter that tracks the number of occurrences of learned features, which is directly analogous to the count and ratio statistics that comprise the bulk of metrics used in human-engineered featurizations.

3. CORPUS AND DATA REPRESENTATION

We train and evaluate models on a corpus of 2,500 scores spanning five centuries of choral, piano, and chamber compositions from the KernScores collection [17]. An overview of this collection is provided in Table 1. In this work, we consider each movement of a multi-movement composition to be a distinct score. Our models extract only the note data (pitch, note-value, and voicing) from scores, ignoring all other markings such as time signatures, key signatures, tempo markings, instrumentation, and movement names. For the Renaissance composers in this collection (Du Fay through Japart) we shorten the length of all note-values by a factor of 4 to crudely account for the shift in duration conventions between mensural and modern notation [4].

We represent a score by lossless encoding of its pitch, voice, and note-value contents, transcoded from a **kern file to a binary representation suitable for input to a neural network. Specifically, we encode a score S as a binary tensor $\mathbf{x} \in \mathcal{S} = \{0,1\}^{T \times P \times (N+D+1)}$ where T, P, N, D are defined as follows:

- T - The number of rows of pitch/note-value data in the score S.
- P - The maximum number of concurrent **kern columns (spines): 6 for this corpus.
- N - The range of note pitches: 78 for this corpus, ranging from C1 to F#7.
- D - The number of distinct note values (i.e. durations): 55 for this corpus.

For each $t \in \{0, \ldots, T-1\}$, $p \in \{0, \ldots, P-1\}$, $n \in \{0, \ldots, N-1\}$, and $d \in \{0, \ldots, D-1\}$ we set

$\mathbf{x}_{t,p,n} = 1$ iff pitch n occurs at time t in spine p,
$\mathbf{x}_{t,p,N+d} = 1$ iff note-value d occurs at time t in spine p,
$\mathbf{x}_{t,p,N+D} = 1$ iff pitch n continues at time t in spine p.

Composer	Dates	Sub-Collection	Scores
Du Fay	1397-1474	Choral	35
Ockeghem	1410-1497	Choral	98
Busnois	1430-1492	Choral	68
Martini	1440-1497	Choral	122
Compere	1445-1518	Choral	27
Josquin	1450-1521	Choral	423
de la Rue	1452-1518	Choral	178
Orto	1460-1529	Choral	43
Japart	1474-1507	Choral	22
Corelli	1653-1713	Trio Sonatas	188
Vivaldi	1678-1741	Concertos	33
Bach	1685-1750	Chorales	370
		Well-Tempered Clavier	96
D. Scarlatti	1685-1757	Keyboard Sonatas	59
Haydn	1732-1809	String Quartets	209
Mozart	1756-1791	Piano Sonatas	69
		String Quartets	82
Beethoven	1770-1827	Piano Sonatas	102
		String Quartets	67
Hummel	1778-1837	Preludes	24
Chopin	1810-1849	Preludes and Mazurkas	76
Joplin	1868-1917	Ragtimes	47

Table 1. Details of the KernScores collection used for training and evaluation in this paper.

For example, consider how we would encode line 28 of the **kern excerpt shown in Figure 1. This is the 5th row of pitch/note-value data in the score, so we will encode data from this line into $\mathbf{x}_5$. The periods "." in columns (i.e. spines) 0 and 1 indicate that a note or (in this case) a rest is continued from a previous line, so we set $\mathbf{x}_{5,0,N+D} = 1$ and $\mathbf{x}_{5,1,N+D} = 1$. Two notes are indicated in spine 2: 8a and 8f. The number "8" indicates an eighth-note value. Each unique note-value is assigned an (arbitrary) index; we assign index 15 to the eighth-note value, so we set $\mathbf{x}_{5,2,N+15} = 1$. The letters "a" and "f" indicate the pitches A3 and F4, which lie 33 and 41 semitones above C1 (the base of our note range) respectively. Therefore we set $\mathbf{x}_{5,2,33} = 1$ and $\mathbf{x}_{5,2,41} = 1$. Finally, spine 3 indicates an eighth-note F5 so we set $\mathbf{x}_{5,3,53} = 1$ and $\mathbf{x}_{5,3,N+15} = 1$.

The encoding defined about is an essentially verbatim transcoding of the **kern text data to a binary structured format. Converting from text to this structured format will allow us to write convolution operations along the time and pitch axes of the data tensor $\mathbf{x}$. Encoding pitches with binary indicators in N-dimensional vectors is consistent with piano roll representations [23] but departs from the example of [3], which encodes pitch as a single numerical magnitude. The binary pitch encoding is required to support convolutions along the pitch domain, which we will introduce later in models (8) and (9).

The binary note-value encoding also differs from the numerical magnitude encoding used in [3]. We will not introduce models that convolve over durations (there is no translation invariant structure to exploit) so the motivation above for representing pitches with indicators does not ap-

23:	2..r	2..r	2..r	2..r
24:	8r	8r	8r	8dd
25:	=1	=1	=1	=1
26:	1r	1r	8r	4dd
27:	.	.	8f# 8a	.
28:	**.**	**.**	**8a 8f#**	**8ff#**
29:	.	.	8a 8f#	16ee
30:	.	.	.	16dd

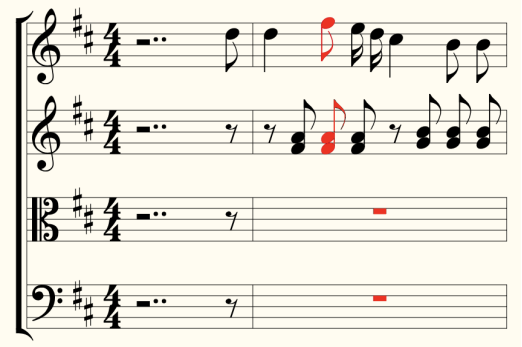


Figure 1. Left: An excerpt from a **kern encoding of of Haydn's Opus 33, No 1, consisting of the first 6 beats of the first movement. Right: A visual rendering of the first two measures of the same score as sheet music. In the **kern format, time proceeds from top to bottom, whereas in traditional notation time proceeds left to right. The contents of the beginning of the 6th beat is highlighted (red) in each format to aid comparison between the **kern format and sheet music.

ply to durations. Rather, we are motivated by the observation that note-values of similar duration may not be more alike in any musical sense than note-values with less similar duration. We avoid imposing this notion of similarity a-priori by encoding durations as categorical indicators: in this encoding, all note-values are equally distant in the Euclidean sense.

We also contrast our note-value encodings with piano roll representations, such as the representation used in [23]. In a piano roll representation, time is discretized: the value of a note is indicated implicitly by the number of discrete time-slices over which it is sustained. We choose an explicit representation of note-values because it more directly reflects the contents of a written score, and results in shorter time series overall than discretized representations.

4. PROBLEM FORMULATION

Our aim is to learn a classifier that predicts a composer y given a score $\mathbf{x}$. There are $C \equiv 19$ composers in our corpus: we assign each composer a label from 0 to $C-1$. We will construct a model $f_\theta : \mathcal{S} \to \{0,1\}^C$ that assigns vector $f_\theta(\mathbf{x})$ to a score $\mathbf{x}$ where each component $f_\theta(\mathbf{x})_i$ indicates model's (un-normalized) confidence that composer i wrote score $\mathbf{x}$. We predict $\hat{y}_\theta(\mathbf{x}) \equiv \arg\max_i f_\theta(\mathbf{x})_i$, the composer the model has most confidence in.

We evaluate our models via accuracy on holdout sets $\mathbf{x}^{\text{holdout}}$, where accuracy is the zero-one loss defined by

$$\text{Accuracy}(\mathbf{x}^{\text{holdout}}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(\hat{y}_\theta(\mathbf{x}_i^{\text{holdout}}) = y_i).$$

Here $\mathbf{1} : \text{Bool} \to \{0,1\}$ is the indicator function: $\mathbf{1}(p) = 1$ if the proposition p is true, otherwise $\mathbf{1}(p) = 0$. The results in Section 6 report 10-fold cross validated accuracies. It is standard practice in the machine learning community to report results on a single holdout set. But for for the small datasets considered in composer classification, cross-validating is essential to cut down the variance of estimated accuracy.

Given a collection of labeled scores (training data) $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ and a parameterized family of models $\{f_\theta : \theta \in \Theta\}$ we learn an optimal model f_θ by empirical risk minimization of the negative log-likelihood under the softmax-normalized probability distribution of model outputs:

$$\min_{\theta\in\Theta} \sum_{i=1}^{n} -\log\left(\frac{\exp(f_\theta(\mathbf{x}_i)_{y_i})}{\sum_{k=1}^{C}\exp(f_\theta(\mathbf{x}_i)_{y_k})}\right). \qquad (1)$$

For each of iteration of 10-fold cross-validation, in addition to the holdout fold $\mathbf{x}^{\text{holdout}}$, we hold out a second fold as validation data and optimize the objective (1) on the remaining 8 folds. We train our models using the Adam optimizer [13], regularizing with retrospective early stopping at the point with best accuracy on the validation fold.

5. MODELS

Every model class f_θ that we consider in this paper takes the following general approach.

1. Compute a set of local features at or around each time index in the score.
2. Average these features across time ("pool" the features, in neural networks parlance) into a single global feature vector.
3. Construct a linear classifier on top of this global feature vector to predict the composer of the score.

The approach above is motivated by the need to manage the high-dimensionality of a score: given even the first 5 indices of the score tensor $\mathbf{x}$ described in Section 3, we can easily fit a classifier that achieves 100% training accuracy but fails to generalize to new data. As discussed in Section 2, the classical approach to this overfitting phenomenon is to reduce a score to a low-dimensional summary of pre-determined features and fit a classifier to this summary. The present work aims to learn features from scratch, but if we permit our model to learn any features it wants then it will simply overfit to the training data.

We therefore cripple our models in two ways. First, step 1 of the general approach above limits our model to learn features that are local in scope. We will allow our

models to learn features that are sensitive to correlations between co-occurring notes (harmony), or between short sequences of notes (melody, rhythm). But by construction, our models will not be able to learn features that capture correlations between (for example) the first and last notes of a score. This precludes us from learning certain high-level patterns that could have predictive power (e.g. Mozart is more likely to repeat a section verbatim than Bach) but saves of us from learning a multitude of spurious patterns that appear to have predictive power on the training data but fail to generalize to new observations.

Second, step 2 of the general approach prevents our model from learning features that occur at fixed time locations. As discussed above, even knowing the first 5 indices of the score tensor is enough to easily identify every score in the corpus. By pooling features together across time, we force our models to classify based on the overall prevalence of the features it learns, rather than the occurrence of a particular feature at a particular time.

Note that classical approaches to feature engineering largely follow the same modeling restrictions outlined above. The engineered features used in e.g. [6] (Table 1, page 7) or [2] (Table 1, page 2) consist primarily of overall frequencies, prevalences, and rates of occurrence. These features capture properties of either a single time index or short sequences, aggregated across an entire score. The use of n-gram features also fits this mold: an n-gram is by definition a local feature of n time indices, and an n-gram featurization computes aggregate (i.e. pooled) counts of the occurrences of each particular n-gram across a score. The use of genuinely non-local features is rare. They are used in [12]: see for example the "maximum fraction of overlap with opening material within first half of movement" feature. The use of these features may account for the effectiveness of [12] in the Haydn versus Mozart classification task, which our models underperform on.

5.1 Sub-sampling Scores

The approach outlined above requires us to average features across entire scores. Each score in our corpus has a unique length, ranging from 10 to 4000 time indices. As a practical matter, it is difficult to deal with such variable-length data in machine-learning systems; our tools are designed to operate efficiently on homogeneous batches of data. One solution to this problem is to sub-sample scores; for example, [23] train models on the first s quarter notes where $s = 70$ or $s = 400$. Those authors found that the larger sample consistently outperforms the smaller one. We confirm this finding with the experiments in Table 2, which show that our models consistently perform better with larger samples of the score.

We therefore make the following compromise between using all available information from a score and operating on homogeneous inputs: we sample the first s, middle s, and last s indices from our score $\mathbf{x}$, resulting in $3s$ time indices sampled from each score. We use $s = 500$ for all experiments except the experiments in Table 2 that explore how models behave as we vary this hyperparameter. The average score in our corpus has 534 time indices, so for most scores this means we sample the entire score (for scores shorter than 500 time indices we pad out our sample with zeros). Only for scores longer than $1,500$ time indices (there are 117 in our corpus) do we lose any information with this approach.

	Sample Size					
Model	10	20	50	100	250	500
Histogram (Eqn 2)	50.0	59.0	62.0	63.0	66.1	64.2
Voices (Eqn 5)	60.0	61.6	63.9	72.0	75.5	76.9
Hybrid (Eqn 9)	59.3	62.1	68.9	77.1	79.9	81.7

Table 2. Comparison of model accuracies using a variety of samples sizes: accuracy uniformly increases with larger samples of the scores. See referenced equations (Eqn) for formal definitions of the models.

5.2 Histogram Models

The simplest models we consider are histogram models. Averaging the input data $\mathbf{x}$ over voices and time gives us a histogram vector $h \in \{0,1\}^{N+D+1}$:

$$h(\mathbf{x};\theta) = \frac{1}{TP}\sum_{t=1}^{T}\sum_{p=1}^{P}\mathbf{x}_{t,p}.$$

Multiplying this histogram by a weight matrix $W_\theta \in \mathbb{R}^{(N+D+1)\times C}$ with parameterized entries gives us a simple linear model:

$$f_\theta(\mathbf{x}) = W_\theta^\top h(\mathbf{x};\theta). \tag{2}$$

No features are learned in this model; all that is learned are the linear weights W_θ on the histogram features. The model can be interpreted as a simplified version of the global feature models discussed in Section 2. In this case, the global features are the prevalences at which each of the $N + D + 1$ note and duration symbols occur in a score.

5.3 Voice Convolutional Models

Now let's consider a simple neural model inspired by n-gram features. Let k be a number of features we desire to learn and n be a number of time indices. Define the function relu : $\mathbb{R} \to \mathbb{R}$ by $t \mapsto t\mathbf{1}(t > 0)$. Given a weight matrix $W_\theta^1 \in \mathbb{R}^{n(N+D+1)\times k}$ we can construct a "convolutional" feature representation $h_{t,p} \in \mathbb{R}^{T\times P\times k}$ at each time index t in each voice p defined by

$$h_{t,p}(\mathbf{x};\theta) = \text{relu}\left((W_\theta^1)^\top \mathbf{x}_{t:t+n,p}\right). \tag{3}$$

We define $\mathbf{x}_{t:t+n}$ to be a slice of $\mathbf{x}$ from index t to index $t + n$ (non-inclusive); when $t + n > T$, we pad $\mathbf{x}$ with zeros. We then pool these features across voices and time to construct a single, global feature representation $h \in \mathbb{R}^k$, to which we can apply a linear classifier with weights $W_\theta \in \mathbb{R}^{k\times C}$:

$$\begin{aligned} h(\mathbf{x}_t;\theta) &= \frac{1}{TP}\sum_{t=1}^{T}\sum_{p=1}^{P} h_{t,p}(\mathbf{x};\theta), \\ f_\theta(\mathbf{x}) &= (W_\theta)^\top h(\mathbf{x};\theta). \end{aligned} \tag{4}$$

This is a non-linear model (because of the non-linear relu "activation") and we can view h as a learned feature representation of the score $\mathbf{x}$. The weights ("filters") W_θ^1 learn to extract k relevant patterns of length n from voices, analogous to–but more expressive and compact than–classical n-gram featurizations. In our experiments we set $k = 500$ and $n = 3$; the choice of n is consistent with the pervasive use of 3-grams features in prior work [8, 9, 12, 24].

5.4 Deeper Representations

A natural way to extend the convolutional feature extraction discussed in Section 5.3 is to compose multiple layers of convolutions. Given the feature representation $h_{t,p}$ given by Equation 3 and a parameterized weight tensor $W_\theta^2 \in \mathbb{R}^{nk_1 \times k_2}$, we can construct a second layer of features

$$h_{t,p}^2(\mathbf{x};\theta) = \text{relu}\left((W_\theta^2)^\top h_{t:t+n,p}(\mathbf{x};\theta)\right).$$

We can loosely interpret such a representation as building hierarchical features of features. In principle we can build arbitrarily deep stacks of features in this way; in our experiments, we were unable to realize significant gains using architectures with more than two convolutional layers.

Building a classifier over these features proceeds identically to the shallower models:

$$\begin{aligned} h_{\text{conv}}(\mathbf{x};\theta) &= \frac{1}{TP}\sum_{t=1}^{T}\sum_{p=1}^{P} h_{t,p}^2(\mathbf{x};\theta) \\ f_\theta(x) &= (W_\theta)^\top h_{\text{conv}}(\mathbf{x};\theta). \end{aligned} \tag{5}$$

For this model we set $n = 3$, $k = 300$, and $k_2 = 300$.

5.5 Full-Score Convolutional Models

The models considered in Sections 5.3 and 5.4 are largely monophonic: they extract features from individual voices (although they classify based on a pool of these features gathered from all the voices). Notably, those models have no ability to capture harmonic patterns in the interactions between voices. We now consider a model that can capture these interactions.

Let $W_\theta^1 \in \mathbb{R}^{nP(N+D+1)\times k}$, $W_\theta^2 \in \mathbb{R}^{nk\times k_2}$ and consider the model

$$\begin{aligned} h_t(\mathbf{x};\theta) &= \text{relu}\left((W_\theta^1)^\top \mathbf{x}_{t:t+n}\right) \in \mathbb{R}^{T\times k}, \\ h_t^2(\mathbf{x};\theta) &= \text{relu}\left((W_\theta^1)^\top h_{t:t+n}\right) \in \mathbb{R}^{T\times k2}, \\ h(\mathbf{x}_t;\theta) &= \frac{1}{T}\sum_{t=1}^{T} h_t^2(\mathbf{x};\theta), \\ f_\theta(\mathbf{x}) &= (W_\theta)^\top h(\mathbf{x}_t;\theta). \end{aligned} \tag{6}$$

We parameterize this model with $n = 3$, $k = 300$ and $k_2 = 300$. This model is strictly more expressive than the part-wise models (4) or (5), capable of capturing patterns that the part models can't. However, the underperformance of this model (6) relative to less expressive models (4) and (5) suggest that it is prone to capture spurious patterns, leading to overfitting (compare results in Table 3).

5.6 Harmonic Models

All the models considered so far treat pitch classes as categorical data. We recognize, for example, that C4 is distinct from E4 or G4, but not that C4 is 4 semi-tones below E4 and 7 semi-tones below G4. This section introduces a model that exploits this structural order of pitch-classes, by convolving along the pitch-axis of the input tensor.

For notational convenience, we decompose the input tensor $\mathbf{x} = \mathbf{f} \oplus \mathbf{d}$ into separate pitch components $\mathbf{f} \in \{0,1\}^{T\times P\times N}$ and note-value components $\mathbf{d} \in \{0,1\}^{T\times P\times(D+1)}$. Let $W_\theta^1 \in \mathbb{R}^{jP\times k}$ and convolve along the pitch-axis to construct a features $h_{t,n}(\mathbf{f};\theta) \in \mathbb{R}^{T\times N\times k}$:

$$h_{t,u}(\mathbf{f};\theta) = \text{relu}\left((W_\theta^1)^\top \mathbf{f}_{t,:,u:u+j}\right).$$

Here j is a hyper-parameter indicating the height of the convolution; analogous to the width-n hyperparameter in our time-domain convolutions for models (4), (5), and (6). Unlike the time domain, we find that setting a large value of j (in our models, $j = N/2$) is desirable; a similar phenomenon is observed for frequency-domain convolutions in [21].

We proceed to pool the features $h_{t,u}$ together across the pitch domain to construct $h_t \in \mathbb{R}^{T\times k}$:

$$h_t(\mathbf{f};\theta) = \frac{1}{N}\sum_{u=1}^{N} h_{t,u}(\mathbf{f},\theta). \tag{7}$$

The idea of this pooling is to construct a feature-set that is invariant to pitch translation. We are interested in learning features such as, for example, the occurrence of general major chords rather than the occurrence of a particular major chord such as the one rooted at A3. The pooling operation above precludes us from learning the latter type of feature.

We then construct a second layer of features to integrate the harmonic features h_t together with the note-value features $\mathbf{d}_t$. Using weights $W_\theta^2 \in \mathbb{R}^{k\times k_2}$ and $W_\theta^3 \in \mathbb{R}^{(D+1)\times k2}$ we build $h_t^2(\mathbf{x};\theta) \in \mathbb{R}^{T\times k_2}$. We then pool the representations h_t^2 across time and construct a linear classifier on the resulting representation:

$$\begin{aligned} h_t^2(\mathbf{x};\theta) &= \text{relu}\left((W_\theta^2)^\top h_t(\mathbf{f};\theta) + (W_\theta^3)^\top \mathbf{d}_t\right), \\ h_{\text{harmonic}}(\mathbf{x}_t;\theta) &= \frac{1}{T}\sum_{t=1}^{T} h_t^2(\mathbf{x};\theta), \\ f_\theta(\mathbf{x}) &= (W_\theta)^\top h_{\text{harmonic}}(\mathbf{x}_t;\theta). \end{aligned} \tag{8}$$

We parameterize this model with $k = 64$ and $k_2 = 500$.

5.7 Hybrid Models

Looking back at the models we've introduced, observe that the voice models (4) and (5) exploit temporal structure within voices, but pool away any harmonic patterns between voices. In contrast, the harmonic model (8) exploits harmony between voices but pools away any sequential patterns across time indices. The full-score convolutional model can capture both types of structure, but is prone to capture spurious patterns and overfit.

This motivates the introduction of our final, hybrid model that weakly combines temporal and harmonic models to increase predictive power without overfitting. The idea is to feed the input tensor separately through temporal and harmonic models to construct features representations h_{conv} (5) and h_{harmonic} (8) respectively. We combine these features in a final, linear layer using weights $W_\theta^c \in \mathbb{R}^{k_2 \times C}$ and $W_\theta^h \in \mathbb{R}^{k_2 \times C}$ to make a prediction:

$$f_\theta(\mathbf{x}) = (W_\theta^{\text{c}})^\top h_{\text{conv}}(\mathbf{x};\theta) + (W_\theta^{\text{h}})^\top h_{\text{harmonic}}(\mathbf{x};\theta). \quad (9)$$

Because temporal and harmonic information are only combined in the final linear layer, this model is unable to learn expressive relationships between these features, such as the classical XOR relationship [14]. As we see in Table 3, this combination increases accuracy over either the temporal or harmonic models on their own.

6. RESULTS AND CONCLUSIONS

The results of all models discussed in this paper, evaluated on the full corpus, are presented in Table 3. We sort the rows in this table by the number of scores for each composer; we observe a trend towards increasing accuracy when we have more data (with some outliers).

	Models					
Composer	(2)	(4)	(5)	(6)	(8)	(9)
Japart	0.0	13.6	13.6	9.1	18.2	13.6
Hummel	41.7	54.2	66.7	62.5	87.5	91.7
Compere	0.0	25.9	22.2	25.9	40.7	37.0
Vivaldi	30.3	94.4	91.6	54.5	45.5	54.5
Du Fay	45.7	82.9	74.3	71.4	80.0	74.3
Orto	0.0	18.6	37.2	25.6	46.5	48.8
Joplin	85.1	91.5	93.6	93.6	95.7	91.5
D. Scarlatti	44.1	59.3	62.7	78.0	79.7	72.9
Busnois	13.2	48.5	48.5	51.5	60.3	60.3
Chopin	55.3	54.2	64.5	72.4	76.3	68.4
Ockeghem	13.3	55.1	69.4	52.0	66.3	72.4
Martini	44.3	68.0	75.4	59.8	68.0	73.8
Mozart	34.8	56.3	61.6	63.6	70.2	67.5
Beethoven	72.2	82.2	83.4	78.7	84.0	89.3
de la Rue	27.5	57.9	71.3	63.5	73.6	79.2
Corelli	89.4	89.9	86.2	93.1	93.6	95.2
Haydn	85.6	75.6	71.3	79.9	82.3	83.7
Josquin	81.1	78.7	76.4	75.9	77.3	82.3
Bach	92.3	95.7	96.1	97.2	97.2	97.6
Overall	64.2	75.4	76.9	75.5	79.8	81.7

Table 3. Results of the 19-way classification problem on the full corpus for each model considered in this paper.

To compare with previous work, we train additional models on subsets of the corpus. We invite comparisons between the results in Table 4 and the results of [2], and between the results in Table 5 and the results of [6]. These comparisons are imperfect: neither [2] nor [6] report the precise scores used in their experiments. Nevertheless our corpus is derived from the same KernScores sources as [2] and [6], and contains a comparable number of scores to the counts reported in [6]. Therefore we believe our subsets are similar to the corpora used in these works and that comparison is meaningful. For future reference, the exact dataset used for the present work can be found online. [1]

	Bach	Orto	Fay	Ock.	Josq.	Rue
Bach	100.0	0.0	0.0	0.0	0.0	0.0
Orto	0.0	39.5	0.0	7.0	51.2	2.3
Du Fay	0.0	0.0	82.9	11.4	5.7	0.0
Ockegham	0.0	2.0	5.1	81.6	9.2	2.0
Josquin	0.7	1.4	1.2	3.3	84.4	9.0
de la Rue	1.1	0.0	0.0	0.6	25.8	72.5

	Bach	Orto	Fay	Ock.	Josq.	Rue
(9)	**100.0**	**39.5**	**82.9**	**81.6**	**84.4**	72.5
KNN [2]	94.5	38.9	42.9	70.0	60.6	80.6
SVM [2]	98.5	33.3	25.0	60.0	60.0	**87.1**

Table 4. (Top) Confusion matrix for the hybrid model (9), trained and evaluated on a 6-composer subset of the corpus. Compare to the results in Tables 3 and 4 (page 6) of [2]. (Bottom) Accuracy comparisons of our hybrid model to the KNN and SVM models from [2].

	Bach	Haydn	Beethoven
Bach	99.8	0.2	0.0
Haydn	3.4	93.3	3.3
Beethoven	3.0	10.6	86.4

	Bach	Haydn	Beethoven
(9)	**99.8**	**93.3**	**86.4**
SVM [6]	94.6	80.3	64.8

Table 5. (Top) Confusion matrix for the hybrid model (9), trained and evaluated on a 3-composer subset of the corpus. Compare to the results in Table 9 (page 18) of [6]. (Bottom) Accuracy comparisons of our hybrid model to the SVM model from [6].

For the popular Haydn versus Mozart string quartet classification task [5, 8, 12, 22], we were unsuccessful. The standard evaluation metric for this task is LOOCV, which we could not perform due to the computational expense of our models. With 10-fold cross validation, we observed exceedingly high variance upon repeat optimizations of the same model. However none of our optimizations exceeded 80%. Due to imbalance between Haydn and Mozart quartets (209 versus 82 scores) a classifier that simply predicts Haydn given any input achieves 71.8%.

Overall, we conclude that the convolutional models proposed in this paper perform quite well. We find this notable, given that success in neural modeling is often associated with much larger datasets. Furthermore, we do not believe that the potential of these methods has been exhausted; further investigation may yield even better convolutional architectures for composer classification.

[1] http://homes.cs.washington.edu/~thickstn/ismir2019classification/

7. ACKNOWLEDGEMENTS

This work was supported by NSF Grant DGE-1256082. We also thank NVIDIA for their donation of a GPU.

8. REFERENCES

[1] Yoko Anan, Kohei Hatano, Hideo Bannai, Masayuki Takeda, and Ken Satoh. Polyphonic music classification on symbolic data using dissimilarity functions. In *ISMIR*, pages 229–234, 2012.

[2] Andrew Brinkman, Daniel Shanahan, and Craig Sapp. Musical stylometry, machine learning and attribution studies: A semi-supervised approach to the works of josquin. In *Proc. of the Biennial Int. Conf. on Music Perception and Cognition*, pages 91–97, 2016.

[3] Giuseppe Buzzanca. A supervised learning approach to musical style recognition. In *Music and artificial intelligence. Additional proceedings of the second international conference, ICMAI*, volume 2002, page 167, 2002.

[4] Julie E Cumming. Motet & cantilena. *A Performer's Guide to Medieval Music*, 2000.

[5] William Herlands, Ricky Der, Yoel Greenberg, and Simon Levin. A machine learning approach to musically meaningful homogeneous style classification. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[6] Dorien Herremans, David Martens, and Kenneth Sörensen. Composer classification models for music-theory building. In *Computational Music Analysis*, pages 369–392. Springer, 2016.

[7] Ruben Hillewaere, Bernard Manderick, and Darrell Conklin. Global feature versus event models for folk song classification. In *ISMIR*, volume 2009, page 10th. Citeseer, 2009.

[8] Ruben Hillewaere, Bernard Manderick, and Darrell Conklin. String quartet classification with monophonic models. In *ISMIR*, pages 537–542, 2010.

[9] María Hontanilla, Carlos Pérez-Sancho, and Jose M Inesta. Modeling musical style with language models for composer recognition. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 740–748. Springer, 2013.

[10] Maximos A Kaliakatsos-Papakostas, Michael G Epitropakis, and Michael N Vrahatis. Musical composer identification through probabilistic and feedforward neural networks. In *European Conference on the Applications of Evolutionary Computation*, pages 411–420. Springer, 2010.

[11] Maximos A Kaliakatsos-Papakostas, Michael G Epitropakis, and Michael N Vrahatis. Weighted markov chain model for musical composer identification. In *European Conference on the Applications of Evolutionary Computation*, pages 334–343. Springer, 2011.

[12] Katherine C Kempfert and Samuel WK Wong. Where does haydn end and mozart begin? composer classification of string quartets. *arXiv preprint arXiv:1809.05075*, 2018.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Marvin Minsky and Seymour A Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 2017.

[15] Emanuele Pollastri and Giuliano Simoncelli. Classification of melodies by composer with hidden markov models. In *Proceedings First International Conference on WEB Delivering of Music. WEDELMUSIC 2001*, pages 88–95. IEEE, 2001.

[16] Pasha Sadeghian, Casey Wilson, Stephen Goeddel, and Aspen Olmsted. Classification of music by composer using fuzzy min-max neural networks. In *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 189–192. IEEE, 2017.

[17] Craig Stuart Sapp. Online database of scores in the humdrum file format. In *ISMIR*, pages 664–665, 2005.

[18] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[19] Ayaka Takamoto, Mayu Umemura, Mitsuo Yoshida, and Kyoji Umemura. Improving compression based dissimilarity measure for music score analysis. In *2016 International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA)*, pages 1–5. IEEE, 2016.

[20] Ayaka Takamoto, Mitsuo Yoshida, Kyoji Umemura, and Yuko Ichikawa. Feature selection for composer classification method using quantity of information. In *2018 10th International Conference on Knowledge and Smart Technology (KST)*, pages 30–33. IEEE, 2018.

[21] John Thickstun, Zaid Harchaoui, Dean P Foster, and Sham M Kakade. Invariances and data augmentation for supervised music transcription. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2241–2245. IEEE, 2018.

[22] Peter Van Kranenburg and Eric Backer. Musical style recognition: a quantitative approach. In *Handbook of Pattern Recognition and Computer Vision*, pages 583–600. World Scientific, 2005.

[23] Gissel Velarde, Tillman Weyde, Carlos Eduardo Cancino Chacón, David Meredith, and Maarten Grachten. Composer recognition based on 2d-filtered piano-rolls. In *ISMIR*, pages 115–121, 2016.

[24] Jacek Wołkowicz and Vlado Kešelj. Evaluation of n-gram-based classification approaches on classical music corpora. In *International Conference on Mathematics and Computation in Music*, pages 213–225. Springer, 2013.

[25] Jacek Wołkowicz, Zbigniew Kulka, and Vlado Kešelj. N-gram-based approach to composer recognition. *Archives of Acoustics*, 33(1):43–55, 2008.

A DIPLOMATIC EDITION OF IL LAURO SECCO: GROUND TRUTH FOR OMR OF WHITE MENSURAL NOTATION

Emilia Parada-Cabaleiro[1,2] **Anton Batliner**[1] **Björn Schuller**[1,3]
[1]ZD.B Chair of Embedded Intelligence for Health Care and Wellbeing, University of Augsburg, Germany
[2]Instituto Complutense de Ciencias Musicales (ICCMU), Universidad Complutense de Madrid, Spain
[3]GLAM – Group on Language, Audio & Music, Imperial College London, UK
eparada@iccmu.es

ABSTRACT

Early musical sources in white mensural notation—the most common notation in European printed music during the Renaissance—are nowadays preserved by libraries worldwide trough digitalisation. Still, the application of music information retrieval to this repertoire is restricted by the use of digitalisation techniques which produce an uncodified output. Optical Music Recognition (OMR) automatically generates a symbolic representation of image-based musical content, thus making this repertoire reachable from the computational point of view; yet, further improvements are often constricted by the limited ground truth available. We address this lacuna by presenting a symbolic representation in original notation of *Il Lauro Secco*, an anthology of Italian madrigals in white mensural notation. For musicological analytic purposes, we encoded the repertoire in **mens and MEI formats; for OMR ground truth, we automatically codified the repertoire in agnostic and semantic formats, via conversion from the **mens files.

1. INTRODUCTION

White and black mensural notations are both defined by the use of strictly measurable unambiguous characters, a codification system introduced for the first time around 1280 in the *Ars cantus mensurabilis* [15]. Yet, white notation, unlike its predecessor black notation, passed through minimal changes during its period of existence (aprox. 1450–1600), thus becoming a consolidated European notation system typical of Renaissance vocal polyphonic music [1]. Furthermore, the development of relatively standardised musical sources in white mensural notation was also encouraged by the advancement of new printing technologies [12], which led at that time to a prolific production of early music prints, many of them still available nowadays.

The high cultural and historical value of these musical sources led to libraries worldwide utilising digitalisation mechanisms to preserve them. [1] Considering the often low quality of such scanned sources, this created a new challenge for Optical Music Recognition (OMR) systems. Much effort has already been made on symbolically representing this repertoire by developing suitable storage and encoding formats [28, 31] as well as improving OMR performance [5, 19, 24, 34]. Still, OMR technology is not yet reliable enough to accurately extract the musical content of some sources [28], making manual encoding—which provides the ground truth needed to improve the systems—often necessary. Despite this, digital editions of early music in original notation, as that presented in the *Measuring Polyphony Project*, [2] are still an exception [9, 20, 29].

We present a diplomatic edition and OMR ground truth of the anthology *Il Lauro Secco*—originally printed in 1582 in white mensural notation, previously encoded in original notation in Lilypond format by [20], and also transcribed in modern notation in **kern and MEI formats by [21]. To encode the anthology in the original (white mensural) notation, we considered **mens [28] and MEI [31] formats, chosen as the most adequate for manual encoding and storage, respectively. We also encoded the repertoire in the so-called 'agnostic' and 'semantic' formats, chosen as adequate to provide OMR ground truth [6, 27]. These were automatically generated from the **mens files through the `mens2agnostic` and `mens2semantic` converters, which we present as a way to reduce human efforts. A total of 660 codified scores, 150 engraved images, 150 original prints, and the converters are freely available. [3]

The paper is laid out as follows: in Section 2, an overview of related work is given; Section 3 describes our methodology; Sections 4 and 5 discuss the encoding criteria; finally, in Sections 6 and 7, we deal with limitations of our work, conclusions, and future directions.

2. PREVIOUS WORK

Manuscripts and prints, since the only remaining source of Renaissance music, have great value for the conservation and understanding of this music and its historical context,

[1] *Gallica* (Bibliothèque Nationale de France) and *Early Music Online* (British Library) have digitised around 600 musical sources from the 16^{th} century which are freely accessible and downloadable online.

[2] `http://measuringpolyphony.org/`

[3] `https://github.com/SEILSdataset/SEILSdataset`

a process in which digital technology plays a fundamental role [13]. Despite symbolic codification of the musical content is essential for a systematic study of the considered repertoire, the lack of a common methodology across the available digital collections makes their comparison difficult and might even bias research outcomes [8]. Furthermore, although initiatives such as *Tasso in Music Project* [26], [4] *Gesualdo Online Project*, [5] *Marenzio Online Digital Edition* (MODE), [6] *Josquin Research Project*, [7] or *The Lost Voices Project* [8] have as a main goal the codification of early musical sources in machine-readable formats, those preserving the original notation during the symbolic representation process are still rare [9, 20, 29]. In addition, despite the limitations of editing music in book format [35], even recently published—carefully curated—diplomatic editions of early music present transcriptions in modern notation of the musical content [10], without providing any symbolically codified support.

Music XML, considered the best music notation interchange format [11], is commonly used to transfer codified music across different platforms; still, Music XML might present limitations when working with early music, e. g., in the codification of different notations. In this regard, the Music Encoding Initiative (MEI) has been established [30], not only covering the encoding of a wide range of notations but also providing a set of features especially suited to the digital edition of music such as the inclusion of critical comments [22]. However, conversion routines between user-friendly encoding formats which support early notation, as, e. g., Lilypond [18] [9] and MEI, are still missing. Indeed, only recently suitable formats for manual encoding and conversion into MEI, such as the Humdrum representation scheme **kern [17], have been adapted for mensural notation, i. e., the representation scheme **mens [28], which as well as MEI can be rendered with the dedicated music engraving library Verovio [23] and through the online platform Verovio Humdrum Viewer (VHV [33]).

OMR has been progressively improved, e. g., by processing inconsistently notated handwritten scores [16, 25] and low quality early printed sources [24, 34]. Still, for all the machine learning systems, an adequate ground truth is essential to properly set-up an OMR framework. The so-called 'annotated dataset' or ground truth is a version of the evaluated data in which every instance (musical character) is identified with the label expected to be predicted. From a musicological prospective, a digital diplomatic edition [14], i. e., a symbolically codified source that faithfully mirrors the original print, would be the most appropriate approach to encode OMR ground truth of early music.

3. METHODOLOGY

We present a digital diplomatic edition of the anthology encoded in two open community driven formats with the capability to codify white mensural notation: **mens (a Humdrum representation scheme which presents a user-friendly encoding syntax [28]), and MEI (which has a specific module to encode mensural notation [31] and is considered to be the most appropriate storing format [28]). For OMR applications, we encoded the anthology's 'ground truth' in agnostic and semantic formats, i. e., two symbolic representations originally presented for western modern notation [6] but considered also as being appropriate to codify early notated sources [27].

3.1 Data description

The *Il Lauro Secco* anthology is a collection of 30 madrigals—secular polyphonic a cappella compositions—for five voices, [10] which has been published in a set of separate 'partbooks', i. e., according to a printing format in which each vocal part is presented in a different book [12]. This means that for each of the 30 madrigals in the anthology, five 'parts', i. e., individual scores presented in the corresponding partbooks (we will use the Italian term *particella(s)* to refer to these) are presented; no 'choral score', i. e., a score in which all the voices are displayed over-imposed in the same sheet, is available. Since the anthology encompasses 150 *particellas* (30 madrigals x 5 voices), and we considered four encoding formats—**mens, MEI, agnostic, and semantic—a total of 600 symbolically codified *particellas* are presented.

For the diplomatic edition, the 150 *particellas* were encoded in **mens and MEI (i. e., a total of 300 files), which present a faithful representation of the original source. The MEI files were also engraved with Verovio [23] and saved as images in pdf format (i. e., 150 files). Additionally, one choral score with the five voices over-imposed, from the lower to the higher in the same sheet, was also encoded in **mens and MEI for each madrigal (i. e., a total of 60 files). Still, the choral scores are only presented to encourage / facilitate analysis and performance; thus, they should not be considered as part of the diplomatic edition: They do not respect the original's editorial choice (i. e., partbook printing), and, contradicting the original print, lyrics normalisation and musical corrections to preserve vertical alignment were applied to them.

For the OMR ground truth, the 150 *particellas* were encoded in agnostic and semantic format, yielding a total of 300 files. The agnostic encoding gives the sequential representation of the musical symbols by indicating their exact position within the staff but without providing any musical meaning [6]. Differently, the semantic representation gives a simplified version of the scores while still keeping the musical meaning, e. g., the indication FM (F major) for modern notation implies that a flat is considered in the key-signature [6]. To the best of our knowledge, lyrics have not been considered in any of these formats, neither for mensural [27] nor for modern [6] notation; thus, the *particellas* in agnostic and semantic formats do not contain lyrics. In the future, these might be required to perform Optical

[4] http://www.tassomusic.org/
[5] https://ricercar.gesualdo-online.cesr.univ-tours.fr/
[6] http://www.marenzio.org/
[7] http://josquin.stanford.edu/
[8] http://digitalduchemin.org/
[9] http://lilypond.org/

[10] Note that the polychoral madrigal composed by Luca Marenzio is not taken into account [20].

Character Recognition (OCR) and OMR together. Then, the codified lyrics and their musical alignment can be retrieved from the *particellas* in **mens and MEI.

3.2 Data conversion and manual encoding

Even though the *Il Lauro Secco* anthology has already been codified in original notation [20], since this work employed the encoding format Lilypond—mostly used as a 'final' encoding choice rather than an interchange format [27]—direct conversion to **mens or MEI was not possible. Considering this, as starting codified version of the anthology, we chose the modern notated transcription encoded in **kern format [21], which contains a choral score for each madrigal (i. e., 30 **kern files in total). We converted the 30 choral scores from **kern to **mens by compiling the filter `kern2mens` [11] implemented in the Verovio Humdrum Viewer [12] (VHV [33]). Conversion errors were corrected (cf. Section 3.3), and aspects exclusive for mensural notation—thus missing in the **kern files, as e. g., colorature or ligatures—were manually integrated in the **mens choral scores (cf. Section 4.1).

When the musical content of the 30 choral scores in **mens was edited according to the original sources—note, as already mentioned, that a few musical corrections were required in order to preserve vertical alignment across voices—these were split into the individual parts. The extraction of the voices in separated files was performed with the Humdrum Toolkit, [13] through the command `extractx` of the Humdrum extras, [14] by that generating five **mens files per madrigal with two spines (one for music and another for lyrics) each. After that, the lyrics of the 150 *particellas* in **mens were manually 'unnormalised', i. e., corrected according to the original source (cf. Section 4.2), and the few musical corrections in the choral scores to preserve vertical alignment were also made according to the original. Finally, the **mens *particellas* were automatically converted into MEI syntax through the Verovio command-line interface. [15]

For the encoding of the agnostic and the semantic files, two Python scripts to convert from **mens to both formats, `mens2agnostic` and `mens2semantic`, were created. Even though agnostic and semantic representations have already been considered to encode Spanish white mensural notation [27], the repertoire in this notation, unlike the music prints considered by us, is handwritten, thus presenting many differences with respect to the white mensural notation of printed sources [29]. Due to this, for the agnostic and semantic encoding, we followed specific criteria (cf. Sections 5.1 and 5.2), which were developed from those originally presented for modern western notation [6]. Since lyrics were not considered for these representations, before automatically converting the 150 **mens *particellas*, they were removed by extracting the spines with musical content as individual files.

[11] `http://doc.verovio.humdrum.org/filters/`
[12] `https://verovio.humdrum.org/`
[13] `http://www.humdrum.org/`
[14] `http://extras.humdrum.org/man/`
[15] `https://www.verovio.org/humdrum.xhtml`

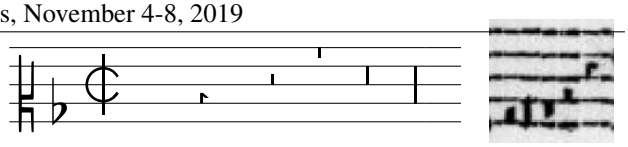


Figure 1. At left, a codified representation of mensural notation engraved in VHV presents (from left to right) the following symbols: C2 clef, B flat, 𝔠 time signature, and rests of *semiminima* in line 2 (L2), *minima* (L3), *semibrevis* (L5), *brevis* (L3), and *longa* (L3). At right, an extract of a original print taken from the *Basso* part of Belli's madrigal presents (from left to right) rests of: *minima* in line 1 (L1), *brevis* (L1), *semibrevis* (L2), *minima* (L2), and *semiminima* (L3). Rests position (standard in **mens) is given considering their proximity to the center of the staff (L3).

3.3 Data post-processing

Musical aspects specific of mensural notation, e. g., 'proportions' (transcribed in modern notation with triplets), were not interpreted in the conversion from **kern to **mens. Thus, manual post-processing of the **mens files was required to address the following conversion issues: (i) 16^{th} notes in **kern were converted in **mens as *semibrevis* instead of *semifusas*; (ii) triplets, used in **kern to transcribe *proportio tripla* of *tempus imperfectum* [1], i. e., the *diminution* of the *prolatio* (reduction of the subdivision values) by changing from a time signature with binary meter and binary subdivision (e. g., 𝐂 and 𝔠) to another with binary meter and ternary subdivision (e. g., `3` and $\mathbf{C}\frac{3}{2}$), were not recognised in the conversion, yielding compilation failure of the **mens files.

Although 'coloration' and 'custos' can be encoded in MEI, these are not yet implemented in VHV for **mens (cf. Section 4.1). Due to this, in the conversion from **mens to MEI, these were not recognised, but manually encoded in MEI. Similarly, since lyrics linked to rests are not supported in VHV (cf. Section 4.2), these were not recognised in the conversion from **mens to MEI, but also manually encoded. In **mens representation, unlike in the original source, rests are indicated in a standard position within the staff (cf. Figure 1); yet, in the agnostic representation the original position of the rests should be indicated. The same applies for accidentals, which in **mens, unlike in the original source (cf. Figure 3), are displayed in the same staff position of the note they refer to. Since in the conversion from **mens to agnostic, the specific position of rests and accidentals was missing, this was manually encoded in the agnostic files (cf. Section 5.1).

4. CRITERIA FOR DIGITAL DIPLOMATIC EDITIONS OF WHITE MENSURAL NOTATION

While MEI is a suitable encoding format for storage, **mens is more appropriate for manual encoding [28]. The following criteria will refer mainly to the codification in **mens; manual interventions in MEI will only be described when required. For white mensural notation encoding guidelines in **mens, cf. [28]; [16] for MEI, cf. [32]. [17]

[16] `http://doc.verovio.humdrum.org/humdrum/mens/`
[17] `http://www.verovio.org/features.xhtml?id=mensural`

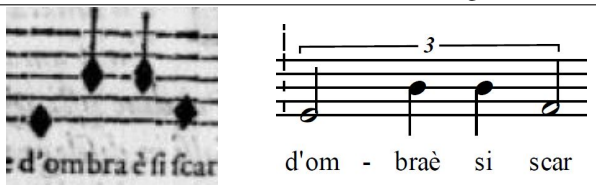


Figure 2. At left, the rhythmic sequence *semibrevis*, *minima*, *minima*, *semibrevis* (all blackened), which should be interpreted as *minor color*; at right, the transcription of this fragment of Belli's madrigal in modern notation [20].

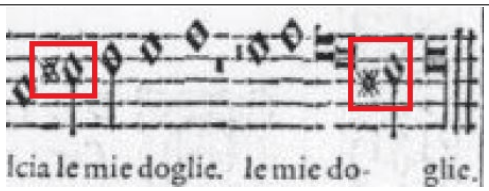


Figure 3. Two different representations of the accidental sharp displayed in Marenzio's madrigal. The first in the standard position, i. e., aligned to the note; the second not.

4.1 Musical encoding criteria

Unlike the anthology codified in Lilypond [20], where different musical aspects are modified w.r.t. the original source to encourage the analysis and performance of the repertoire, the present encoding in **mens and MEI tries to represent the original source as much as possible. For this, the following criteria were considered:

(i) In previous symbolic versions of the anthology [20, 21], rests have often been encoded in shorter length than in the original source. This was due to the use of barlines, which made it impossible to display rests longer than a measure. In **mens and MEI encoding, the exact duration of the rests indicated in the original source was considered; yet, as in these formats the position of rests within the staff cannot be defined, the default position displayed in the images engraved with VHV might not always coincide with that given in the original print (cf. Figure 1).

(ii) Stems up to the third staff line (included) are generally displayed upwards, while above the third space (included), they are displayed downwards. This applies always when engraving codified scores in VHV; yet, for melodic reasons, notes in the third line of the original source might also present downwards stems. Thus, the direction of the stems which did not follow the standard disposition in the original source was manually specified in the **mens encoding. Furthermore, since *Longa*'s stems in VHV are always engraved downwards, these were also specified, when needed, according to the original.

(iii) Coloration or 'blackening' [1]—used to indicate rhythms with ternary subdivision (i. e., *perfect prolatio*)—is an attribute unique of white mensural notation; thus, it was not indicated in the modern notated transcription in **kern [21]. Due to this, coloration was manually encoded in **mens according to [28] by indicating '∼'. Since coloration is not yet implemented in VHV, this is not displayed when engraving **mens files, and it was also not correctly converted from **mens to MEI; thus, coloration was manually encoded in MEI by specifying the note attribute 'colored="true"' [32]. As blackened *minimas* appear graphically as *semiminimas*, it depends on the musicological interpretation whether such a character should be encoded as the former or the latter. To prevent interpretation bias, coloration was only indicated for *brevis* and *semibrevis*, i. e., the notes which do not have an already existing equivalent when blackened. Still, from a musicological prospective, the so-called *minor color*—a *semibrevis* followed by a *minima* both blackened that must be performed as a triplet of whole and half notes [1]—should be considered; thus, in some cases, *semiminimas* might be interpreted as blackened *minimas* (cf. Figure 2).

(iv) *Ligatures*—symbols that in mensural notation represent a combination of two or more notes [1]—were indicated according to the VHV documentation, [18] i. e., notes within the ligature are delimited with brackets: square brackets were used for *recta ligatures*, angle brackets for *obliquous ligatures*. However, since the rhythmic stems of *ligatures* are not yet implemented in VHV, they are not displayed when engraving **mens and MEI encoding.

(v) *Custodes* (singular: custos)—a symbol at the end of a staff indicating the pitch of the first note in the next staff [1]—are not yet implemented for **mens encoding. Considering this, we introduced the indication `*custos` in the *particellas* in **mens, which although is not engraved, indicates the exact position where the custos is displayed, i. e., the end of each staff as shown in the original source. This might be similar to the indication `!!linebreak:original`, typically used in **kern encoding. Unlike in **mens, *custodes* are already implemented in MEI encoding [32]; thus, these were manually indicated by adding the event `<custos/>`, in which the exact position of the symbol is also indicated by the attributes pitch name (`pname`) and octave (`oct`).

(vi) Accidentals in mensural notation present differences w.r.t. modern notation; e. g., single sharps in mensural are indicated with 'x', while this symbol indicates double sharp in modern notation. Furthermore, unlike in modern notation, accidentals in mensural are not always aligned—displayed in the same vertical staff position—to the note they precede (cf. Figure 3). Still, since in **mens encoding accidentals' position cannot be specified, these are displayed, when engraved in VHV, according to the default alignment; [19] thus, their position in the engraved images may not coincide with that of the original source.

(vii) Measures were considered in the *particellas* to indicate staff breaks, i. e., after each `*custos`, a new measure was indicated to break the musical content according to the distribution displayed in the original source. [20]

(viii) Clefs, key, and time signatures were indicated at the beginning of the score (first staff) and within staves, [21] but not at the beginning of consecutive staves, since automatically engraved in VHV. In mensural notation, unlike

[18] `http://doc.verovio.humdrum.org/humdrum/mens/`

[19] Despite in MEI the specific position of the accidentals can be specified, this was not yet performed due to time-constraints.

[20] In the 'choral scores', in order to facilitate interpretation and analysis, the numbers of measures are given regularly to fragment the music.

[21] Since verovio and VHV are still in development, repetitions within a staff of the same clef and some time signatures might not be engraved.

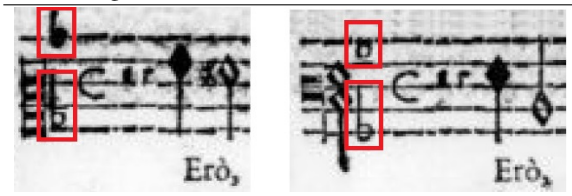


Figure 4. B–flat showed twice for C2 (left) and F3 (right) clefs, in the *Canto* and *Basso* parts of da L'Occa madrigal.

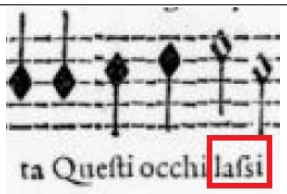


Figure 5. Two different representations of the letter 's' in the word *lassi*, displayed in Fiorino's madrigal.

in modern, when having a flat in key signature, this is displayed twice for C2 and F3 clefs (cf. Figure 4); yet, only one is engraved by VHV for **mens and MEI encoding.

4.2 Textual encoding criteria

The lyrics of the *particellas* encoded in **mens were 'unnormalised' w.r.t. the standardised transcription in **kern format [21], i. e., they were rewritten according to the original source. In this process, punctuation was introduced when missing, and modified or removed when needed. Contractions (e. g., *altrov'adopra*), abbreviations (e. g., *hãno*, *pche*, or *ij*), and the *tironian* symbol '&', previously transcribed as 'et', were indicated as in the original. The arbitrary use of diacritic marks (e. g., *più* and *piu*), letters (e. g., *verde* and *uerde*), links between words (e. g., *invano* and *in vano*), and letter capitalisation was kept inconsistent across voices as in the original partbooks.

Unlike in the transcription encoded in **kern [21], lyrics linked to long rests at the beginning of a voice were encoded in the *particellas* in **mens and MEI; still, since these are not supported in VHV, they are not displayed in the engraved images. Indeed, even though lyrics might be printed under rest in the original source, these should not be sung—they are given for performance reasons, to indicate verse sung by the other parts to voices which start with long rests. Finally, due to graphical limitations, the two different spellings of the letter 's' (cf. Figure 5) were indicated with the unique graphical symbol 's'.

5. ENCODING CRITERIA FOR OMR GROUND TRUTH OF WHITE MENSURAL NOTATION

The vocabularies for agnostic and semantic encoding of modern notation [6] were adapted to the characteristics of white mensural notation. Thus, aspects typical of modern notation, such as slurs or dynamics, were not considered, while elements characteristic of mensural notation, e. g., 'ligatures' and 'coloration', were taken into account. Division and addition dots were indicated as shown by [6] for modern notation. Note that in white mensural notation, the following notes are used [1]: *maxima*, *longa*, *brevis*, *semibrevis*, *minima*, *semiminima*, *fusa*, and *semifusa*.

5.1 Criteria for agnostic encoding

Since in the agnostic format, musical symbols are encoded as graphical objects without musical meaning, for each element, its position within the staff (line or space) is indicated. Lines and spaces are enumerated from the bottom to the top: the five lines from L1 to L5, the additional line below the staff L0, the one above L6; the four spaces from S1 to S4, the space below the staff S0, the one above S5.

(i) The position of rests within the staff is not standardised in mensural notation (cf. Figure 1); thus, all the possible positions were considered in the vocabulary. Still, since in the conversion from **mens to agnostic, such a position was not indicated (cf. Sections 3.3 and 4.1–i), this was manually encoded in the agnostic files.

(ii) The stems whose direction in the original source did not follow the standard rule (cf. Section 4.1–ii), were marked with back-slash '\' or slash '/', to indicate downwards and upwards directions, respectively, e. g., 'note.semifusa\-L3'. Note that this does not apply to *brevis* and *semibrevis*, i. e., notes without stem.

(iii) Coloration (cf. Section 4.1–iii) was indicated for *brevis*, *semibrevis*, and ligatures by adding '∼' when applicable, e. g., 'note.breve∼-S3' or 'ligature∼.start-L5'.

(iv) Ligatures (cf. Section 4.1–iv) were indicated by the word 'ligature' instead of 'note'. Since the considered anthology presents only *recta* ligatures of two notes, in the vocabulary, only the starting and final notes of the ligature, without distinction between *recta* and *obliqua*, were indicated, e. g., 'ligature.start-L2'. Still, middle notes, ligature type, and other attributes could also be defined if needed.

(v) *Custodes* (cf. Section 4.1–v) were indicated by the word 'custos' followed by their position, e. g., 'custos-S3'.

(vi) The position of accidentals is not always standardised in mensural notation (cf. Figure 3); still, since in **mens encoding this cannot be specified (cf. Section 4.1–vi), in the conversion from **mens to agnostic, the accidentals' default position was indicated and manually modified in the agnostic files when required (cf. Sections 3.3).

(vii) Clefs, key, and time signatures were encoded as in the original source, i. e., also at the beginning of each staff—note that the agnostic encoding might be split in staves by introducing a break-line after each custos, if needed. B–flat in key signature, displayed twice for some clefs, was also codified, e. g., C2 clef with a B–flat in key signature (cf. Figure 4) would be encoded in agnostic as 'clef.C-L2 accidental.flat-L5 accidental.flat-S1'.

5.2 Criteria for semantic encoding

In the semantic encoding, each element is intended with its musical meaning; thus, no position markers such as staff line or stem direction are given, since implicitly indicated.

(i) Renaissance music can be grouped into two systems, *durus* and *mollis*, which together with the cleffing, i. e., the use of standard (up to C1) or high (up to G2) clefs, were the common criteria used by publishers in the 16^{th} cen-

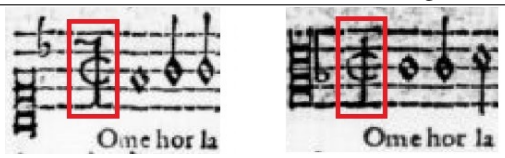


Figure 6. Representations of the time signature 𝔠 showed in the *Canto* (left) and *Alto* (right) of Fronti's madrigal.

tury to group madrigals in different collections [7]. *Durum*, in Latin, corresponds to B–natural in modern terminology while *molle* corresponds to B–flat [12]; thus, madrigals in *durus* system (i. e., with B–*durum* in key signature) would use the Lydian scale while those in *mollis* (i. e., with B–*molle* in key signature) would use the F major scale [1]. In the semantic encoding, the words 'mollis' and 'durus' were considered to indicate B–flat and no alteration in key signature, e. g., 'keySignature-durus'.

(ii) Accidentals in mensural notation must be interpreted according to the musical context, e. g., a sharp might indicate that a note is natural (instead of sharp) if it previously was flat, and altered notes might be notated without accidentals. Indeed, in early music—specially from the 14^{th} century—there was a tendency to use many altered notes for performance (the so-called *musica ficta*); still, often it was not allowed to indicate such altered notes in written music, since these did not follow the 'guidonian rule' of the hexachords (the so-called *musica retta*) written music was based upon at that time [15]. The theorisation on this topic creates great interest in the research community [2], and due to its complexity—which goes beyond the purpose of this paper—we only encode the accidentals printed in the original source without further interpretations; for a transcription in modern notation which contains 'editorial accidentals' of the *musica ficta*, cf. [20].

(iii) Unlike modern notation, where—except for dotted notes—each note is always divided in two equal neighbour smaller notes (e. g., a quarter note is divided by two eighth notes), in mensural notation a non dotted note might be divided not only in two, but also in three neighbour smaller notes, depending on its 'mensuration' [1]. The mensuration of *brevis* and *semibrevis* (so-called *tempus* and *prolatio*) is indicated by the time signature, e. g., 𝐂 indicates *tempus imperfectum cum prolatione imperfecta* [1], i. e., a duple metre with binary subdivision. While imperfect *prolatio* would be the equivalent of a simple metre in modern notation (i. e., binary subdivision of each bit), perfect *prolatio* would be the equivalent of a compound metre (i. e., ternary subdivision of each bit). [22] In the semantic encoding, mensuration was indicated by adding '_imperfect' or '_perfected' to each note, rest, and ligature, e. g., 'note-D4_minima_imperfect'.

6. LIMITATIONS

Beyond the already discussed aspects that might differ between the engraved images, i. e., the visual representation of **mens and MEI files engraved by VHV, and the original source, such as different position of accidentals and rests, or missing elements (e. g., custos, ligature stems, or clefs, time, and key signatures in specific cases), other differences between the presented diplomatic edition and the original print should be mentioned. One is the time signature so-called, the *alla breve* [1], i. e., 𝔠, which indicates, as well as 𝐂, *tempus* and *prolatio* imperfect. Although in the original source, this time signature presents two graphical variants (cf. Figure 6), this is displayed by a unique symbol in the engraved images (cf. Figure 1). Another element which is not included in the engraved images is the use of descriptive or decorative initials, which gained great relevance in printing music collections since, unlike ordinary text editions, music prints required normally one or even two woodcut initials for every page [3]. Since the *Il Lauro Secco* anthology was the first music collection published by Baldini as ducal printer in the court of Alfonso II d'Este, ornamental elements as the initials (cf. Figure 7) received special attention in the luxurious collection [4]. [23]

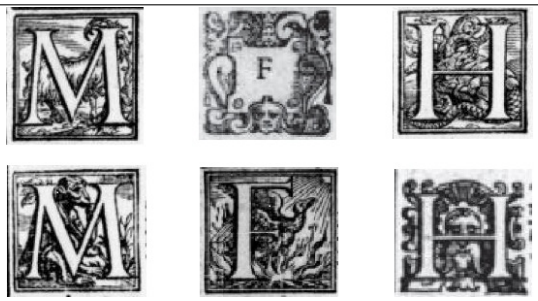


Figure 7. The anthology presents two different sets of decorative initials, which also contemplate variations of the same letter. In the musical parts two alternatives of the initials 'M' (left), 'F' (middle), and 'H' (right) are displayed.

7. CONCLUSIONS AND FUTURE WORK

We symbolically codified the *Il Lauro Secco* anthology in white mensural notation. For analytic and performance purposes, a diplomatic edition, encoded in **mens and MEI formats as well as its engraved images in pdf format, is provided. For OMR applications, ground truth in agnostic and semantic formats is presented, and the converters required to automatically generate such files from **mens encoding are also freely available. The recent development of a user-friendly encoding format for mensural notation (**mens) and a suitable encoding interface (VHV) made this work possible. Yet, given the novelty of these tools, specific aspects of mensural notation are still being developed, e. g., the lack of rests' and accidentals' position markers, or a custos indicator. By evaluating these aspects, we aim at encouraging a further development of the available tools. Due to the higher standardisation—w.r.t. handwritten and black mensural notation—of white mensural notated printed sources, and to the vast array of available scanned copies, we want to apply the methodology presented in this work to similar repertoires, thus stimulating further advance of OMR technology for early music.

[22] In modern notation, a dot must be added to the quarter note (bit note) in compound metre while in mensural notation this would not be required.

[23] Given the importance of decorative initials in Renaissance prints, the possibility to include them in VHV might be something to consider.

8. REFERENCES

[1] W. Apel, *The notation of polyphonic music, 900-1600*. Cambridge, MA, USA: The Mediaeval Academy of America, 1961.

[2] M. Bent, "Diatonic ficta," in *Renaissance music*, K. Kreitner, Ed. New York, NY, USA: Routledge, 2011, pp. 327–374.

[3] J. A. Bernstein, *Music printing in Renaissance Venice: The Scotto press (1539-1572)*. Oxford, UK: Oxford University Press, 1998.

[4] K. Butler, "Printed borders for sixteenth-century music or music paper and the early career of music printer Thomas East," *The Library*, vol. 19, pp. 174–202, 2018.

[5] J. Calvo-Zaragoza, I. Barbancho, L. J. Tardón, and A. M. Barbancho, "Avoiding staff removal stage in optical music recognition: Application to scores written in white mensural notation," *Pattern Analysis and Applications*, vol. 18, pp. 933–943, 2015.

[6] J. Calvo-Zaragoza and D. Rizo, "End-to-end neural optical music recognition of monophonic scores," *Applied Sciences*, vol. 8, pp. 1–23, 2018.

[7] S. J. Coluzzi, "Black sheep: The phrygian mode and a misplaced madrigal in Marenzio's seventh book (1595)," *The Journal of Musicology*, vol. 30, pp. 129–179, 2013.

[8] J. E. Cumming, C. McKay, J. Stuchbery, and I. Fujinaga, "Methodologies for creating symbolic corpora of western music before 1600," in *Proc. of the International Society for Music Information Retrieval Conference*. Paris, France: ISMIR, 2018, pp. 491–498.

[9] K. Desmond, "Measuring polyphony: A project to encode and analyse late medieval polyphony," in *Workshop on SIMSSA XI*. Vancouver, BC, Canada: AMS/SMT, 2016.

[10] M. P. Ferreira, *The notation of the Cantigas de Santa Maria: Diplomatic edition*. Lisbon, Portugal: CESEM, 2017.

[11] D. Fober, S. Letz, and Y. Orlarey, "Open source tools for music representation and notation," in *Proc. of Sound and Music Computing Conference*, Paris, France, 2004, pp. 91–95.

[12] R. Freedman, *Music in the Renaissance*. New York, NY, USA: WW Norton, 2013.

[13] E. Gardiner and R. G. Musto, *The digital humanities: A primer for students and scholars*. Cambridge, UK: Cambridge University Press, 2015.

[14] J. Grier, *The critical editing of music: History, method, and practice*. Cambridge, UK: Cambridge University Press, 1996.

[15] D. J. Grout and C. V. Palisca, *A history of Western music*. New York, NY, USA: Norton, 2001.

[16] J. Hajic jr, M. Dorfer, G. Widmer, and P. Pecina, "Towards full-pipeline handwritten OMR with musical symbol detection by U-Nets," in *Proc. of the International Society for Music Information Retrieval Conference*. Paris, France: ISMIR, 2018, pp. 23–27.

[17] D. Huron, "Music information processing using the Humdrum Toolkit: Concepts, examples, and lessons," *Computer Music Journal*, vol. 26, pp. 11–26, 2002.

[18] H.-W. Nienhuys and J. Nieuwenhuizen, "Lilypond, a system for automated music engraving," in *Proc. of the XIV Colloquium on Musical Informatics*, Florence, Italy, 2003, pp. 167–171.

[19] A. Pacha and J. Calvo-Zaragoza, "Optical music recognition in mensural notation with region-based convolutional neural networks," in *Proc. of the International Society for Music Information Retrieval Conference*. Paris, France: ISMIR, 2018, pp. 23–27.

[20] E. Parada-Cabaleiro, A. Batliner, A. E. Baird, and B. Schuller, "The SEILS dataset: Symbolically encoded scores in modern-early notation for computational musicology," in *Proc. of the International Society for Music Information Retrieval Conference*. Suzhou, P. R. China: ISMIR, 2017, pp. 575–581.

[21] E. Parada-Cabaleiro, M. Schmitt, A. Batliner, and B. W. Schuller, "Musical-linguistic annotations of Il Lauro Secco," in *Proc. of the International Society for Music Information Retrieval Conference*. Paris, France: ISMIR, 2018, pp. 461–467.

[22] L. Pugin, "Going digital: Finding the right path for critical music editions," in *À Fresco: Mélanges offerts au professeur Étienne Darbellay*, B. Boccardo and G. Starobinski, Eds. Bern, Switzerland: Peter Lang, 2013, pp. 247–268.

[23] L. Pugin, "Interaction perspectives for music notation applications," in *Proc. of the International Workshop on Semantic Applications for Audio and Music*. Monterey, CA, USA: ACM, 2018, pp. 54–58.

[24] L. Pugin and T. Crawford, "Evaluating OMR on the early music online collection," in *Proc. of the International Society for Music Information Retrieval Conference*. Curitiba, Brazil: ISMIR, 2013, pp. 439–444.

[25] A. M. Rebelo Silva, "Robust optical recognition of handwritten musical scores based on domain knowledge," Ph.D. dissertation, University of Porto, 2012.

[26] E. Ricciardi, "The Tasso in music project," *Early Music*, vol. 43, pp. 667–671, 2015.

[27] D. Rizo, J. Calvo-Zaragoza, and J. M. Iñesta, "Muret: A music recognition, encoding, and transcription tool,"

in *Proc. of the International Conference on Digital Libraries for Musicology*. Paris, France: ACM, 2018, pp. 52–56.

[28] D. Rizo, N. P. León, and C. S. Sapp, "White mensural manual encoding: From humdrum to mei," *Cuadernos de Investigación Musical*, pp. 373–393, 2019.

[29] D. Rizo, B. Pascual Sánchez, J. M. Iñesta, A. Ezquerro Esteban, and L. A. González Marín, "Towards the digital encoding of hispanic white mensural notation," *Anuario Musical*, pp. 293–304, 2017.

[30] P. Roland, "MEI as an editorial music data format," in *Digitale edition zwischen experiment und standardisierung*, P. Stadler and J. Veit, Eds. Tübingen, Germany: Max Niemeyer, 2009, pp. 175–194.

[31] P. Roland, A. Hankinson, and L. Pugin, "Early music and the music encoding initiative," *Early Music*, vol. 42, pp. 605–611, 2014.

[32] P. Roland and J. Kepper, "Music encoding initiative guidelines," *Version 3.0.0*, 2016.

[33] C. S. Sapp, "Verovio humdrum viewer," in *Music Encoding Conference (MEC)*, Tours, France, 2017.

[34] L. J. Tardón, S. Sammartino, I. Barbancho, V. Gómez, and A. Oliver, "Optical music recognition for scores written in white mensural notation," *Journal on Image and Video Processing*, pp. 1–23, 2009.

[35] F. Wiering, "Digital critical editions of music: A multidimensional model," in *Modern Methods for Musicology*, T. Crawford and L. Gibson, Eds. London, UK: Routledge, 2016, pp. 23–45.

THE HARMONIX SET: BEATS, DOWNBEATS, AND FUNCTIONAL SEGMENT ANNOTATIONS OF WESTERN POPULAR MUSIC

Oriol Nieto[1] **Matthew McCallum**[1] **Matthew E. P. Davies**[2]
Andrew Robertson[3] **Adam Stark**[4] **Eran Egozy**[5]
[1] Pandora Media, Inc., Oakland, CA, USA
[2] INESC TEC, Porto, Portugal
[3] Ableton AG, Berlin, Germany
[4] MI·MU, London, UK
[5] MIT, Cambridge, MA, USA
onieto@pandora.com

ABSTRACT

We introduce the Harmonix set: a collection of annotations of beats, downbeats, and functional segmentation for over 900 full tracks that covers a wide range of western popular music. Given the variety of annotated music information types in this set, and how strongly these three types of data are typically intertwined, we seek to foster research that focuses on multiple retrieval tasks at once. The dataset includes additional metadata such as MusicBrainz identifiers to support the linking of the dataset to third-party information or audio data when available. We describe the methodology employed in acquiring this set, including the annotation process and song selection. In addition, an initial data exploration of the annotations and actual dataset content is conducted. Finally, we provide a series of baselines of the Harmonix set with reference beat-trackers, downbeat estimation, and structural segmentation algorithms.

1. INTRODUCTION

The tasks of beat detection [8], downbeat estimation [2], and structural segmentation [34] constitute a fundamental part of the field of MIR. These three musical characteristics are often related: downbeats define the first beat of a given music measure, and long structural music segments tend to begin and end on specific beat locations – frequently on downbeats [10]. The automatic estimation of such information could result in better musical systems such as more accurate automatic DJ-ing, better intra- and inter-song navigation, further musicological insights of large collections, *etc*. While a few approaches exploiting more than one of these musical traits have been proposed [2, 11, 25], the amount of human annotated data containing the three of them for a single collection is scarce. This limits the training potential of certain methods, especially those that require large amounts of information (e.g., deep learning [18]).

In this paper we present the Harmonix set: human annotations of beats, downbeats, and functional segmentation for 912 tracks of western popular music. These annotations were gathered with the aim of having a significant amount of data to train models to improve the prediction of such musical attributes, which would later be applied to various products offered by Harmonix, a video game company that specializes in musically-inspired games. By releasing this set to the public, our aim is to let the research community explore and exploit these annotations to advance the tasks of beat tracking, downbeat estimation, and automatic functional structural segmentation. We discuss the methodology to acquire these data, including the song selection process, and the inclusion of standard identifiers (AcoustID and MusicBrainz) and a set of automatically extracted onset times for the first 30 seconds of the tracks to allow other researchers to more easily access and align, when needed, the actual audio content. Furthermore, we present a series of results with reference algorithmic approaches in the literature with the goal of having an initial public benchmark of this set.

The rest of this work is organized as follows: Section 2 contains a review of the most relevant public datasets of the tasks at hand; Section 3 discusses the Harmonix set, including the data gathering, their formatting, and various statistics; Section 4 presents numerous benchmarks in the set; and Section 5 draws some final conclusions and discusses future work.

2. BACKGROUND

Several datasets with beat, downbeat, and/or segment annotations have been previously published, and in this section we review the most relevant ones.

2.1 Beat and Downbeat Tracking Sets

Over the last 15 years, many annotated datasets for beat and downbeat tracking have appeared in the literature whose primary purpose has been to allow the comparison of newly proposed and existing algorithms. However, the well-known difficulties of sharing the audio component of large annotated datasets has led to a rather ad-hoc usage of different datasets within the literature, and to a lesser extent, the choice of which evaluation metrics are selected to report accuracy. Conversely, the MIREX evaluation campaign provides a more rigid model for evaluation, by withholding access to private test datasets, and instead relying on the submission of the competing algorithms in order to compare them under controlled conditions. To this end, MIREX can be a useful reference point to consider these two music analysis tasks from the perspective of annotated data.

The MIREX Audio Beat Tracking (ABT) task [1] first appeared in 2006 and ran on a single dataset [28,30] with the performance of the submitted algorithms determined using one evaluation metric, the P-Score. After a brief hiatus, the task reappeared in 2009 with the addition of a dataset of Chopin Mazurkas [36], and the inclusion of multiple evaluation metrics [5]. The task continued to run in this way until the incorporation of the SMC dataset [16] in 2012, from which point it has remained constant. In 2014, the Audio Downbeat Estimation (ADE) task [2] was launched which comprised six different datasets from diverse geographic and stylistic sources: The Beatles [24]; Hardcore, Jungle, Drum and Bass (HJDB) [15]; Turkish [41]; Ballroom [21]; Carnatic [42]; and Cretan [17], with the evaluation conducted using the F-measure. While the datasets contained with these two MIREX tasks are by no means exhaustive, they provide a useful window to explore both how the audio data is chosen and how the annotation is conducted for these MIR tasks. To this end, we provide the following breakdown of different properties including reference to both MIREX and non-MIREX datasets.

Duration: Unlike the task of structural segmentation, beat and downbeat tracking datasets can be comprised of musical excerpts [14, 15, 21, 28] rather than full tracks [9,12,13,24]. **Number of annotators**: The initial MIREX beat tracking dataset [28] was unique in that it contained the annotations of 40 different people who tapped the beat to the music excepts. Conversely, other datasets used multiple annotators contributing across the dataset [16], a single annotator for all excerpts [14], or even deriving the annotations in a semi-automatic way from the output of an algorithm [24]. **Annotation post-processing**: Given some raw tap times or algorithm output, these can either be left unaltered [28] or, as is more common, iteratively adjusted until they are considered perceptually accurate by the annotator(s) [14–16]. **Style-specificity**: While some datasets are designed to have broad coverage across a range of musical styles [13, 14, 23], others target a particular group of styles [15, 21], a single style [9], the work of a given artist [12, 24] or even multiple versions of the same pieces [36]. **Western / Non-Western**: Similarly, the make up of the dataset can target underrepresented non-western music [33,41,42]. **Perceived difficulty**: Finally, the choice of musical material can be based upon the perceived difficulty of the musical excerpts, either from the perspective of musical or signal level properties [16].

2.2 Structural Segmentation Sets

The task of structural segmentation has been particularly active in the MIR community since the late 2000s. Similarly to the beat tracking task, several datasets have been published, and some of them have evolved over time. This task is often divided into two subtasks: segment boundary retrieval and segment labeling. All well-known published datasets contain both boundary and label information. One of the major challenges with structural segmentation is that this task is regarded as both *ambiguous* (i.e., there may be more than one valid annotation for a given track [26]) and *subjective* (i.e., two different listeners might perceive different sets of segment boundaries [4]). This has led to different methodologies when annotating and gathering structural datasets, thus having a diverse ecosystem of sets to choose from when evaluating automatic approaches.

The first time this task appeared on MIREX was in 2009, [3] where annotations from The Beatles dataset (which also includes beat and downbeat annotations, as previously described) and a subset of the Real World Computing Popular Music Database (RWC) [13] were employed. These sets contain several functional segment annotations for western (The Beatles) and Japanese (RWC) popular music. These segment functions describe the *purpose* of the segments, e.g.: "solo," "verse," "chorus." A single annotation per track is provided for these two sets. The Beatles dataset was further revised at the Tampere University of Technology, [4] and additional functional segment annotations for other bands were added to The Beatles set, which became known as the Isophonics Dataset [24]. No beat or downbeat annotations were provided to the rest of the tracks in Isophonics, and the final number of tracks with functional structural segment annotations is 300. The number of annotated tracks in RWC is 365.

To address the open problems of ambiguity and subjectivity, further annotations per track from several experts could be gathered. That is the case with the Structural Annotations for Large Amounts of Music Information (SALAMI) dataset [39], where most of its nearly 1,400 tracks have been annotated by at least 2 musical experts. Similarly, the Structural Poly Annotations of Music (SPAM) dataset [32] provides 5 different annotations for 50 tracks. These two sets not only contain functional levels of annotations, but also large and small scale segments where only single letters describing the similarity between segments are annotated. Thus, these can be seen as sets that contain *hierarchical* data, which pose significant chal-

[1] https://www.music-ir.org/mirex/wiki/2006:Audio_Beat_Tracking
[2] https://www.music-ir.org/mirex/wiki/2014:Audio_Downbeat_Estimation
[3] https://www.music-ir.org/mirex/wiki/2009:Structural_Segmentation
[4] http://www.cs.tut.fi/sgn/arg/paulus/beatles_sections_TUT.zip

lenges, since ambiguity and subjectivity span across multiple layers [26] and remain largely unexploited in the MIREX competition [7,40]. As opposed to Isophonics and RWC, these two sets contain highly diverse music in terms of genre: from world-music to rock, including jazz, blues, and live music.

The following properties typically define segmentation datasets: **Number of annotators**: This can help when trying to quantify the amount of disagreement among annotators [26, 32], or when developing approaches that may yield more than one potentially valid segmentation. **Hierarchy**: The levels of annotations contained in the set. It typically contains functional, large, and/or small segment annotations. When only one level of annotations is provided, these are typically called *flat* segment annotations.

3. THE HARMONIX SET

In this section we present the Harmonix set, including the methodology of acquiring the data, its motivation, its contents, and a set of annotation statistics. The Harmonix set is publicly available on-line. [5]

3.1 Data Gathering

The primary motivation of this work is based on the need to create gameplay data for rhythm-action games (also known as beat matching games). Many such games exist, from early pioneers like Parappa The Rapper and Beatmania, to the rock simulation games Guitar Hero and Rock Band, as well as community-based games like OSU and more recently, VR games like Beat Saber. In most cases the gameplay data (also referred to as beatmaps), consisting of note locations in a song, are hand-authored. In certain games, additional control data may be desirable. For example, in the rock simulation games, where a 3D depiction of a rock concert is rendered, it can be desirable to simulate flashing lights (on the beat) or lighting color palette changes (on section boundaries). Again, these data tend to be hand-authored.

Harmonix's desire was to implement a suite of automatic music analysis tools that estimate certain musical attributes in order to expedite the process of hand-authoring gameplay data, or in some cases, to fully automate the process of creating these data. The songs of the Harmonix set were gathered and hand-annotated to create a ground-truth dataset for training and testing these algorithms.

The mix of genres in this corpus were chosen to be typical of ones used in the rhythm-action games, with a somewhat higher tendency towards EDM and popular songs suitable for dancing (see Figure 1 for the full genre distribution). As such, most tend to have a very stable tempo and a 4/4 time signature. However, we also added a selection of songs that may not be typical of dance or pop music to increase variety. Some of these (Classic Rock, Country, Metal) may have less stable tempo (where drums are played by actual musicians as opposed to drum-machines

[5] https://github.com/urinieto/harmonixset

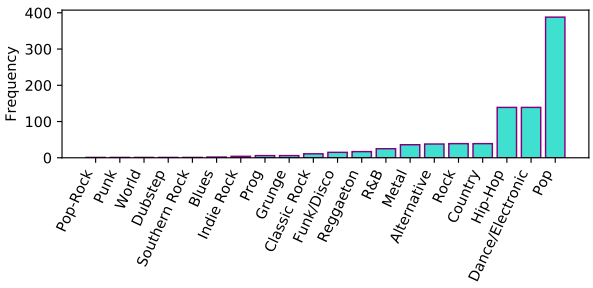


Figure 1. Genre distribution of the Harmonix set.

or DAW-based productions) and may deviate from a strict 4/4 meter.

All songs were annotated by trained professional musicians who regularly work in music production environments. As the project went on, the majority of annotation work fell to only a few individuals who became specialized in this task. Annotations were created in Digital Audio Workstation software (such as Reaper or Logic). First, a MIDI tempo track was established that corresponded to the song audio. Then beats, downbeats, and sections were coded into the MIDI track using note events and text events. MIDI files were then exported and automatically converted to a text-based representation of beats, downbeats, and named section boundaries. Every song was verified once by the original annotator.

3.2 Dataset Contents

The Harmonix set contains manual annotations for 912 western popular music tracks, thus being the largest published dataset to date containing beats, downbeats, and function structural segmentation information. The annotations and some of the song-level metadata are distributed via JAMS [19] files, one per track. This format is chosen given its simplicity when storing multi-task annotations plus song- and corpus-level metadata. Each JAMS file contains the beat, downbeat, and functional segmentation annotations, plus a set of estimated onsets for the first 30 seconds of the audio. These onsets are intended to help aligning the audio in case researchers obtain audio data with different compression formats that might include certain small temporal offsets. This onset information was computed using librosa [27], with their default parameters. [6]

For the sake of transparency and usability, we also publish the raw beats, downbeats, and segmentation data as space-separated text files, two per track: one for beats and downbeats, and the other for segments. We also distribute the code that converts these raw annotations into unified JAMS files. Furthermore, we provide other identifiers with the aim of easily retrieving additional metadata and/or audio content for each song. These identifiers include:

- **MusicBrainz** [7] : open music encyclopedia including

[6] librosa 0.6.3, using Core Audio on macOS 10.13.6.
[7] https://musicbrainz.org/

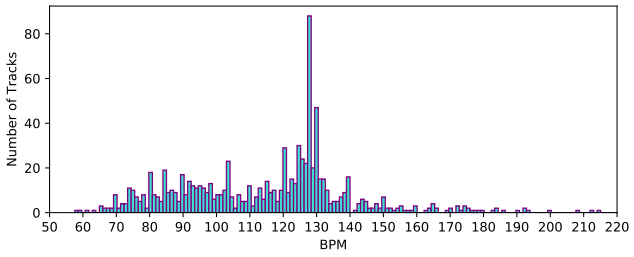


Figure 2. Tempo distribution of the tracks in the set.

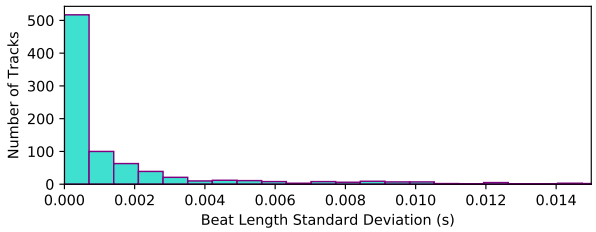


Figure 3. Standard deviation of the tempo distribution.

unique identifiers for recordings, releases, artists, etc.

- **AcoustID** [8] : open source fingerprinting service to easily match audio content, typically associated with MusicBrainz identifiers.

Finally, we provide a single CSV file including additional metadata information such as genre, time signature, and BPM.

3.3 Data Statistics

In this subsection we provide several data insights obtained from the annotations to give an objective overview of the set. In Figure 2 we show the estimated tempo distribution in beats-per-minute (BPM) per track. These estimations were computed using the track-level median inter-beat-interval (IBI) for each of the annotated beats in a given track. There is a clear peak at 128 BPM, which could be explained by being the most common tempo in electronic dance music [29]. Furthermore, in Figure 3 we plot the standard deviation of the IBI. We can clearly see that the tempo is remarkably steady in this dataset, which is expected given the type of musical genres it spans.

In terms of segment statistics, we show data based on certain attributes described in a MIREX meta-analysis of the segmentation task [40]. In Figure 4 we plot track-level histograms for the number of segments, and the number of unique segments (i.e., those with the same associated label). Both distributions seem to be unimodal and centered around 10 and 11 for the number of segments per tracks, and around 6 and 7 for the number of unique labels per track. This differs from the number of unique segments in The Beatles dataset, which is centered around 4 per track [31].

[8] https://acoustid.org/

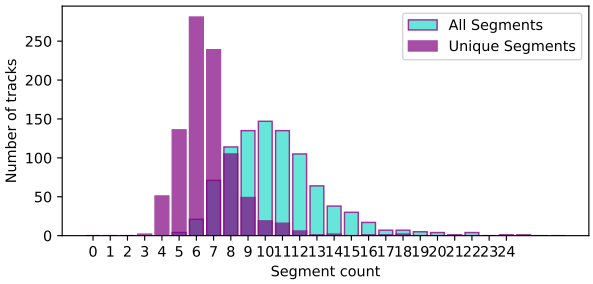


Figure 4. Number of segments per track, based on their segment labels.

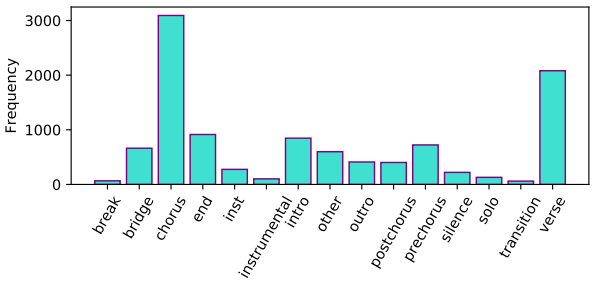


Figure 5. Most common segment labels.

Figure 5 shows the frequency in which the most common segment labels appear in the set. The labels "chorus" and "verse" dominate the distribution, as these functional parts are common in western popular music. The plot also shows potentially repeated labels like "inst" and "instrumental." A further inter-song analysis of the labels might be necessary to potentially merge certain labels and thus unify the vocabulary of the set.

We plot in Figure 6 the distribution of the segment lengths, in seconds, across the entire dataset. As we showed in Figure 2, there is a majority of tracks at 128 BPM, for which a duration of 15 seconds would correspond to a segment of exactly 32 beats. This, in the common 4/4 time signature, would result in 8 bars per each 15-second segment in that tempo, and 8 bars are common in electronic dance music [29].

Finally, and thanks to having access to the annotated downbeats, we show in Figure 7 the number of segments

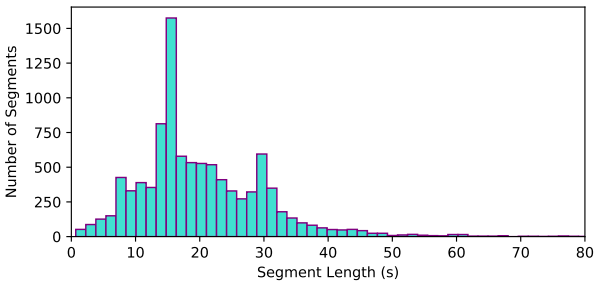


Figure 6. Segment length distribution.

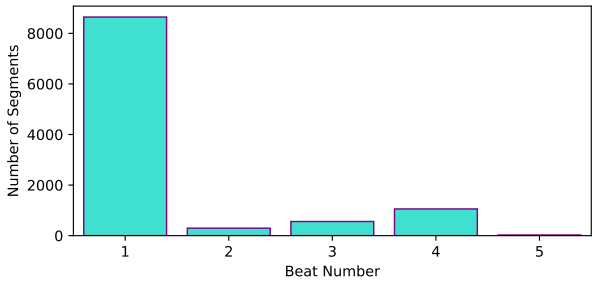


Figure 7. Number of segments based on their starting beat position within a bar.

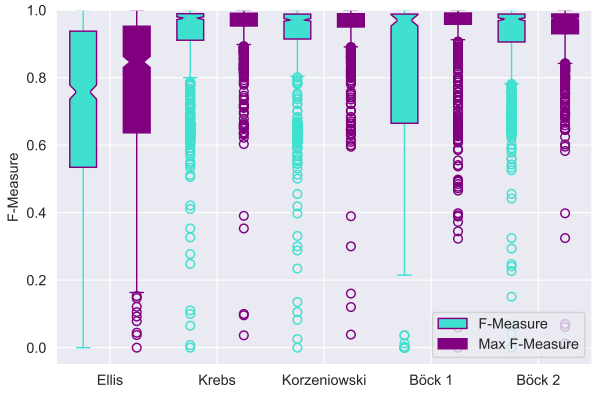


Figure 8. Beat tracking performance over the Harmonix set, for the algorithms Ellis [8], Krebs [22], Korzeniowski [20], Böck 1 - the "BeatDetector" technique from [3], and Böck 2 - the "BeatTracker" technique from [3].

starting at a specific beat within a given bar. We can see that the vast majority of segments (81.1%) start in a downbeat. Interestingly, several segments (10%) start in position 4, thus showing that 1-beat count-ins are more common than other types of count-ins on this dataset (a popular example of a 1-beat count-in song is Hey Jude by The Beatles, where the (1) is on the Jude and Hey is the (4) of the previous bar).

4. RESULTS

4.1 Beat Results

In order to establish performance baselines over the dataset for the task of beat-tracking, we have evaluated a number of openly available beat tracking algorithms on the dataset [3,8,20,22]. Each of these algorithms can be found in either the madmom [1] or librosa python libraries. [9] By running these algorithms in other datasets with the same metrics, a comparison of datasets could ultimately be performed. The results are also included in the dataset repository in CSV format. This is intended as a convenience for any future work that wishes to evaluate novel algorithms against these benchmarks.

The beat tracking results for the aforementioned algorithms are displayed in Figure 8. They are evaluated across two metrics, F-Measure, and Max F-Measure, where the latter refers to the maximum F-Measure obtained per track when evaluated across double and half-time metrical variations in the annotated beats provided with this dataset. In all experiments a tolerance window of ± 70 ms was employed in order to compute the F-Measure. For half-time metrical variations, both the downbeat and upbeat alignments were tested for a maximum F-Measure value. While [8] is the most computationally efficient of the algorithms, we see clear gains in the more recently developed methods. When investigating the types of errors present in the beat position estimates from [8], it was found the most common error was the alignment of beat phase. Often beat positions landed on the half beat or quarter beat, resulting in an F-Measure of 0 when this misalignment is consistent throughout the track. When comparing F-Measure and Max F-Measure metrics, it can be seen that with this dataset both [8] and the "BeatDetector" algorithm from [3] have a significant number of double-half time errors, compared to the other algorithms evaluated. Unlike the "BeatTracker" algorithm in [3], the "BeatDetector" algorithm assumes constant tempo.

4.2 Downbeat Results

Unfortunately, the availability of open source downbeat estimation libraries is limited. In order to provide a baseline for downbeat detection performance with the Harmonix set specifically, results have been evaluated with the downbeat detection algorithms available in [1] in addition to Durand's algorithm [6] [10], making three algorithms in total. The algorithms from the madmom python package [1] include the method proposed in [2] using the annotated beat positions as input, and the dynamic Bayesian bar tracking processor using the input from the RNN bar processor activation function. The results can be seen in Figure 9 in terms of F-Measure with a tolerance window of ± 70 ms. The superior performance of [2], which has oracle annotated beat information, highlights the importance of reliable beat tracking for downbeat estimation performance, and the interdependence between the beat tracking and downbeat estimation tasks.

4.3 Segmentation Results

There are several open source structural segmentation algorithms available in the Music Structure Analysis Framework (MSAF) [32]. [11] We run the best performing ones on the Harmonix set: (i) Structural Features [38] to identify boundaries, and (ii) 2D-Fourier Magnitude Coefficients (2D-FMC) [31] to label the segments based on their acoustic similarity. Constant-Q Transforms [37] are the selected

[9] We used madmom 0.16.1 and librosa 0.6.3. We noticed a bias in this librosa version where beats were offset by a consistent number of milliseconds. More specifically, we employed librosa's `beat.beat_track` method with default arguments on macOS 10.13.6.

[10] Not open source, shared via private correspondence.

[11] MSAF version dev-0.1.8.

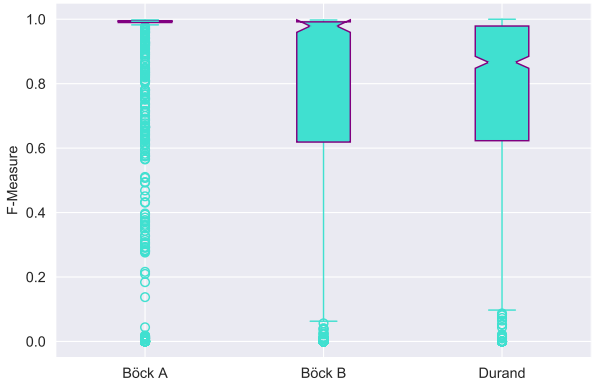


Figure 9. Downbeat tracking performance over the Harmonix set, for the algorithms Böck A [2] and Böck B - a dynamic Bayesian network provided within the madmom package [1], and Durand [6].

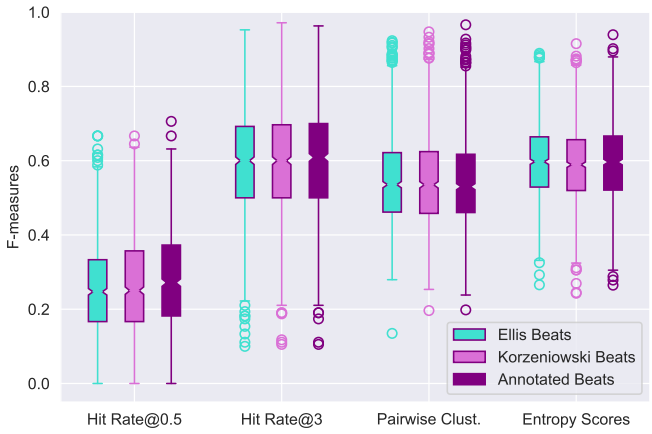


Figure 10. Segmentation results over the Harmonix set, using Structural Features for boundaries, 2D-FMC for the labeling process, and three types of beat information.

audio features given their ability to capture both timbral and harmonic content, and the default parameters in MSAF are the ones employed when computing these results. We use `mir_eval` [35] to evaluate these algorithms, and report the F-measures for the most common metrics: Hit Rate with 0.5 and 3 second windows for boundary retrieval, and Pairwise Frame Clustering and Entropy Scores for the labeling process. These algorithms can use beat-synchronized features, and we ran each algorithm three times, depending on the following beat information: (i) Ellis' estimations, (ii) Korzeniowski's estimations, and (iii) annotations from the Harmonix set. Thus, we are able to assess the segmentation results when employing the worst and best performing beat trackers from our previous study, plus those computed using human annotated beats. Song-level results for these three different runs are available as CSV files in the dataset repository disclosed above.

In Figure 10 all segmentation results are shown. The results in turquoise boxplots (on the left side) display the metrics of the algorithms when running on Ellis' beat-synchronized features, those in light pink (in the middle) correspond to the results computed with Korzeniowski's beats, while the purple boxplots (on the right) show those using annotated beats instead. Given how related boundary retrieval is with respect to precise beat placement, it is not unexpected to see an improvement in the boundary metrics (Hit Rates) when using more accurate beat data. The boxplots further show that the smaller the time window used in the Hit Rate metrics the more accurate the beat information should ideally be. In other words, Korzeniowki's beats yield very similar results than those from human annotations when using a 3 second window, but there is clearly room for enhancement (in terms of beat tracking) when using 0.5 second windows, where the segmentation results using human annotated beats outperform any of the others that employ estimated ones. On the other hand, it is worth noting that the label results do not seem to depend as much on the quality of the beats in order to produce their outcomes, as the three different runs yield similar results for the Pairwise Frame Clustering and Entropy Scores metrics. As mentioned in Section 2.2, structural segmentation is a challenging task especially due to ambiguity, subjectivity, and hierarchy, and this is reflected in the overall results, which exhibit notable room for improvement.

5. CONCLUSIONS

We presented the Harmonix set, the largest dataset in terms of human annotations containing the following three types of music information: beats, downbeats, and function structural segments. This set contains mostly western popular music, with strong emphasis on Pop, EDM, and Hip-Hop. We provide metadata in terms of genre, song title, and artist information along with standard identifiers such as MusicBrainz and AcoustID plus predicted onset information to allow easier matching and alignment with audio data. We discussed a set of results using current algorithms in the literature in terms of beat tracking, downbeat estimation, and structural segmentation to disclose an initial public benchmark of the set. Given the rather large nature of the set and the three different types of music information contained in it, it is our hope that researchers employ these data not only to further advance one of these three MIR tasks individually, but also to potentially combine them to yield superior approaches in the near future.

6. ACKNOWLEDGMENTS

We would like to thank Simon Durand for sharing his downbeat estimation implementation. Matthew E.P. Davies is supported by Portuguese National Funds through the FCT-Foundation for Science and Technology, I.P., under the project IF/01566/2015.

7. REFERENCES

[1] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer. Madmom: A new python audio and music signal processing library. In *Proceedings of the 24th*

ACM international conference on Multimedia, pages 1174–1178, 2016.

[2] S. Böck, F. Krebs, and G. Widmer. Joint Beat and Downbeat tracking with recurrent neural networks. *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 255–261, 2016.

[3] S. Böck and M. Schedl. Enhanced beat tracking with context-aware neural networks. In *Proc. Int. Conf. Digital Audio Effects*, pages 135–139, 2011.

[4] M. J. Bruderer, M. F. McKinney, and A. Kohlrausch. The Perception of Structural Boundaries in Melody Lines of Western Popular Music. *Musicæ Scientiæ*, 13(2):273–313, 2009.

[5] M. E. P. Davies, N. Degara, and M. D. Plumbley. Evaluation methods for musical audio beat tracking algorithms. Technical Report C4DM-TR-09-06, Centre for Digital Music, Queen Mary University of London, 2009.

[6] S. Durand, J. P. Bello, B. David, and G. Richard. Feature adapted convolutional neural networks for downbeat tracking. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 296–300. IEEE, 2016.

[7] A. F. Ehmann, M. Bay, J. S. Downie, I. Fujinaga, and D. D. Roure. Music Structure Segmentation Algorithm Evaluation: Expanding on MIREX 2010 Analyses and Datasets. In *Proc. of the 13th International Society for Music Information Retrieval Conference*, pages 561–566, Miami, FL, USA, 2011.

[8] D. P. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[9] V. Eremenko, E. Demirel, B. Bozkurt, and X. Serra. Audio-aligned jazz harmony dataset for automatic chord transcription and corpus-based research. In *Proc. of the 16th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 483–490, 2018.

[10] J. T. Foote. Methods for the automatic analysis of music and audio. *FXPAL Technical Report FXPAL-TR-99-038*, 1999.

[11] M. Fuentes, B. McFee, H. C. Crayencour, S. Essid, and J. P. Bello. A Music Structure Informed Downbeat Tracking System Using Skip-Chain Conditional Random Fields and Deep Learning. In *Proc. of the 44th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brighton, UK, 2019.

[12] B. D. Giorgi, M. Zanoni, S. Böck, and A. Sarti. Multipath beat tracking. *Journal of the Audio Engineering Society*, 64(7/8):493–502, 2016.

[13] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC Music Database: Popular, Classical, and Jazz Music Databases. *International Conference on Music Information Retrieval*, (October):287–288, 2002.

[14] S. Hainsworth and M. Macleod. Particle filtering applied to musical tempo tracking. *EURASIP Journal on Applied Signal Processing*, 15:2385–2395, 2004.

[15] J. Hockman, M. E. P. Davies, and I. Fujinaga. One in the jungle: Downbeat detection in hardcore, jungle, and drum and bass. In *Proc. of the 13th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 169–174, 2012.

[16] A. Holzapfel, M. E. P. Davies, J. R. Zapata, J. L. Oliveira, and F. Gouyon. Selective sampling for beat tracking evaluation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(9):2539–2548, 2012.

[17] A. Holzapfel, F. Krebs, and A. Srinivasamurthy. Tracking the "odd": Meter inference in a culturally diverse music corpus. In *Proc. of the 15th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 425–430, 2014.

[18] E. J. Humphrey, J. P. Bello, and Y. LeCun. Moving Beyond Feature Design: Deep Architecture and Automatic Feature Learning in Music Informatics. In *Proc. of the 13th International Society for Music Information Retrieval Conference*, pages 403–408, Porto, Portugal, 2012.

[19] E. J. Humphrey, J. Salamon, O. Nieto, J. Forsyth, R. M. Bittner, and J. P. Bello. JAMS: A JSON Annotated Music Specification for Reproducible MIR Research. In *Proc. of the 15th International Society for Music Information Retrieval Conference*, pages 591–596, Taipei, Taiwan, 2014.

[20] F. Korzeniowski, S. Böck, and G. Widmer. Probabilistic extraction of beat positions from a beat activation function. In *ISMIR*, pages 513–518, 2014.

[21] F. Krebs, S. Böck, and G. Widmer. Rhythmic pattern modeling for beat and downbeat tracking in musical audio. In *Proc. of the 14th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 227–232, 2013.

[22] F. Krebs, S. Böck, and G. Widmer. An efficient state-space model for joint tempo and meter tracking. In *ISMIR*, pages 72–78, 2015.

[23] U. Marchand and G. Peeters. Swing ratio estimation. In *Proc. of the 18th Intl. Conf. on Digital Audio Effects (DAFx)*, pages 423–428, 2015.

[24] M. Mauch, C. Cannam, M. Davies, S. Dixon, C. Harte, S. Kolozali, D. Tidhar, and M. Sandler. OMRAS2 Metadata Project 2009. In *Late Breaking Session of the 10th International Society of Music Information Retrieval*, Kobe, Japan, 2009.

[25] M. C. McCallum. Unsupervised Learning of Deep Features for Music Segmentation. In *Proc. of the 44th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brighton, UK, 2019.

[26] B. McFee, O. Nieto, M. M. Farbood, and J. P. Bello. Evaluating hierarchical structure in music annotations. *Frontiers in Psychology*, 8(1337), 2017.

[27] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto. librosa: Audio and Music Signal Analysis in Python. In *Proc. of the 14th Python in Science Conference*, pages 18–25, Austin, TX, USA, 2015.

[28] M. F. McKinney, D. Moelants, M. E. P. Davies, and A. Klapuri. Evaluation of audio beat tracking and music tempo extraction algorithms. *Journal of New Music Research*, 36(1):1–16, 2007.

[29] D. Moelants. Hype vs. Natural Tempo: a Long-term Study of Dance Music Tempi. In *Proc. of the 10th International Conference on Music Perception and Cognition*, Sapporo, Japan, 2008.

[30] D. Moelants and M. McKinney. Tempo perception and musical content: What makes a piece fast, slow or temporally ambiguous. In *Proc. of the 8th Intl. Conf. on Music Perception and Cognition*, pages 558–562, 2004.

[31] O. Nieto and J. P. Bello. Music Segment Similarity Using 2D-Fourier Magnitude Coefficients. In *Proc. of the 39th IEEE International Conference on Acoustics Speech and Signal Processing*, pages 664–668, Florence, Italy, 2014.

[32] O. Nieto and J. P. Bello. Systematic Exploration of Computational Music Structure Research. In *Proc. of the 17th International Society for Music Information Retrieval Conference*, pages 547–553, New York City, NY, USA, 2016.

[33] L. O. Nunes, M. Rocamora, L. Jure, and L. W. Biscainho. Beat and Downbeat Tracking Based on Rhythmic Patterns Applied to the Uruguayan Candombe Drumming. In *Proc. of the 16th Intl. Society for Music Information Retrieval Conf. (ISMIR)*, pages 264–270, 2015.

[34] J. Paulus, M. Müller, and A. Klapuri. Audio-Based Music Structure Analysis. In *Proc of the 11th International Society of Music Information Retrieval*, pages 625–636, Utrecht, Netherlands, 2010.

[35] C. Raffel, B. Mcfee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis. mir_eval: A Transparent Implementation of Common MIR Metrics. In *Proc. of the 15th International Society for Music Information Retrieval Conference*, pages 367–372, Taipei, Taiwan, 2014.

[36] C. Sapp. Comparative Analysis of Multiple Musical Performances. In *Proc. of the 8th Intl. Conf. on Music Information Retrieval, (ISMIR)*, pages 497–500, 2007.

[37] C. Schörkhuber and A. Klapuri. Constant-Q Transform Toolbox for Music Processing. In *Proc. of the 7th Sound and Music Computing Conference*, pages 56–64, Barcelona, Spain, 2010.

[38] J. Serrà, M. Müller, P. Grosche, and J. L. Arcos. Unsupervised Music Structure Annotation by Time Series Structure Features and Segment Similarity. *IEEE Transactions on Multimedia, Special Issue on Music Data Mining*, 16(5):1229 – 1240, 2014.

[39] J. B. Smith, J. A. Burgoyne, I. Fujinaga, D. De Roure, and J. S. Downie. Design and Creation of a Large-Scale Database of Structural Annotations. In *Proc. of the 12th International Society of Music Information Retrieval*, pages 555–560, Miami, FL, USA, 2011.

[40] J. B. L. Smith and E. Chew. A meta-analysis of the MIREX Structure Segmentation task. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 251–256, Curitiba, Brazil, 2013.

[41] A. Srinivasamurthy, A. Holzapfel, and X. Serra. In search of automatic rhythm analysis methods for turkish and indian art music. *Journal of New Music Research*, 43(1):94–114, 2014.

[42] A. Srinivasamurthy and X. Serra. A Supervised Approach to Hierarchical Metrical Cycle Tracking from Audio Music Recordings. In *Proc. of the 39th IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5237–5241, 2014.

FMP NOTEBOOKS: EDUCATIONAL MATERIAL FOR TEACHING AND LEARNING FUNDAMENTALS OF MUSIC PROCESSING

Meinard Müller, Frank Zalkow
International Audio Laboratories Erlangen, Germany
{meinard.mueller, frank.zalkow}@audiolabs-erlangen.de

ABSTRACT

In this paper, we introduce a novel collection of educational material for teaching and learning fundamentals of music processing (FMP) with a particular focus on the audio domain. This collection, referred to as FMP notebooks, discusses well-established topics in Music Information Retrieval (MIR) as motivating application scenarios. The FMP notebooks provide detailed textbook-like explanations of central techniques and algorithms in combination with Python code examples that illustrate how to implement the theory. All components including the introductions of MIR scenarios, illustrations, sound examples, technical concepts, mathematical details, and code examples are integrated into a consistent and comprehensive framework based on Jupyter notebooks. The FMP notebooks are suited for studying the theory and practice, for generating educational material for lectures, as well as for providing baseline implementations for many MIR tasks, thus addressing students, teachers, and researchers.

1. INTRODUCTION

Music information retrieval (MIR) is an exciting and challenging area of research. Music not only connects people but also relates to many different research disciplines including signal processing, information retrieval, machine learning, musicology, and psychoacoustics. In its beginnings, research in MIR has borrowed many ideas and concepts from more established disciplines such as speech processing or computer linguistics. After twenty years, the MIR field has matured to an independent research area that has many things to offer to signal processing and other research disciplines [16]. In particular, thanks to the rich and challenging domain of music, there are many MIR tasks that can serve as motivation application scenarios for introducing, explaining, and studying techniques for audio processing, time-series analysis, and information retrieval.

In this paper, we introduce the **FMP notebooks**, which provide educational material for teaching and learning fundamentals of music processing. One primary goal of these notebooks is to give an exciting and easy-to-understand introduction to MIR with a particular focus on audio-related analysis and retrieval tasks. Following the textbook [13], the notebooks treat many well-established MIR tasks as summarized in Figure 1. Within each MIR task, fundamental algorithmic approaches and techniques are discussed in detail. Going beyond and complementing traditional toolboxes, the FMP notebooks closely combine textbook-like explanations with Python code examples. Interleaving technical concepts, mathematical details, code examples, illustrations, and sound examples within a unifying and interactive Jupyter notebook framework helps to bridge the gap between theory and practice. Furthermore, the notebooks can be easily adapted to generate educational material (such as figures and sound examples) for lectures and to realize baseline approaches for many MIR tasks. The FMP notebooks (as well as HTML exports) are accessible under a Creative Commons license at: https://www.audiolabs-erlangen.de/FMP

There are some excellent software toolboxes such as essentia [2], librosa [12], madmom [1],

Part	Title	Notions, Techniques & Algorithms	HTML	IPYNB
B	Basics	Basic information on Python, Jupyter notebooks, Anaconda package management system, Python environments, visualizations, and other topics	[html]	[ipynb]
0	Overview	Overview of the notebooks (https://www.audiolabs-erlangen.de/FMP)	[html]	[ipynb]
1	Music Representations	Music notation, MIDI, audio signal, waveform, pitch, loudness, timbre	[html]	[ipynb]
2	Fourier Analysis of Signals	Discrete/analog signal, sinusoid, exponential, Fourier transform, Fourier representation, DFT, FFT, STFT	[html]	[ipynb]
3	Music Synchronization	Chroma feature, dynamic programming, dynamic time warping (DTW), alignment, user interface	[html]	[ipynb]
4	Music Structure Analysis	Similarity matrix, repetition, thumbnail, homogeneity, novelty, evaluation, precision, recall, F-measure, visualization, scape plot	[html]	[ipynb]
5	Chord Recognition	Harmony, music theory, chords, scales, templates, hidden Markov model (HMM), evaluation	[html]	[ipynb]
6	Tempo and Beat Tracking	Onset, novelty, tempo, tempogram, beat, periodicity, Fourier analysis, autocorrelation	[html]	[ipynb]
7	Content-Based Audio Retrieval	Identification, fingerprint, indexing, inverted list, matching, version, cover song	[html]	[ipynb]
8	Musically Informed Audio Decomposition	Harmonic/percussive separation, signal reconstruction, instantaneous frequency, fundamental frequency (F0), trajectory, nonnegative matrix factorization (NMF)	[html]	[ipynb]

Figure 1. Overview of FMP notebooks.

Marsyas [20], or the MIRtoolbox [7], which provide open source software for MIR and music processing applications. We will give an overview of related toolboxes in Section 2. While such toolboxes aim at implementing a wide range of MIR functionalities, the main goal of the FMP notebooks is to promote the understanding of MIR concepts. Therefore, rather than providing compact and efficient code, the programming style used in the FMP code examples is simple and explicit with a flat functional hierarchy (at the cost of having some redundancy). The mathematical notation and the naming conventions used in the FMP notebooks are carefully matched to establish a close relationship between theory and practice. Furthermore, the notebooks allow a user to generate appealing multimedia objects such as figures and sound examples, which may be useful for lectures and scientific publications. In summary, educational and didactic considerations are the main guide in the development of the FMP notebooks. As such, we hope that these notebooks nicely complement existing open source toolboxes, fostering education and research in MIR.

The remainder of the paper is organized as follows. In Section 2, we review related software frameworks and toolboxes for audio and music processing. Then, in Section 3, we deal with the structure, content, and implementation of the FMP notebooks. In Section 4, we give some concrete examples of how the FMP notebooks can be used for learning and teaching music processing and MIR. Conclusions can be found in Section 5.

2. RELATED WORK

As said before, the main aim of the FMP notebooks is to help users to gain a deeper understanding of essential MIR techniques. Providing explicit and simple code examples (by consciously introducing redundancies), the notebooks are not intended to form a toolbox in a stricter sense. Instead, the FMP notebooks reimplement, integrate, and apply various functions that are also provided by existing toolboxes. In the following, we give a summary of open-source toolboxes that have been specifically designed for supporting MIR research.

There are a number of comprehensive and well-document toolboxes that provide modular source code for processing and analyzing music and audio signals. Prominent examples are the Marsyas toolbox [20], the MIRtoolbox [7], the jAudio toolbox [10], and the essentia library [2]. All these collections provide code for audio feature extraction as well as for MIR applications including music classification, melody extraction, beat tracking, and structure analysis.

There are also various toolboxes that focus on specific MIR applications such as the Chroma Toolbox [14] for chroma feature extraction, the Constant-Q Toolbox [19] for computing time-frequency transforms, the TSM Toolbox [3] for time-scale modification, the Tempogram Toolbox [5] for tempo and pulse tracking, and the SM Toolbox [15] as well as the MSAF toolbox [17] for audio structure analysis. While most of these toolboxes cover more traditional MIR techniques, the recent Python library madmom [1] also offers code for MIR approaches that employ deep learning techniques. Other useful toolboxes provide code for the evaluation of MIR approaches such as the mir_eval library [18] or for data augmentation such as the Audio Degradation Toolbox [9] or the muda library [11]. Other useful sources are the MIR notebooks [1] provided by Steve Tjoa as well as the companion website [2] of the textbook [8] on audio content analysis, which offers code for hands-on experience in audio and music processing. Furthermore, Xambó et al. [22] introduce a browser-based learning environment for teaching MIR and programming in high-schools.

In particular, we want to draw attention to the Python package librosa [12], which provides basic functions as well as advanced processing pipelines for several music and audio analysis tasks. librosa also comprises a gallery of advanced examples [3], which nicely illustrate how to use the package for approaching MIR tasks such as onset detection, music synchronization, harmonic-percussive separation, and audio structure analysis. The FMP notebooks are inspired by librosa and integrate, extend, and complement elements offered by this package. While librosa is designed to be an easy-to-use toolbox with convenient presets, the emphasis of the FMP notebooks is on the educational side providing detailed explanations of theoretical and practical aspects. We hope that the FMP notebooks serve as a good basis for carrying on with more advanced techniques as provided by powerful toolboxes such as librosa, essentia, or madmom.

3. STRUCTURE OF NOTEBOOKS

The FMP notebooks are structured in ten parts as shown by the table of Figure 1. **Part 0** (also containing this table) is the starting notebook, which is opened when calling https://www.audiolabs-erlangen.de/FMP. Besides giving an overview, this notebook also provides information on the license (Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License), the main contributors, and some links to related toolboxes. **Part B** provides **basic** introductions to the Jupyter notebook framework, the Python programming language, and other technical concepts underlying these notebooks (see Section 3.1). The main body of the FMP notebooks, which covers different music processing and MIR scenarios, consists of **Part 1** to **Part 8** (closely following the eight chapters of the textbook [13]). These parts are described in Section 3.2.

3.1 Technical Framework

The notebooks of **Part B** serve different purposes. First, these notebooks describe the main tools used for developing the FMP notebooks. Second, they give short intro-

[1] https://musicinformationretrieval.com/

[2] https://www.AudioContentAnalysis.org

[3] https://librosa.github.io/librosa/advanced.html

ductions of the relevant technical concepts while providing links to more detailed tutorials. Third, the notebooks give examples for best practices in programming as well as for generating and using code, figures, and sound elements. In the following, we describe the technical framework underlying the FMP notebooks while summarizing the content of **Part B**.

3.1.1 Jupyter Notebook

The FMP notebooks are based on the Jupyter notebook framework. This open-source web application allows users to create documents that contain live code, text-based information, mathematical formulas, plots, images, sound examples, and videos. Jupyter notebooks are often used as a publishing format for reproducible computational workflows [6]. They can be exported to a static HTML format, which makes it possible to generate web applications that can be accessed through standard web browsers with no specific technical requirements. **Part B** introduces some relevant elements of the Jupyter framework including practical aspects such as the most important Jupyter operators and keyboard shortcuts.

3.1.2 Installation

To run the FMP notebooks, one needs to install Python, Jupyter, and additional Python packages. In **Part B**, we introduce the Anaconda Python distribution with its package and environment manager, which allows for quickly installing, running, and updating the required software packages. Furthermore, we provide a file which specifies an environment called `FMP`. This environment comprises all packages (specified by name and version number) needed for the FMP notebooks. Giving a step-by-step description, we explain how to use Anaconda to set up this environment.

3.1.3 Multimedia

One notebook of **Part B** gives a short overview of how to integrate multimedia objects (in particular, audio, image, and video objects) into a Jupyter notebook. Rather than being comprehensive, we only give a selection of possibilities as used in the other parts of the FMP notebooks. In particular, we discuss two alternatives: a direct integration of images, video, and audio elements using HTML tags as well as an integration using the Python module `IPython.display`.

3.1.4 Python

In the FMP notebooks, we use Python as the programming language. The reason for this choice is that Python is a open-source general-purpose language, which is widely used in scientific computing and offers plenty of resources in data sciences and machine learning. Furthermore, being a beginner-friendly language, it suits the didactic orientation of the FMP notebooks well. **Part B** contains a short introduction to Python summarizing the most important data types, control structures, and functions as occurring in later parts of the FMP notebooks. One of our design principles is to keep the required programming skills at an elementary level. Furthermore, one finds code examples that illustrate how to create appealing figures, process audio files, and program interactive plots.

3.1.5 Numba

As one side topic, we also give a short introduction to the Python package `Numba`, which offers an open source just-in-time (JIT) compiler that translates a subset of Python code into fast machine code. Even though not crucial from a functionality point of view, this package can be used to significantly speed up (sometimes a factor of 100) some of the implementations offered by the FMP notebooks.

3.1.6 Further Topics and Summary

Further topics covered by **Part B** are descriptions of relevant Python libraries, some basic information of the version control system Git, and links to tools that are helpful for music processing and MIR.

In summary, with having the notebooks of **Part B**, our goal is to make the FMP notebooks self-contained (at least, to a high degree). Rather than trying to be comprehensive, we give useful and instructive code examples that become relevant in the other parts. Furthermore, **Part B** also motivates and documents how the FMP notebooks were created.

3.2 Music Processing and MIR Scenarios

The main music processing and MIR topics covered by the FMP notebooks are organized in eight parts, which follow the eight chapters of the textbook on Fundamentals of Music Processing [13]. The notebooks include introductions for each MIR task, provide important mathematical definitions, and describe computational approaches in detail. One primary purpose of the FMP notebooks is to provide audio-visual material as well as Python code examples that implement the computational approaches described before. Additionally, the FMP notebooks provide code that allows a user to experiment with parameters and to gain an understanding of the computed results by suitable visualizations and sonifications. These functionalities also make it easily possible to input different music examples and to generate figures and illustrations that can be used in lectures and scientific articles. This way, the FMP notebooks complement and go beyond the textbook [13], where one finds a more mathematically oriented approach to MIR. In the following, we summarize the main content of the music processing and MIR scenarios covered by the FMP notebooks.

Part 1 (Music Representations). Musical information can be represented in many different ways. In this part, we cover three widely used music representations: sheet music, symbolic, and audio representations. Besides introducing basic terminology that is used throughout the following FMP notebooks, we provide Python code to study musical and acoustic properties of audio signals including aspects

such as frequency, pitch, dynamics, and timbre. For example, there are code snippets for comparing different tuning systems (equal-tempered, Pythagorean, harmonic series) and for generating Shepard tones.

Part 2 (Fourier Analysis of Signals). In this part, we approach the Fourier transform (used as the main signal processing tool in these notebooks) from various perspectives. We provide code to better understand complex numbers and exponential functions, which form the basis for the discrete Fourier transform (DFT). Also, the fast Fourier transform (FFT)—an algorithm of great beauty and high practical relevance—is covered in theory and practice. As another important topic, we discuss the short-time Fourier transform (STFT). In this context, we address issues such as sampling, padding, and axis conventions—issues that are often neglected in theory—from a practical perspective.

Part 3 (Music Synchronization). The objective of music synchronization is to temporally align different versions of the same underlying piece of music. Considering this scenario, we provide code examples for generating chroma-based music features, which capture properties that are related to harmony and melody. In this context, we also address issues of high practical relevance including tuning, logarithmic compression, as well as spectral and temporal resolution—aspects that have a significant influence on the features' properties. Furthermore, we study an alignment technique known as dynamic time warping (DTW), a concept that is applicable for the analysis of general time series. For its efficient computation, we discuss an algorithm based on dynamic programming—a widely used method for solving a complex problem by breaking it down into a collection of simpler subproblems.

Part 4 (Music Structure Analysis). In this part, we address a central and well-researched area within MIR known as music structure analysis. Given a music recording, the objective is to identify critical structural elements and to segment the recording according to these elements. Within this scenario, we discuss fundamental segmentation principles based on repetitions, homogeneity, and novelty—principles that also apply to other types of multimedia beyond music. In particular, we provide code for generating, visualizing, and understanding self-similarity matrices and for modifying their structural properties using a variety of enhancement strategies. The notebooks also cover classical approaches for novelty detection and audio thumbnailing. Finally, we introduce scape plot representations and demonstrate how this concept can be used to generate beautiful visualizations of time-dependent properties in a compact and hierarchical way.

Part 5 (Chord Recognition). Another essential and long-studied MIR task is the analysis of harmonic properties of a piece of music by determining an explicit progression of chords from a given audio recording—a task often referred to as automatic chord recognition. Within this scenario, we first discuss some basic theory of harmony including concepts such as intervals, chords, and scales. To better understand these musical concepts, the notebooks provide code for generating and interacting with suitable sound examples. Furthermore, we introduce a simple baseline system for chord recognition based on a template-based matching procedure. This system is then extended by hidden Markov models (HMMs)—a concept of central importance for the analysis of temporal patterns in time-dependent data streams including speech, gestures, and music. Besides algorithmic aspects and their implementation, we use the chord recognition scenario to illustrate the importance of feature design choices and the effect of temporal smoothing strategies. Such issues become of crucial importance when comparing, understanding, and exploring the potential of more involved chord recognition systems (e. g., based on deep learning).

Part 6 (Tempo and Beat Tracking). Tempo and beat are fundamental properties of music. In this part, we introduce the basic ideas on how to extract tempo-related information from audio recordings. A first task, known as onset detection, aims at locating note onset information by detecting changes in energy and spectral content. The notebooks not only introduce the theory but also provide code for implementing and comparing different onset detectors. To derive tempo and beat information, note onset candidates are analyzed concerning quasiperiodic patterns. This second step leads us to the study of general methods for local periodicity analysis of time series. In particular, we introduce two conceptually different methods: one based on Fourier analysis and the other one based on autocorrelation. Furthermore, the notebooks provide code for visualizing time-tempo representations, which deepen the understanding of musical and algorithmic aspects. Finally, the FMP notebooks cover fundamental procedures for predominant local pulse estimation and global beat tracking.

Part 7 (Content-Based Audio Retrieval). A central topic in MIR is concerned with the development of search engines that enable users to explore music collections in a flexible and intuitive way. In this part, we discuss audio retrieval strategies that follow the query-by-example paradigm: given an audio query, the task is to retrieve all documents that are somehow similar or related to the query. Within this scenario, we discuss the issue of specificity, which refers to the degree of similarity between the query and the database documents. First, we deal with the problem of audio identification (a retrieval task of high specificity), where the objective is to identify the particular audio recording that is the source of the query. In particular, we introduce the main ideas of an audio identification system based on spectral peaks—a technique used in many commercial applications such as Shazam [21]. Then, the notebooks cover two related retrieval tasks of lower specificity referred to as audio matching and version identification, where the goal is to identify recordings with performance variations and other versions (e. g., cover songs). The main goals of the notebooks are to provide code for baseline systems and for gaining a better understanding of

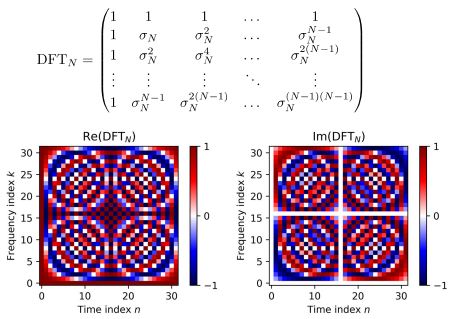


Figure 2. The matrix DFT_N and a visualization of its real and imaginary parts for the case $N = 32$.

the practical requirements of the different retrieval tasks.

Part 8 (Musically Informed Audio Decomposition). In the final part on audio decomposition, the notebooks cover challenging research directions that are related to source separation. Within this wide research area, we consider three subproblems: harmonic–percussive separation, main melody extraction, and score-informed audio decomposition. Within these scenarios, the notebooks offer detailed explanations and implementations of essential techniques including instantaneous frequency estimation, fundamental frequency (F0) estimation, spectrogram inversion, and nonnegative matrix factorization (NMF). These techniques are useful for a variety of general multimedia processing tasks beyond source separation and music processing. Besides algorithmic and computational aspects, we again encounter in this part of the notebooks a variety of acoustic and musical properties of audio recordings. Providing tools and instructive scenarios for gaining a good understanding of such properties is a central and overarching objective of the FMP notebooks.

4. EXAMPLES

In this section, we give some short examples that illustrate some of the educational aspects of the FMP notebooks.

We start with a classical signal processing topic. Given a discrete signal $x = (x(0), x(1), \ldots, x(N-1))^\top \in \mathbb{R}^N$ of length N, the discrete Fourier transform (DFT) is defined by $X(k) := \sum_{n=0}^{N-1} x(n) \exp(-2\pi i k n/N)$ for $k \in [0 : N-1]$. The vector $X \in \mathbb{C}^N$ can be interpreted as frequency representation of the time-domain signal x. The FMP notebooks approach the DFT in various ways, including the usage of inner products and their geometric interpretation. Defining the complex number $\sigma_N := \exp(-2\pi i/N)$, Figure 2 shows the matrix DFT_N (given by $\mathrm{DFT}_N(n,k) = \sigma_N^{nk}$ for $n, k \in [0 : N-1]$) along with a visualization of its real and imaginary parts. Furthermore, the notebooks explain how to evaluate the DFT efficiently using the fast Fourier transform (FFT). The general idea of the FMP notebooks is not to shy away from mathematics. Instead, the notebooks provide rigorous introductions to the theory, which are interleaved with code examples that further explain, implement, and visualize abstract concepts.

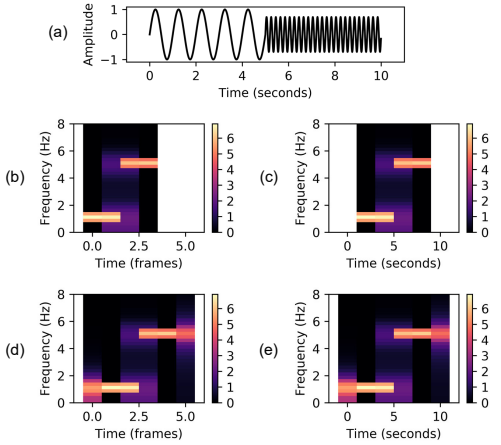


Figure 3. Time-domain signal (a) and its STFT without padding (b/c) and with zero-padding (d/e).

To recover time information hidden in the Fourier domain, the main idea of the short-time Fourier transform (STFT) is to consider only small sections of the signal. These sections are obtained by multiplying shifted versions of a window function with the original signal and by computing a Fourier transform for each of the resulting windowed signals. This results in a sequence of spectral vectors, also called frames. In practice, the correct physical interpretation of discrete objects such as samples, frames, and spectral coefficients can be tricky. Also, there are many different conventions when applying windowing. Figure 3 shows a signal (two subsequent sinusoids of 1 Hz and 5 Hz, respectively) and its STFT. In the FMP notebooks, we explain how to correctly interpret discrete parameters, discuss different windowing conventions (including padding), and show how to correctly visualize feature representations (taking a centric view).

For Western music, one often uses a twelve-tone equal-tempered scale, where the 12 pitch classes correspond to the twelve chroma values $\{\mathrm{C}, \mathrm{C}^\sharp, \mathrm{D}, \mathrm{D}^\sharp, \ldots, \mathrm{B}\}$. Aggregating all spectral information that relates to a given pitch class into a single coefficient, a spectrogram can be transformed in a chromagram, see Figure 4. This example also demonstrates how to generate accurate and visually appealing figures, which can be a tricky and time-consuming effort. In the FMP notebooks, we give numerous examples on how to enhance, place, and align figure elements. In Figure 4, for example, a waveform (given in samples) and a chromagram (given in frames) are visually aligned using a common time axis (given in seconds). Furthermore, using an adapted colormap enhances essential structures in the chromagram. Finally, the size and the placement of the three subplots are controlled using a grid structure.

Such chromagram representations, which particularly capture harmonic and melodic properties of an audio recording, have turned out to be a powerful tool for various MIR tasks. One such task is music synchronization, where the objective is to automatically link different versions of the same piece of music. Figure 5 shows a synchronization result when aligning two different recordings of the be-

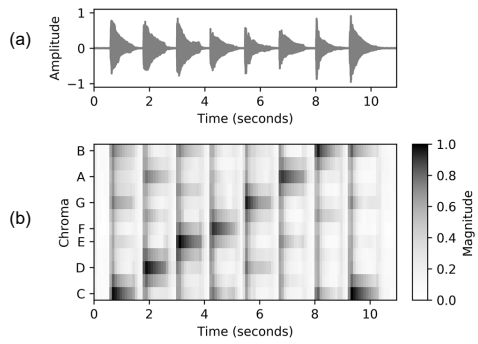


Figure 4. Waveform (a) of a C-major scale and resulting chromagram (b).

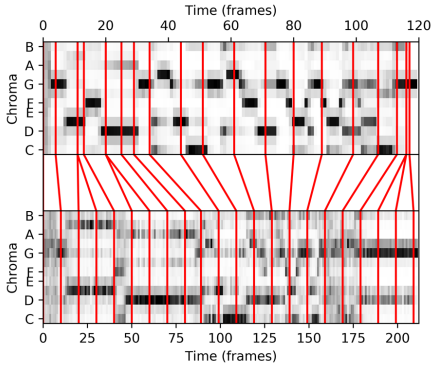


Figure 5. Chromagram representations of two different recordings of the beginning of Beethoven's Fifth Symphony and resulting synchronization result (aligned time positions are indicated by red lines).

ginning of Beethoven's Fifth Symphony. Using a chroma-based alignment procedure based on dynamic time warping as an example approach, the FMP notebooks provide a detailed treatment of music synchronization including a musically informed motivation, algorithmic descriptions, implementation details, graphical representations, and application scenarios.

Chromagrams are also commonly used representations for tasks such as structure analysis and chord recognition. Figure 6 shows the annotated score as well as a chord recognition result obtained from an audio recording of the The Beatles' song "Let It Be." The FMP notebooks not only describe and implement computational approaches, but also discuss the results in a musically informed fashion by looking at real-world music and audio examples. Furthermore, common evaluation measures such as precision, recall, and F-measure are introduced including a discussion of their benefits and limitations within concrete MIR scenarios.

In our final example, we consider a task that is often referred to as harmonic–percussive separation (HPS), where the goal is to decompose a given audio signal into two parts: one consisting of the harmonic and another one of the percussive events of the original signal [4]. Since this task is very instructive from an educational point of view, it was included in the FMP notebooks. First, the task is suited to reflect on acoustic qualities of sound sources, see

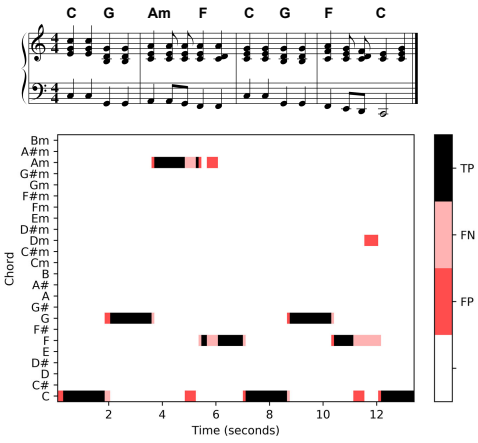


Figure 6. Chord recognition result for the first measures of The Beatles' song "Let It Be."

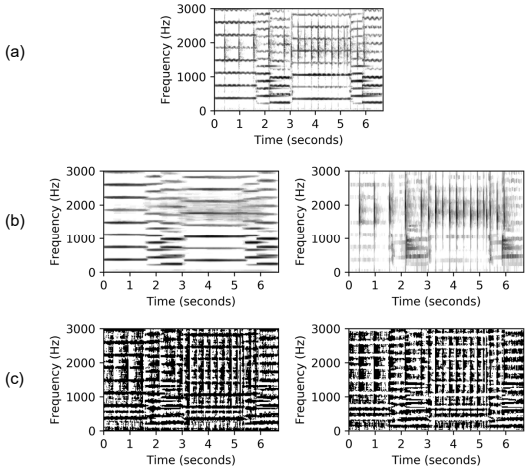


Figure 7. Spectrogram (a) of a recording consisting of violin sounds (harmonic components) superimposed with castanet clicks (percussive components). Furthermore, the figure shows filtered spectrograms (b) and derived binary masks (c).

Figure 7a. Second, the HPS approach discussed involves fundamental (vertical and horizontal) filtering techniques applied to a spectrogram representation, see Figure 7b. Third, it involves spectral masking techniques related to Wiener filtering, see Figure 7c. Finally, one requires signal reconstruction techniques by inverting a modified STFT. The FMP notebooks deal with these central topics using the HPS scenario as motivation and illustration.

5. CONCLUSIONS

The FMP notebooks put together a package for studying central MIR tasks providing detailed explanations, mathematical details, code examples, instructive sound examples, and illustrative figures within a unifying framework. While going hand in hand with existing toolboxes such as `librosa` [12], the FMP notebooks complement existing MIR resources with the aim to bridge the gap between theory and practice and by providing educational material useful for teaching and learning fundamentals of music processing.

Acknowledgments: We thank Stefan Balke, Sebastian Rosenzweig, Christof Weiß, and all other contributors for helping us with programming and testing. This work was supported by the German Research Foundation (MU 2686/10-1, DFG MU 2686/11-1, MU 2686/12-1). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

6. REFERENCES

[1] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: A new Python audio and music signal processing library. In *Proceedings of the ACM International Conference on Multimedia (ACM-MM)*, pages 1174–1178, Amsterdam, The Netherlands, 2016.

[2] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, José R. Zapata, and Xavier Serra. Essentia: An audio analysis library for music information retrieval. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 493–498, Curitiba, Brazil, 2013.

[3] Jonathan Driedger and Meinard Müller. TSM Toolbox: MATLAB implementations of time-scale modification algorithms. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 249–256, Erlangen, Germany, 2014.

[4] Derry FitzGerald. Harmonic/percussive separation using median filtering. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 246–253, Graz, Austria, September 2010.

[5] Peter Grosche and Meinard Müller. Tempogram Toolbox: MATLAB tempo and pulse analysis of music recordings. In *Late-Breaking and Demo Session of the Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Miami, Florida, USA, October 2011.

[6] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter development team. Jupyter notebooks—a publishing format for reproducible computational workflows. In *Proceedings of the International Conference on Electronic Publishing*, pages 87–90, Göttingen, Germany, 2016.

[7] Olivier Lartillot and Petri Toiviainen. MIR in MATLAB (II): A toolbox for musical feature extraction from audio. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 127–130, Vienna, Austria, 2007.

[8] Alexander Lerch. *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. John Wiley & Sons, 2012.

[9] Matthias Mauch and Sebastian Ewert. The audio degradation toolbox and its application to robustness evaluation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 83–88, Curitiba, Brazil, 2013.

[10] Daniel McEnnis, Cory McKay, Ichiro Fujinaga, and Philippe Depalle. jAudio: A feature extraction library. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 600–603, London, UK, 2005.

[11] Brian McFee, Eric J. Humphrey, and Juan Pablo Bello. A software framework for musical data augmentation. In *Proceedings of International Society for Music Information Retrieval Conference (ISMIR)*, pages 248–254, Málaga, Spain, 2015.

[12] Brian McFee, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. Librosa: Audio and music signal analysis in Python. In *Proceedings the Python Science Conference*, pages 18–25, Austin, Texas, USA, 2015.

[13] Meinard Müller. *Fundamentals of Music Processing*. Springer Verlag, 2015.

[14] Meinard Müller and Sebastian Ewert. Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 215–220, Miami, Florida, USA, 2011.

[15] Meinard Müller, Nanzhu Jiang, and Harald Grohganz. SM Toolbox: MATLAB implementations for computing and enhancing similarity matrices. In *Proceedings of the Audio Engineering Society (AES) Conference on Semantic Audio*, London, UK, 2014.

[16] Meinard Müller, Bryan Pardo, Gautham J. Mysore, and Vesa Välimäki. Recent advances in music signal processing. *IEEE Signal Processing Magazine*, 36(1):17–19, 2019.

[17] Oriol Nieto and Juan Pablo Bello. Systematic exploration of computational music structure research. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 547–553, New York City, USA, 2016.

[18] Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P. W. Ellis. mir_eval: A transparent implementation of common MIR metrics. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 367–372, Taipei, Taiwan, 2014.

[19] Christian Schörkhuber and Anssi P. Klapuri. Constant-Q transform toolbox for music processing. In *Proceedings of the Sound and Music Computing Conference (SMC)*, Barcelona, Spain, 2010.

[20] George Tzanetakis. Music analysis, retrieval and synthesis of audio signals MARSYAS. In *Proceedings of the ACM International Conference on Multimedia (ACM-MM)*, pages 931–932, Vancouver, British Columbia, Canada, 2009.

[21] Avery Wang. An industrial strength audio search algorithm. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 7–13, Baltimore, Maryland, USA, 2003.

[22] Anna Xambó, Alexander Lerch, and Jason Freeman. Learning to code through MIR. In *Demos and Late Breaking News of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017.

AUTOMATIC ASSESSMENT OF SIGHT-READING EXERCISES

Jiawen Huang
Center for Music Technology
Georgia Institute of Technology
Atlanta, Georgia
jhuang448@gatech.edu

Alexander Lerch
Center for Music Technology
Georgia Institute of Technology
Atlanta, Georgia
alexander.lerch@gatech.edu

ABSTRACT

Sight-reading requires a musician to decode, process, and perform a musical score quasi-instantaneously and without rehearsal. Due to the complexity of this task, it is difficult to assess the proficiency of a sight-reading performance, and it is even more challenging to model its human assessment. This study aims at evaluating and identifying effective features for automatic assessment of sight-reading performance. The evaluated set of features comprises task-specific, hand-crafted, and interpretable features designed to represent various aspect of sight-reading performance covering parameters such as intonation, timing, dynamics, and score continuity. The most relevant features are identified by Principal Component Analysis and forward feature selection. For context, the same features are also applied to the assessment of rehearsed student music performances and compared across different assessment categories. The results show potential of automatic assessment models for sight-reading and the relevancy of different features as well as the contribution of different feature groups to different assessment categories.

1. INTRODUCTION

Sight-reading, also known as *prima vista*, describes the task of reading and performing an unknown piece of music from its musical score with little or no preparation. It is a challenge to most students who are learning a musical instrument.

Sight-reading performance reflects the player's ability in different aspects including reading music, applying fingering and playing techniques, and interpreting music in a relatively short time. As an important skill for musicians, sight-reading is often part of school curricula as well as auditions for professional orchestras [6]. The assessment of sight-reading in auditions and teaching environments faces multiple difficulties. While there are efforts to make human assessments comparable and "less subjective," for example by using grading rubrics, the fairness of the assessment can be impacted by bias effects (gender, ethnicity, general appearance, etc.), fatigue effects after hours of listening and assessing, as well as individual preferences and tolerances for various error types. An automatic assessment system can potentially provide objective, repeatable, and unbiased assessments. Thus, it could be helpful both as a tool available to judges to inform their decisions as well as a tutoring system for students providing feedback in individual practice sessions. It can also help understand the important performance parameters of sight-reading assessment and how they compare to the assessment of general (student) music performances.

In this study, we create a prototype and investigate the feasibility of a sight-reading assessment system by designing interpretable features for the task and evaluating the system on a large database of professionally rated recordings. We also inspect commonalities and differences of feature sets for the assessment of sight-reading vs. prepared performances of sheet music. More specifically, we perform feature selection and detailed feature analysis on a score-aligned hand-crafted feature set, identify the most effective features for sight-reading assessment and observe the difference in the assessment ratings of sight-reading and a rehearsed performance.

The paper is structured as follows: the related work on sight-reading assessment is introduced in Sect. 2 and the evaluated features are presented in Sect. 3. Section 4 explains the experiments and discusses the results of the feature analysis. The final Sect. 5 gives concluding remarks and outlines future work.

2. RELATED WORK

2.1 Sight-reading skills and parameters

Sight-reading involves coordination of auditory, visual, spatial, and kinesthetic systems to produce an accurate and musical performance [11]. In sight-reading exercises, multiple layers of visual information are processed simultaneously when reading the score while playing the instrument. Besson et al. has demonstrated distinct processing between melodic and rhythmic information [2]. This indicates that pitch accuracy and rhythmic accuracy can be treated as two independent assessment categories. Elliott found a strong positive relationship between wind instrumentalists' general sight-reading ability and the ability to sight-read rhythm patterns [5]. This suggests that features containing rhythmic information are important for assessing a sight-

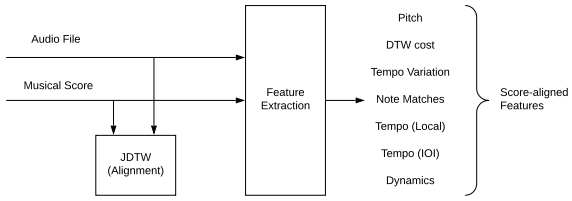


Figure 1. Feature extraction

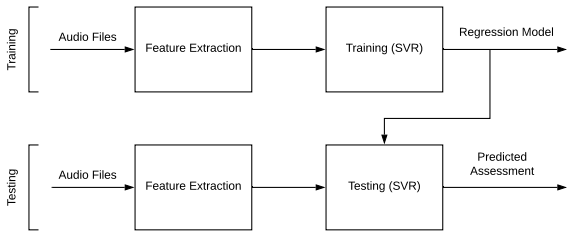


Figure 2. Flow chart of training and testing

reading performance. While intonation, rhythm, and tone quality are typical properties to be assessed, in many cases only an overall rating is given without details on individual properties [1, 4].

2.2 Automatic assessment

There is only a limited number of publications for the automatic assessment of sight-reading. Cheng et al. developed a real-time system for sight-reading evaluation of piano music [4]. The real-time system transcribes the polyphonic music and detects wrong notes. Commercial interest is shown by the existence of systems such as *Sight Reading Practice and Assessment* [1] and *SightReadPlus* [2], which aims at assessing a student playing and tracking the progress of sight-reading.

The automatic assessment of sight-reading has many similarities to the assessment of music performance in general. Therefore, we should expect similar features to be relevant for both tasks and take advantage of the broader spectrum of publications in general performance assessment. Abeßer et al. designed a feature set consisting of 138 features based on the pitch contour of students' vocal and instrumental performances, applied feature selection and used the selected features to train a Support Vector Machine (SVM) [1]. They found that features describing the similarity of score and audio, and the variability of note durations are the most impactful features. Fukuda et al. presented a piano tutoring system which applied non-negative matrix factorization for transcription and DTW for audio-to-score alignment [8]. They basically use, similar to Cheng et al. [4], the number of detected mistakes as core information for performance assessment. Wu et al. proposed assessing a performance independent of the musical score using features based on pitch, amplitude, and rhythm histograms [16]. Vidwans et al. extracted a set of pitch, dynamics, and tempo features after aligning the performance to the score with Dynamic Time Warping (DTW) [15]. Their work is followed by Gururani et al., who investigated the impact of hand-crafted descriptors for the assessment of student alto saxophone technical exercises by feature selection [9]. The results reveal that score-aligned features have a higher correlation with human assessments than score-independent features.

More recently, deep learning methods have been applied to automatic performance assessment [14]. Although deep learning might be a useful tool to achieve better performance for the prediction, the current success of such approaches is often impeded by the available dataset sizes which are often insufficient to train the models properly. A maybe even more important drawback of deep learning is that the interpretability is lost in the hidden layers, so that systems based on deep learning might not be able to give meaningful detailed feedback to a student. This is the main reason why we focus on hand-crafted, knowledge-based features in this study.

3. FEATURE EXTRACTION

3.1 Overview

The flow chart of feature extraction process is shown in Figure 1. [3] Given a recording of a student's sight-reading exercise, the pitch contour is extracted by pYIN [12] from the audio signal (sample rate 44.1 kHz, window and hop size 1024 and 256 samples, respectively). This pitch contour is then aligned to the score of that piece using a modified DTW algorithm which we refer to as Jump-enabled Dynamic Time Warping (JDTW), a DTW variant which can account for repeated score passages. After the alignment, features that capture pitch, rhythmic, and dynamics properties are extracted. The following sections will introduce JDTW, the extracted features, and the inference model.

3.2 Jump-enabled Dynamic Time Warping

Intuitively, we expect the main difference between sight-reading and the performance of a rehearsed piece of music, besides a higher likelihood of errors and more variability in tempo, to be in a higher probability of the student stopping and restarting from a preceding score position after a pause. The frequent occurrence of these jumps has been verified through informal dataset analysis. As standard alignment approaches such as DTW cannot properly handle such jumps, a modification of the DTW algorithm is necessary to properly align the audio sight-reading performance to the score (in our case in MIDI format). Therefore, we propose a Jump-enabled Dynamic Time Warping (JDTW) which is able to handle these repetitions in the students' sight-reading performance. The approach is inspired by Fre-

[1] `http://standardassessmentofsightreading.com`, Last access: 2019/04/10

[2] `http://mymusicta.com/products`, Last access: 2019/04/10

[3] Source code can be accessed at `https://github.com/jhuang448/FBA_code_2019`

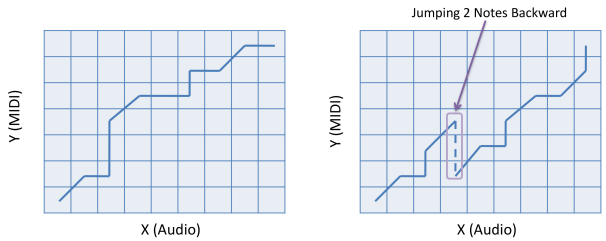


Figure 3. Illustration of paths of index pairs for a sequence X of length N = 9 and a sequence Y of length M = 7. Left: original DTW; Right: JDTW.

merey et al.'s jumpDTW [7] but uses different constraints in terms of potential jump positions and jump lengths.

Dynamic Time Warping (DTW) is a commonly used path finding technique based on dynamic programming to find an optimal alignment between two time series through a pair-wise distance matrix [13]. It has been widely used in speech recognition and musical information retrieval. It only allows sequential alignment, which means that we can neither walk back in a sequence nor jump in time. Given the two sequences $X := (x_1, x_2, ...x_N)$ (audio) of length $N \in \mathbb{N}$ and $Y := (y_1, y_2, ...y_M)$ (midi) of length $M \in \mathbb{N}$, the recursion formula of the accumulated cost matrix D of the classical DTW is as follows:

$$D(n,m) = \min\{D(n-1,m-1), D(n-1,m), D(n,m-1)\} + c(x_n, y_m) \quad (1)$$

for $1 < n \leq N$ and $1 < m \leq M$; $c(x_n, y_m)$ is a measure of distance between x_n and y_m.

The modified accumulated cost matrix D_J for JDTW introduces an additional cost term $J(n,m)$ as follows:

$$D_\text{J}(n,m) = \min\{D_\text{J}(n-1,m-1), D_\text{J}(n-1,m), D_\text{J}(n,m-1), J(n,m)\} + c(x_n, y_m) \quad (2)$$

in which $J(n,m)$ is the minimum accumulated cost for a path jumping to point (n,m):

$$J(n,m) = \begin{cases} \min\limits_{i \leq I}\{D_\text{J}(n-1, m+i) + p\}, & \text{pause before n} \\ \infty, & \text{otherwise} \end{cases} \quad (3)$$

for $1 < n \leq N$ and $1 < m \leq M$, where I is the largest distance in notes allowed for a jump and p is the penalty for jumps. Figure 3 illustrates the paths of the original DTW and the JDTW for an example.

3.2.1 Parametrization and implementation

The adjustment of two JDTW parameters is essential: I, the maximum length of a jump in notes, and p, the penalty of the jump itself. The parametrization with the lowest accumulated cost is found empirically from a simulated validation set of 120 synthesized sound files, leading to the values of $I = 3$ and $p = 3 \cdot \text{mean}(C)$, in which C is the cost matrix between X and Y (meaning that the penalty depends on the average cost).All other DTW-related parametrizations follow standard settings.

Two details are noteworthy in the context of the current implementation: (i) after obtaining the pitch contour from the audio and before computing the alignment, silent frames are temporarily removed from both pitch contour and MIDI sequence, and (ii) the distance between pitch contour x_n and MIDI pitch y_m is computed after tuning frequency adjustment as the octave-independent *wrapped* distance to eliminate pYin's frequent octave errors, however, a small penalty of 1 is added for distances equal or higher than 12 to account for possible octave jumps in the score. After successful application of JDTW, each audio frame is aligned to a note in the MIDI sequence.

Index	Group	Description
1–8	Pitch	Mean and std of pitch dev. (mean, std, max, min)
9–11	DTW cost	Cost of whole path, jumped path and correct path
12–14	Tempo var.	Slope dev., number and distance of jumps
15–16	NIR, NDR	% of silence inserted % of short notes
17–18	Tempo (local)	Inversed tempo per note (mean, std)
19–24	Tempo (IOI)	Crest, bin resolution, skewness, kurtosis, roll-off, power ratio of the IOI histogram
25–32	Dynamics	amplitude envelope and amplitude spikes (mean, std, max, min)

Table 1. Overview of extracted features.

3.3 Feature set

The evaluated feature set can be divided into seven categories: pitch, DTW cost, tempo variation (DTW-based), note matches, tempo (local), tempo (Inter-Onset-Interval-based), and dynamics. Table 1 lists all 32 features explained below with their feature indices.

- **Pitch** ($d = 8$)**:** For each note, the mean and the standard deviation of the pitch deviation from the MIDI pitch is computed. Then, these features are aggregated over the whole performance using mean, standard deviation, maximum, and minimum of each series. The resulting eight features are used to capture intonation accuracy.
- **DTW cost** ($d = 3$)**:** As a result of the alignment, we can compute three cost metrics from the path. The first one is the overall cost of the whole JDTW path. The second cost is the cost of the discarded parts, i.e., the accumulated cost of all the repeated parts except the last run. The third cost is the overall cost of the path ignoring these discarded parts. These three cost features are normalized by the length of the overall

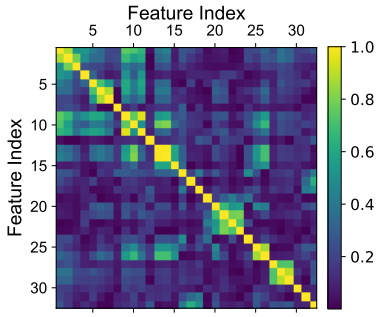


Figure 4. The covariance matrix among features.

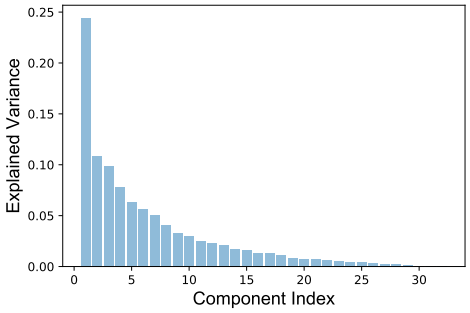


Figure 5. Explained variance for each principle component.

path, and are a measure of pitch similarity between the two sequences.

- **Tempo variation (DTW-based)** ($d = 3$)**:** In addition to the cost-based features, additional features can be extracted from the alignment path. We extract the deviation of the path slope from the diagonal of the matrix, the number of jumps, and the total accumulated distance of jumps.
- **Note matches** ($d = 2$)**:** The Note Insertion Ratio (NIR) is a feature representing additional notes in the student performance, and the Note Deletion Ratio (NDR) represents the missing notes in the performance. As the alignment is performed on pitch contour after removing all the silent frames, these frame have to be inserted back. It is possible that a note is split into multiple notes and that very short (less than 3 frames) notes occur. The NIR is the duration ratio of the inserted silence to the total duration of pitched region. The NDR is the duration ratio of very short notes to the duration of pitched region.
- **Tempo (local)** ($d = 2$)**:** The mean and the standard deviation of the inverse of the tempo per note is an estimate of the overall (inverse) tempo and its variability. For example, an eighth note lasting 1 s results in an inverse local tempo of $\frac{8\,\text{notes}}{1\,\text{s}}$.
- **Tempo (IOI-based)** ($d = 6$)**:** From the histogram of Inter-Onset-Intervals, the crest factor, bin resolution, skewness, kurtosis, roll-off, and the peak power ratio (ratio of the sum of the peak values to the sum of all histogram values) are extracted. These features describe general tempo characteristics.
- **Dynamics** ($d = 8$)**:** For every note, the standard deviation of the envelope as well as amplitude spikes (number of sharp amplitude changes within a note) is computed. Similar to the pitch features, the mean, standard deviation, maximum, and minimum are aggregated over all notes. The resulting eight features are used to capture the dynamic properties of the performance.

4. METHODOLOGY

Our assessment system follows a general machine learning setup as visualized in Figure 2. Our evaluation aims at not only investigating the general feasibility of assessing sight-reading automatically, but also an analysis of which features are most relevant. Furthermore, a general music performance assessment is compared with sight-reading assessment in order to identify similarities and differences between the two tasks.

This section first introduces the dataset used. Then, feature analysis is performed with Principal Component Analysis and forward feature selection. Finally, the performance and features of sight-reading assessment and general music performance assessment is studied.

4.1 Dataset

The dataset used for this study is provided by the Florida Bandmasters Association (FBA). It consists of audio recordings of Florida All-State auditions of middle and high school students in the three years 2013, 2014, and 2015. Each recording consists of exercises such as etudes, scales, and sight reading and provides one expert assessments per exercise in four categories: *musicality*, *note accuracy*, *rhythmic accuracy*, and *tone quality*. For this study we focus on the first three categories. Only a subset of this dataset is used: we are focusing on the sight-reading exercise played by middle school student performers for the instrument Alto Saxophone. The recordings of technical exercise are used to compare sight-reading assessment with the assessment of prepared and rehearsed performances. There are a total of 391 students' audition recordings in the 3 years. Each recording contains technical exercise, sight-reading exercise, and other sections. The total lengths of technical and sight-reading exercise recordings are 192 minutes and 344 minutes, respectively. As the rating scales differ over years and categories (most of the ratings are given within 0–10, others have the ranges 0–5, 0–15, and 0–20), they are all linearly mapped to our target range $[0, 1]$.

The musical score of the sight-reading exercise has been transcribed manually after reviewing multiple highly-rated performances from the three years.

4.2 Principal Component Analysis

Principal Components Analysis is a method to linearly transform a set of possibly correlated variables into a set of uncorrelated variables (components). For the presented analysis, we will use both the covariance matrix of the features and the PCA loading matrix.

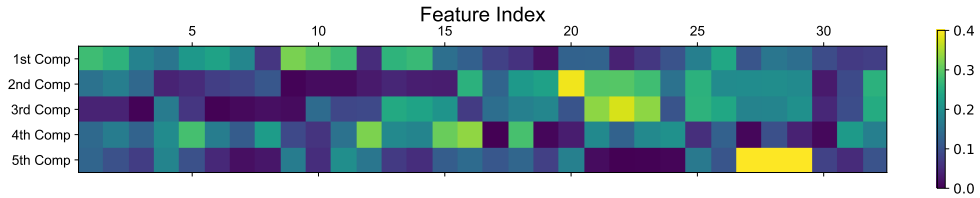


Figure 6. The PCA loading matrix.

The covariance matrix of the features is shown in Figure 4. It can be observed that —unsurprisingly— features are correlated with each other within groups. This is true for pitch features (1–3) and (5–7), DTW cost features (9–11), Tempo variation features (13,14), Tempo (IOI) features (20–23), as well as Dynamics features (25,26) and (27–29). This expected result shows that features within one group carry similar information and verifies that the proposed feature grouping is reasonable.

In addition to high correlation within each feature group, some high correlation is observed across groups. The pitch features (1–7), for instance, are highly correlated with DTW cost features (9–11). This is the case because the DTW-cost features are the accumulated difference between the pitch being played and the reference pitch. The cost of the jumped path (10) is highly correlated with number and distance of jumps (13,14). All of these three features are a measure of the amount of jumps in the performance. The std of the amplitude envelope (26) is also correlated with the jump features. One possible reason for this is that a high number of pauses and jumps might significantly impact the amplitude variation. Other feature correlations are less interpretable; for example, the correlation between min of amplitude spikes (32) and mean of the inverse local tempo (17) is not easily explained.

Figure 5 displays the explained variance by principal components. The eigenvalue of the first component is considerably higher than that of the following components. The first five components explain 60% of the total variance. The loading matrix, shown for these first five components in Figure 6, indicates that the first component is mostly a combination of pitch features (1–7) and the pitch-related DTW cost features (9–11). Both the second and the third component are combinations of rhythmic IOI features with the second focusing on tempo (20) and the third component mostly describing tempo variation (21–23). While the interpretation of the fourth component is difficult, the fifth component clearly represents dynamics (27–29).

4.3 Inference

A SVM Regression model is trained using the extracted features. As a linear kernel gave comparable results to an RBF kernel, the linear kernel was chosen for sake of simplicity. Libsvm [3] is used as implementation.

4.4 Forward Feature Selection

While the PCA gives us insights into feature correlation and which features contribute most to explaining the variance in the feature set, it is of limited use in deciding which features contribute most to the assessment task. In order to identify these, we apply forward feature selection [10]. As this selection approach 'wraps' the target regression algorithm, the selected features will be task-relevant. Forward feature selection is performed on the SVR model with 5-fold cross-validation. The used metric to evaluate success is the R-squared value, which is a common metric for the evaluation of regression systems.

The result of the selection process is a list of features ordered according to their relevance for the task. The indices of the first 10 selected features for each assessment category are listed in Table 2. This table also compares the selected feature sets for sight reading with the sets for a rehearsed student performance. The R-squared results depending on the number of selected features, comparing the sight-reading exercise with the technical exercise, are shown in Figure 7.

4.4.1 Discussion

Figure 7 shows that the R-squared value starts to converge after about 10 iterations of the feature selection. The highest R-squared for musicality and rhythmic accuracy is higher for the practiced performance, while that for note accuracy is higher for sight-reading.

Of the selected features listed in Table 2, two of the dynamics features (25,26) rank high for both practiced performance and sight-reading for all three assessment categories. These two features are the mean and std of the amplitude standard deviation per note. Apparently, the steadiness of loudness plays an important role in assessing the performance.

Looking closer into the Note Accuracy row of Table 2,

Category	Practiced	Sight-reading
Musicality	25,16,21,15,5, 26,11,1,6,18	25,6,32,15,29, 7,26,5,24,23
Note Acc.	9,20,17,3,28, 14,25,21,26,23	26,6,32,15,7, 23,2,9,3,11
Rhythm Acc.	25,21,16,13,26, 18,5,11,2,1	25,6,32,23,15, 29,31,8,20,1

Table 2. The first 10 selected features in forward feature selection. Colors represent different feature groups: **cyan** for pitch, **purple** for DTW cost, **grey** for tempo variance, **apricot** for note matches, **orange** for tempo (local), **pink** for tempo (IOI) and **green** for dynamics.

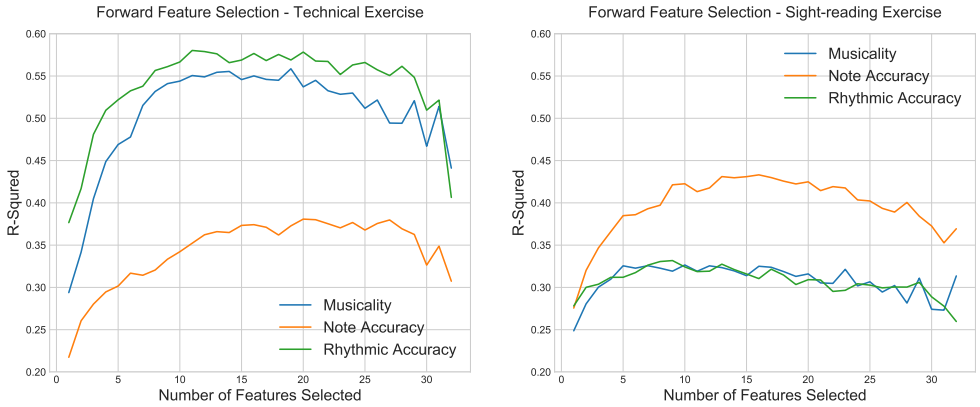


Figure 7. The R-squared curves in feature selection.

we can observe that six of the ten selected features (2,3,6,7,9,11) for sight-reading are features which contribute highly to the first (pitch-related) PCA component. This is not the case for technical exercise, indicating that the pitch features contain more relevant information for sight-reading than for practiced performance. This is also indicated by feature 6 ranking highly in all three assessment categories for sight-reading but not for practiced performance. This feature is one of the aggregated features (standard deviation of absolute differences between played pitch and reference pitch) and is thus a measure of pitch steadiness.

For the assessment categories Musicality and Rhythmic Accuracy, more dynamics features are selected for sight-reading exercise than for the practiced performance. The reason for this might be a different expectation for the two exercises. It might be that, either due to the low complexity of the score or little time for preparation, a dynamically steady performance is preferred by the judges.

During feature selection, the R-squared curve reaches its maximum at about 10–20 iterations and drops dramatically when nearly all the features are selected. This is unexpected behavior for an SVM. A possible reason may be that the dataset is not large enough to train an SVR with all the features or that there might be some 'misleading' features in the feature set.

According to the results above, the automatic assessment of sight-reading is even more challenging than assessing a practiced performance, which performs in the range that we expect (compare [9]) but not so well that it could be considered solved. The higher R-squared for Note Accuracy indicates that our features, especially the intonation features, model this category better for sight-reading than for technical exercise. The low R-squared values for Musicality and Rhythmic Accuracy indicate that we essentially cannot model the human assessments either due to irrelevant features or noisy ground truths. It means that the judges assess the two kinds of exercises differently for these categories and that our regression model fails to capture the information important for sight-reading.

5. CONCLUSION

We presented a feature set of 32 hand-crafted features for the assessment of sight-reading and evaluated them for middle school alto saxophone performances. The feature analysis included PCA and forward feature selection based on the R-squared of the output from an SVR. We can identify the relevant assessment dimensions in the first few principal components and find that the assessment of sight-reading in general is highly influenced by dynamics, and that the assessment of Note Accuracy is mostly focused on pitch-related features. Judging from the absolute results, we can see that the automatic assessment of sight-reading is still an unsolved problem and that the presented features can model a human assessment only imperfectly. In order to be usable in a realistic scenario, we need to either identify additional, more relevant features or move towards state-of-the-art, uninterpretable feature learning solutions. As compared to a practiced and prepared performance, we can identify some commonalities and some differences in the set of relevant features, but the most striking difference is the gap of model performance between assessment categories. Further work is needed to identify where the cause for this gap can be found.

It is likely that rehearsed and sight-reading exercises do not share the same assessment criteria even if the categories are named identically. The performance, as imperfect as it might be, is not assessed by score deviations alone, so that our feature might not represent all critical factors. An additional complication is that in our dataset, we only have the assessment from one judge for each performance. The effect of possible subjectivity and uncertainty makes are complicated task even more challenging. More effort is needed to be able to explain the logic behind the assessment given by judges with quantitative and interpretable indicators before they can be used in music education.

6. ACKNOWLEDGEMENT

We would like to thank the Florida Bandmasters Association (FBA) for providing the dataset used in this work.

7. REFERENCES

[1] J. Abeßer, J. Hasselhorn, A. Lehmann C. Dittmar, and S. Grollmisch. Automatic quality assessment of vocal and instrumental performances of ninth-grade and tenth-grade pupils. In *Proc. of the International Symp. on Computer Music Multidisciplinary Research (CMMR)*, Marseille, 2013.

[2] M. Besson and F. Faïta. An event-related potential (erp) study of musical expectancy: Comparison of musicians with nonmusicians. *Journal of Experimental Psychology: Human Perception and Performance*, 21(6):1278, 1995.

[3] C.C. Chang and C.J. Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3), 2011.

[4] C.C. Cheng, D.J. Hu, and L.K. Saul. Nonnegative matrix factorization for real time musical analysis and sight-reading evaluation. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing*, Las Vegas, 2008. IEEE.

[5] C.A. Elliott. The relationships among instrumental sight-reading ability and seven selected predictor variables. *Journal of Research in Music Education*, 30(1):5–14, 1982.

[6] A.L.P. Farley. *The Relationship Between Musicians' Internal Pulse and Rhythmic Sight-Reading*. PhD thesis, University of Washington, 2014.

[7] C. Fremerey, M. Müller, and M. Clausen. Handling repeats and jumps in score-performance synchronization. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, 2010.

[8] T. Fukuda, Y. Ikemiya, K. Itoyama, and K. Yoshii. A score-informed piano tutoring system with mistake detection and score simplification. In *Proc. of the Sound and Music Computing Conference (SMC)*, Maynooth, 2015.

[9] S. Gururani, A. Pati, C.W. Wu, and A. Lerch. Analysis of Objective Descriptors for Music Performance Assessment. In *International Conference on Music Perception and Cognition (ICMPC)*, Toronto, 2018.

[10] I. Guyon and A. Elisseefi. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, 3(7-8):1157–1182, October 2003.

[11] C.M. Hayward and J. Eastlund Gromko. Relationships among music sight-reading and technical proficiency, spatial visualization, and aural discrimination. *Journal of Research in Music Education*, 57(1):26–36, 2009.

[12] M. Mauch and S. Dixon. pyin: A fundamental frequency estimator using probabilistic threshold distributions. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 659–663, Florence, 2014. IEEE.

[13] M. Müller. Dynamic Time Warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer, Berlin, Heidelberg, 2007.

[14] A. Pati, S. Gururani, and A. Lerch. Assessment of Student Music Performances Using Deep Neural Networks. *Applied Sciences*, 8(4):507, March 2018.

[15] A. Vidwans, S. Gururani, C.W. Wu, V. Subramanian, R.V. Swaminathan, and A. Lerch. Objective descriptors for the assessment of student music performances. In *Proc. of the AES Conference on Semantic Audio*, Erlangen, 2017. AES.

[16] C.W. Wu, S. Gururani, C. Laguna, A. Pati, A. Vidwans, and A. Lerch. Towards the Objective Assessment of Music Performances. In *Proc. of the International Conference on Music Perception and Cognition (ICMPC)*, pages 99–103, San Francisco, 2016.

SUPERVISED SYMBOLIC MUSIC STYLE TRANSLATION USING SYNTHETIC DATA

Ondřej Cífka **Umut Şimşekli** **Gaël Richard**
LTCI, Télécom Paris, Institut Polytechnique de Paris
{ondrej.cifka, umut.simsekli, gael.richard}@telecom-paris.fr

ABSTRACT

Research on style transfer and domain translation has clearly demonstrated the ability of deep learning-based algorithms to manipulate images in terms of artistic style. More recently, several attempts have been made to extend such approaches to music (both symbolic and audio) in order to enable transforming musical style in a similar manner. In this study, we focus on *symbolic music* with the goal of altering the 'style' of a piece while keeping its original 'content'. As opposed to the current methods, which are inherently restricted to be unsupervised due to the lack of 'aligned' data (i.e. the same musical piece played in multiple styles), we develop the first fully *supervised* algorithm for this task. At the core of our approach lies a *synthetic data* generation scheme which allows us to produce virtually unlimited amounts of aligned data, and hence avoid the above issue. In view of this data generation scheme, we propose an encoder-decoder model for translating symbolic music *accompaniments* between a number of different styles. Our experiments show that our models, although trained entirely on synthetic data, are capable of producing musically meaningful accompaniments even for real (non-synthetic) MIDI recordings.

1. INTRODUCTION

Artistic style transfer has become a well-established topic in the computer vision literature and is becoming of increasing interest in other areas of computer science, especially music and natural language processing. More generally, we are dealing with a family of *style transformation* tasks, where the goal is to alter the *style* of a piece of data (e.g., an image, a musical piece, a document) while preserving – to some extent – its *content*. In the music domain, a solution to these problems would have exciting industrial applications, not only as a way to generate new music automatically (as an alternative to fully automatic music composition, which still seems to be a distant goal), but also as a tool for music creators, allowing them to easily incorporate new styles and ideas into their work.

In computer vision, the most popular task in this direction is *style transfer*, where the algorithm has two inputs: the 'content' image to transform and a 'style' image, bearing the style that we wish to impose on (or *transfer* to) the content image. On the other hand, work done on music so far has mostly focused on a different task, which we refer to as *style translation*. Contrary to style transfer, only the 'content' input is given, and the goal is to render it in a target style which is known in advance and usually *learned* from a large set of examples. Note that although this second task is often also referred to as 'style transfer' in the context of music and text generation, we claim that this conflicts with how the term is traditionally understood [11,13,38], and that the term 'translation' is more appropriate and in line with other prior work [17, 24, 28, 40].

The focus of our work is on the latter task, and more specifically, on *accompaniment style translation* for *symbolic music*. In particular, given a piece of music in a symbolic representation, our goal is to generate a new accompaniment for it in a different arrangement style while preserving the original harmonic structure. Even though our approach is generic, to narrow down our scope, we focus on generating bass and piano tracks.

A major difficulty of the music style translation task is that there are no publicly available 'aligned' or 'parallel' datasets (containing examples of the same music played in different styles). As a result, recent works closely related to ours [4, 5] have adopted unsupervised learning frameworks – variational autoencoders (VAE) [19] and CycleGANs [40] – and applied them to genre-labeled datasets. However, these extensions to symbolic music have not yet permitted to obtain results as compelling as those on images [22, 40], text [20, 39], and music audio [28].

In this study, we adopt a different strategy to overcome the lack of aligned data, which is to *synthesize* it. Synthetic training data has proven useful for music information retrieval tasks such as chord recognition [21] and fundamental frequency estimation [25, 32], and is also popular for tasks like semantic segmentation in computer vision [30, 36]. In our case, synthetic data opens up the possibility for supervised learning techniques known from the machine translation field. Moreover, it allows us to work with fine-grained style labels, as opposed to genre labels, which may be too vague or ambiguous for such purposes.

Our main contributions are as follows:

- We propose a supervised, end-to-end neural model for symbolic music style translation, along with a

training data generation scheme.

- Our model is able to translate into a large number of different styles by conditioning a single decoder on the target style. To our knowledge, this is the first time this technique has been applied to music translation with some success.
- To evaluate the performance of our model, we propose an objective metric of music style similarity.
- We show that an approach to music style translation based entirely on synthetic data is viable and generalizes well to more 'natural' inputs, even in unrelated styles.

We believe that our approach will foster new directions in this line of research; some of these will be briefly discussed in the conclusion. The source code of our system, built using TensorFlow, is available online. [1]

2. RELATED WORK

The work performed so far in the area of music style transformation is relatively small in volume but fairly diverse, since, as noted in [8], the transformations can work with different music representations as well as on different conceptual levels.

To our knowledge, the only work on music style transfer – in the original sense, as discussed in the introduction – has been done on audio. Some approaches [9, 35] combine signal decomposition techniques with *musaicing* [41] (a form of concatenative synthesis). In [14], the authors attempt to transfer 'sound textures' from a recording by means of techniques adapted from image style transfer, but without specific focus on the musical aspects. In both cases, the transformation is largely limited to timbre.

The problem of unsupervised music audio translation is tackled in [28], where the authors train a neural network to translate between a number of domains. For symbolic music, style translation is studied in [4, 5], adapting unsupervised learning techniques from computer vision. A different approach is proposed in [23], consisting in training a model on the target style only and then using pseudo-Gibbs sampling to transform a given piece of music.

Finally, we should mention more 'constrained' problems from the symbolic music domain which can also be framed as style translation tasks, e.g. (re-)harmonization [16,29] and expressive performance generation [12,24,37].

3. SYNTHETIC DATA GENERATION

Since we are in a supervised setting, our approach requires a large amount of *paired* examples where each pair consists of one musical fragment arranged in two different styles. Given that no such dataset is currently available, we created a synthetic one, generated using RealBand from the Band-in-a-Box (BIAB) software package [2].

First, we downloaded chord charts of around 3.5K songs in the BIAB format from a popular online archive [3]. We used BIAB to generate arrangements of these songs in different styles and filtered the resulting MIDI files to keep only those in $\frac{4}{4}$ or $\frac{12}{8}$ time. [2] We then chopped those files into segments of 8 bars, splitting notes that overlap segment boundaries.

We selected a total of 70 styles from the '0 MIDI' and '1 MIDI' style packs included in Band-in-a-Box 2018, representing a wide variety of popular music genres. Each style contains up to 5 accompaniment tracks (drums, bass, piano, guitar, strings). [3] We generated each song in 3 randomly picked styles, providing $2 \times \binom{3}{2} = 6$ training pairs per segment, or around 658K training examples in total. An example of a possible training pair is shown in Fig. 1.

In all experiments, we used 2,809 songs for training, 46 songs as a validation set and 46 songs for evaluation, each in 3 examples in different styles. The song names, along with the styles used for each song, are included in the supplementary material [7].

4. PROPOSED MODEL

We propose an architecture based on RNN encoder-decoder sequence-to-sequence models with attention [1], commonly employed in machine translation and other areas of natural language processing. This choice is motivated by the successes of RNNs on symbolic music generation [10, 15, 33, 34] and by the ability of the attention mechanism to condition the generation on arbitrary input data without a prior alignment.

Our model is designed so that it is capable of translating music between a potentially large number of different styles. This is achieved by conditioning the decoder on the target style. An obvious advantage of this design is efficiency: to translate between n styles, we only need to train a single model, compared to n models (one for each target style; possibly with a shared encoder as in [28]) or even $\Theta(n^2)$ models (one for each pair of styles, e.g. [4,5]). Other implications of this choice are investigated in Section 6.2.

On the other hand, to simplify the task and facilitate evaluation, we train a dedicated model for each target instrument track. Our output representation and decoder architecture are chosen accordingly and would not necessarily be suitable for generating several independent tracks.

Input and output representation. A common choice of representation of symbolic non-monophonic music for neural processing is a piano roll. We use a binary-valued piano roll with 128 pitches and 4 columns per beat (quarter note) to encode our input.

For representing the output (and also as an alternative input representation), we opted for a MIDI-like encoding, which – unlike a piano roll – is straightforward to model using an RNN decoder. Specifically, following [33], we encode the music as a sequence of 3 types of events, each with one integer argument:

- `NoteOn(`*`pitch`*`)` : start a new note at the given pitch;

[1] `https://git.io/musicstyle`

[2] The time signature depends on the style as well as on the song itself. A song originally in $\frac{4}{4}$ may have a $\frac{12}{8}$ arrangement and vice versa.

[3] These 5 labels are not always accurate; for example, some styles have two guitar tracks, one of which is labeled as piano.

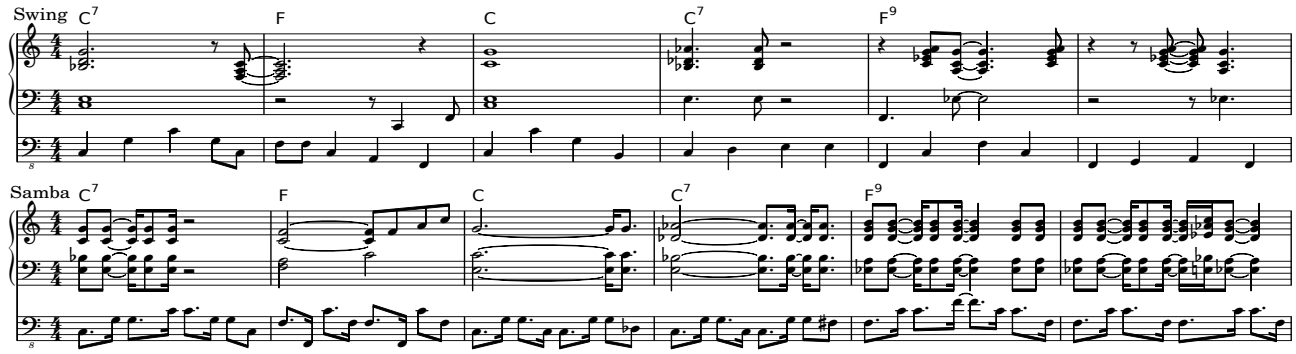
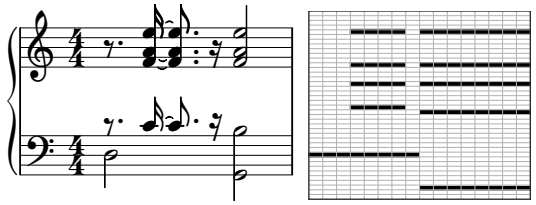


Figure 1: Six bars of an accompaniment (piano and bass) for a 12-bar blues, generated using BIAB in a 'jazz swing' style (top) and a 'samba' style (bottom). The timing is only approximate. The input chord sequence is displayed at the top.



```
NoteOn(50) TimeShift(9) NoteOn(60) NoteOn(65)
NoteOn(69) NoteOn(76) TimeShift(12) NoteOff(60)
NoteOff(65) NoteOff(69) NoteOff(76) TimeShift(3)
NoteOff(50) NoteOn(43) NoteOn(59) NoteOn(65)
NoteOn(69) NoteOn(76) TimeShift(24) NoteOff(All)
```

Figure 2: A bar of music, represented as a piano roll (top right) and as a sequence of 20 event tokens (bottom).

- `NoteOff(pitch)`: end the note at the given pitch;
- `TimeShift(delta)`: move forward in time by the specified amount, measured in 12ths of a beat.

`NoteOn` and `NoteOff` take values in the range 0–127, whereas `TimeShift` is within 1–24.[4] In contrast to [33], our representation is tempo-invariant and we do not model dynamics. Fig. 2 illustrates both representations.

Model architecture and training. The proposed model consists of an encoder and a decoder; the former serves to compute a dense representation of the input, while the latter generates the output event sequence, conditioned on the encoded input and the target style.

The architecture of the encoder depends on the type of input representation:

- If the input is a piano roll, we use a two-layer convolutional network (CNN), followed by a bidirectional RNN with a gated recurrent unit (GRU) [6]. The CNN serves to compress the input, resulting in a sequence of 1280-dimensional vectors with 2 vectors per bar. The bidirectional GRU then adds the ability to incorporate information from a wider context.
- If the input is a sequence of tokens, we use an embedding layer, also followed by a bidirectional GRU.

We refer to the two variants of the model as 'roll2seq' and 'seq2seq', respectively.

The decoder is also implemented using a GRU, conditioned on the target style and equipped with a feed-forward

[4] When encoding the piano track, we compress the sequences by also including a `NoteOff(All)` event which ends all currently active notes.

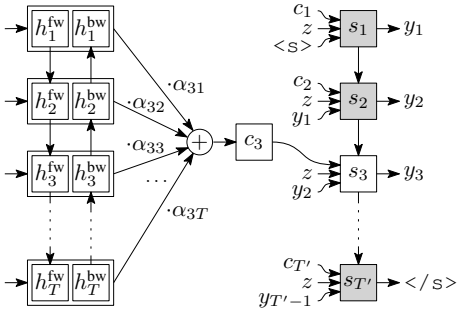


Figure 3: The attention-based decoder. During the i-th decoding step (here $i = 3$), a set of coefficients α_{ij} is computed and used to weight the encoder states $h_j = [h_j^{\text{fw}}, h_j^{\text{bw}}]$ to obtain the context vector c_i, which in turn is used as input for the decoder cell to compute the next state, s_i.

attention mechanism [1] acting on the encoder outputs. More precisely, as illustrated in Fig. 3, the i-th decoder state s_i is computed as

$$s_i = \text{GRU}([c_i, W^{\text{s}}z, W^{\text{e}}y_{i-1}], s_{i-1}),$$

where $[\cdot]$ denotes concatenation, z and y_{i-1} respectively denote the one-hot encoded representations of the target style and the previous output event, $W^{\text{s}}, W^{\text{e}}$ are the corresponding embedding matrices, and c_i is the context vector. The latter is a weighted average of the encoder outputs, computed by the attention mechanism. The purpose of attention is to provide an *alignment* between the encoder and decoder states. The need for this alignment arises from the fact that the positions in the output sequence are not linear in time (due to the chosen encoding), and the decoder therefore needs to be able to move its focus flexibly over the input. For a complete description of attention, see [1].

The training pipeline is portrayed in Fig. 4b. Each training example consists of a song segment x in one style (the source style) along with the corresponding segment y in a different style (z, the target style). We train the model by minimizing the loss on y while passing x to the encoder and conditioning the decoder on z.

The models are trained using Adam [18] with learning rate decay and with early stopping on the development set. Our configuration files with complete hyperparameter set-

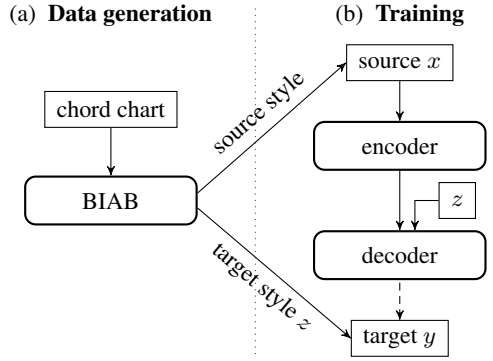


Figure 4: A scheme of the training pipeline. (a) We use BIAB to generate each song in different arrangement styles (see Section 3). (b) The model is trained to predict the target-style segment y given a source segment x and the target style z (see Section 4).

tings are included with the source code.

Once the model is trained, we perform style translation using greedy decoding, i.e. by taking the most likely output token at every step (and using that as input in the next step). We also explored random sampling with different softmax temperatures, but found that this leads to a higher number of errors (i.e. invalid sequences or incorrect timing) and does not significantly improve the quality of the outputs.

5. EVALUATION METRICS

When evaluating a style transformation, we need to consider two complementary criteria: how well the transformed music fits the desired style (*style fit*) and how much content it retains from the original (*content preservation*). Note that it is trivial (but useless) to achieve perfect results on either of these two criteria *alone*, so it is essential to evaluate both of them.

In this section, we describe 'objective', automatically computed metrics for both criteria. Even though we believe these metrics are sound and well-motivated, we acknowledge the limitations of automatic metrics in general and encourage the reader to listen to the provided example outputs [7] to get a real sense of their quality.

Content preservation. We use a content preservation metric similar to the one proposed by [23], computed by correlating the chroma representation of the generated segment with that of the corresponding segment in the source style. This is motivated by the fact that we expect the output to follow the same sequence of chords as the input. More precisely, we compute chroma features for each segment at a rate of 12 frames per beat and smooth each of them using an averaging filter with a window size of 2 beats (24 frames) and a stride of 1 beat (12 frames). Finally, we calculate the average frame-wise cosine similarity between the two sets of chroma features.

Style fit. In some of the recent music style transformation works [4,5], the quality of a transformation is measured by means of a binary style classifier trained on a pair of styles. However, the merit of such evaluation is limited, since a high classifier score merely demonstrates that the output has some of the distinguishing features of the target style, and not necessarily that it actually fits the style. For this reason, we aim for a more interpretable metric of style fit.

As observed by [16, 26, 31], musical style is well captured in pairwise statistics between neighboring events. Drawing inspiration from the features proposed in [26], we devise a key- and time-invariant style representation which we call the *style profile*.

To compute the style profile, we consider all pairs of note onsets less than 4 beats apart and at most 20 semitones apart, and record the time difference and interval for each pair. In other words, we define the following multiset of ordered pairs:

$$\mathcal{S} = \{(t_b - t_a,\ p_b - p_a) \,|\, a, b \in \text{notes},\ a \neq b,\\ 0 \leq t_b - t_a < 4,\ |p_b - p_a| \leq 20\},$$

where t_x is the onset time of the note x (measured in fractional beats) and p_x is its MIDI note number. We then obtain the style profile as a normalized 2D histogram of $\mathcal{S}$ with 6 bins per beat and one bin per semitone, and flatten it to get a 984-dimensional vector.

Finally, to quantify the style fit of a particular set of outputs, we compute their style profile and measure its cosine similarity to a reference profile. Note that an 8-bar segment may not be sufficient to obtain a reliable style profile; instead, we always aggregate the statistics over a number of segments. In particular, we put forward two variants of the style fit metric, obtained as follows:

(a) Compute a style profile aggregated over all outputs of a model in a given target style and measure its cosine similarity to the reference.

(b) Compute a style profile for each translated song separately and measure its cosine similarity to the reference. We report the mean and standard deviation over all songs.

We refer to (a) and (b) as 'macro-style' and 'song-style', respectively. In both cases, the reference style profile is extracted from the training set, separately for each track.

While we do not claim that this metric is able to distinguish between broad style categories (such as genres), yet it can definitely capture the differences and similarities between specific 'grooves', which makes it well-suited for our purpose. This is illustrated in Fig. 5, showing the pairwise similarities between the profiles of the bass tracks of different BIAB styles, with clearly visible clusters of jazz, rock or country styles.

6. EXPERIMENTAL RESULTS

In our experiments, we focus on generating the bass and piano tracks, and we train a dedicated model for each of them. For each track, we consider two scenarios: generating the track given only the corresponding source track (BASS→BASS, PIANO→PIANO), and using all non-drum accompaniment tracks from the input (ALL→BASS, ALL→PIANO).

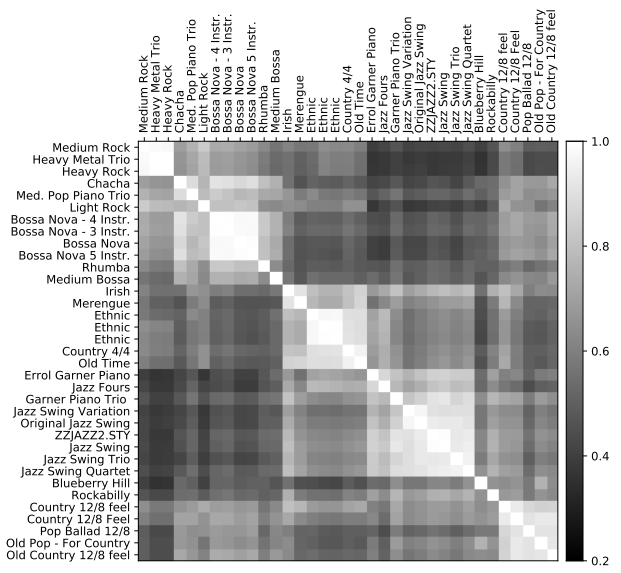


Figure 5: Pairwise cosine similarities of selected style profiles computed on training bass tracks. The styles are ordered based on a hierarchical clustering of the profiles.

For BASS→BASS, we compare the seq2seq and roll2seq architectures defined in Section 4. For all other pairs, where the input is non-monophonic, we only employ roll2seq, since the sequential representation grows disproportionately in length in these cases and the computational cost of the attention mechanism becomes too heavy.

We evaluate our models on our synthetic test set generated by BIAB and on the Bodhidharma MIDI dataset [27]. The latter is a diverse collection of 950 MIDI recordings annotated with genre labels. We filtered and pre-processed the dataset in the same way as the synthetic test set and we extracted the bass and piano tracks. [5]

We also made extensive attempts to train the recent models of [4, 5, 23] on our data using the source code published by the authors, but unfortunately without success. This has prevented us from comparing these models with our proposal. Nonetheless, the provided outputs [7] can serve as a basis for perceptual comparison.

6.1 Evaluation

For a comprehensive evaluation of each model, we translated all inputs to all 70 styles and calculated the content preservation and style fit metrics. The results (averaged) are presented in Fig. 6.

We provide two baselines for each track (bass and piano): 'source', which is simply the same track before the translation, and 'reference', which is a track generated by BIAB based on the chord chart (only available for the synthetic test set). As expected, the style fit is low for the source track (measured with respect to the target style) and close to 1 for the reference track. Our models' outputs generally do not fit the target style as perfectly as the reference does, but still score high compared to the source.

[5] To form the bass track, we retrieve all notes assigned to any Bass instrument. For the piano track, we use the Piano and Organ classes.

As for content preservation, we can notice that the reference value is quite low (0.78 for BASS and 0.79 for PIANO). This should not be too surprising, since we are comparing accompaniments in two different styles, which might have different pitch-class distributions; moreover, there is some random harmonic variation within each style (see e.g. bars 5–6 in Fig. 1). The results achieved by our models on the synthetic test set are very close to the reference. To illustrate the value range of the metric, we provide the results obtained by a 'randomized' baseline (shown as 'random' in Fig. 6), where we randomly permuted the reference segments for each style (obtaining a reference with the correct style, but the wrong content). The resulting value is very low (0.16 for BASS and 0.31 for PIANO) compared both to the true reference and to our models, indicating that the metric is useful and the models are performing well.

On Bodhidharma, content preservation is generally weaker than on the synthetic test set. One interpretation can be that the encoder simply fails to extract the content information accurately, since it was trained on a different domain. However, we also find that the models often make timing errors on Bodhidharma inputs, leading to misalignment between the input and the output, which may also cause the content preservation metric to drop.

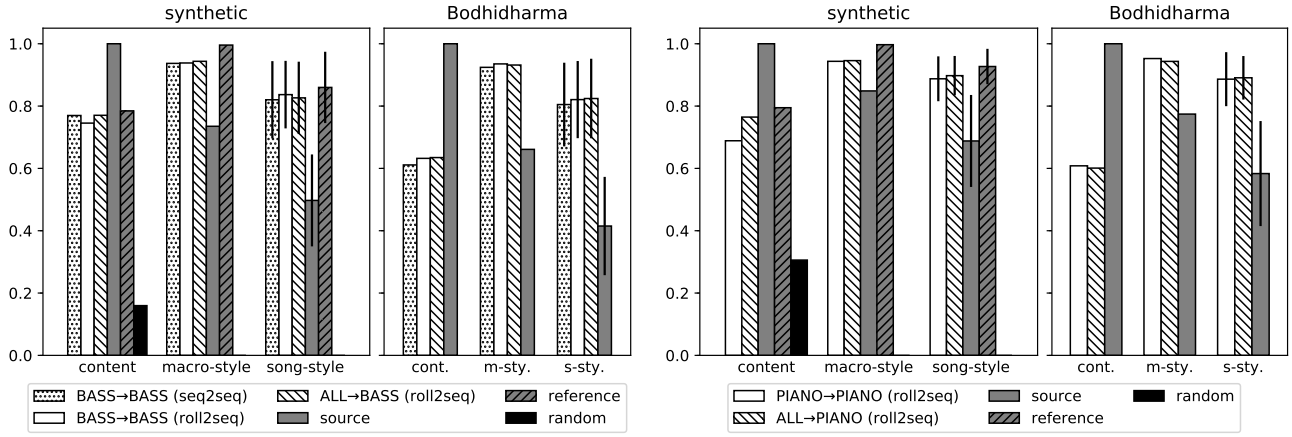
On the other hand, the style fit on Bodhidharma is close to the results on the synthetic test set (and not consistently lower or higher), and the difference to 'source' (i.e. the corresponding input track) is more marked, perhaps reflecting a higher style variability in the Bodhidharma data.

Upon listening, we clearly observe that the outputs are musical and seem to both fit the target style and follow the harmonic structure of the inputs. Besides, even though the piano and the bass tracks are generated independently, they sound surprisingly coherent. However, as mentioned above, we also observe occasional timing errors (especially in heavily syncopated grooves), which become more prominent when the bass and piano tracks are combined. A potential remedy for this issue would be to modify the encoding to make it more robust, e.g. by representing the timing in a beat-aware manner.

We also note that the single-track models output harmonically incorrect notes more often than the ALL models; this is expected, since their *input* is less harmonically rich. This effect is clearly audible (especially in BASS, where important scale degrees are often missing in the input), but cannot be captured by the content preservation metric, which is computed against the same input.

6.2 Comparison with a single-pair model

All models presented so far were trained on music in 70 different styles, as opposed to a single style pair. To investigate the effect of this choice, we picked a pair of fairly dissimilar styles – ZZJAZZSW ('Jazz Swing Variation') and TWIST ('Twist Style', categorized as 'Lite Pop') – and generated a new training, validation and test set with each song rendered in these two styles only. To increase the amount of data, we performed this twice for each song (with different results), obtaining $2 \times 2 = 4$ training pairs



(a) BASS (b) PIANO

Figure 6: Evaluation results on content preservation and style fit. 'Source' is the original track (bass or piano), 'reference' is a track generated by BIAB in the target style and 'random' is a random permutation of the references. For 'song-style', we plot the mean and the standard deviation over all songs and target styles.

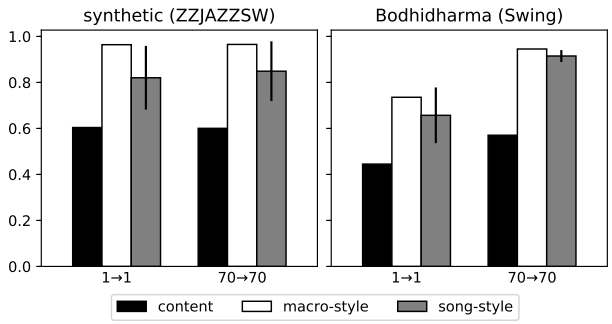


Figure 7: Comparison of a single-style-pair model (1→1) and a full model (70→70) on the ZZJAZZSW→TWIST style pair.

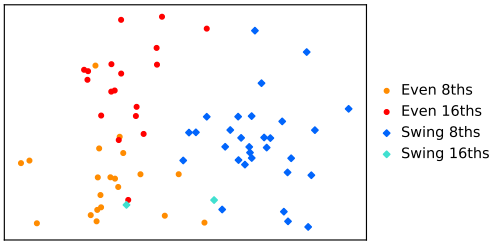


Figure 8: Style embeddings learned by the ALL→PIANO model, labeled with 'feel' annotations provided by BIAB. Dimensionality reduction was performed using linear discriminant analysis (LDA) with the feel labels as targets.

per segment.

We used this new dataset to train single-style-pair versions of all models (in the ZZJAZZSW→TWIST direction only), preserving the original architectures except for the conditioning on the target style. We compare these '1→1' models to the full versions (70→70) on two sets of inputs:

- the synthetic test set in the ZZJAZZSW style;
- the 'Swing' section of Bodhidharma (23 songs).

In Fig. 7, we show the results for the two variants of the ALL→BASS model. While the performance on the synthetic data seems to be the same, the scores of the 1→1 model drop considerably on the Bodhidharma data, suggesting that the model is overfitted to the 'synthetic' swing style. On the other hand, the performance of the 70→70 model stays high, showing that training on many different styles helped the model generalize to real swing.

6.3 Style embedding analysis

Neural representation spaces are often found to exhibit a meaningful geometry, and our learned style embedding space is no exception. As an example, Fig. 8 shows a projection of the embeddings labeled by the 'feel' of each style, with 'even' and 'swing' feel styles being clearly separated. We include more plots in the supplementary material and also make available an interactive visualization. [6]

7. CONCLUSION

In this study, we focused on symbolic music accompaniment style translation. As opposed to the current methods, which are inherently restricted to be unsupervised due to the lack of aligned datasets, we developed the first fully supervised algorithm for this task, leveraging the power of synthetic training data. Our experiments show that our models are capable of producing musically meaningful accompaniments even for real MIDI recordings.

We believe that these results point to interesting research directions. First, synthetic data seem to be an excellent resource for music style translation, and could be used as a starting point even for unsupervised learning, allowing to validate a given approach before moving on to more challenging, unaligned datasets. Second, our supervised approach could be used to address more general music transformation tasks, and we are already working on an extension in this direction.

[6] https://bit.ly/2G5Jgnq

8. ACKNOWLEDGEMENT

This research is supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 765068 (MIP-Frontiers) and by the French National Research Agency (ANR) as a part of the FBIMATRIX (ANR-16-CE23-0014) project.

9. REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[2] Band-in-a-Box. PG Music Inc. `https://www.pgmusic.com/`.

[3] Band-in-a-Box (BIAB) file archive. `https://groups.yahoo.com/group/Band-in-a-Box-Files/`.

[4] Gino Brunner, Andres Konrad, Yuyi Wang, and Roger Wattenhofer. MIDI-VAE: Modeling dynamics and instrumentation of music with applications to style transfer. In *ISMIR*, 2018.

[5] Gino Brunner, Yuyi Wang, Roger Wattenhofer, and Sumu Zhao. Symbolic music genre transfer with CycleGAN. *CoRR*, abs/1809.07575, 2018.

[6] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[7] Ondřej Cífka. Supplementary material: Supervised symbolic music style translation using synthetic data. Zenodo, 2019. `https://doi.org/10.5281/zenodo.3250606`.

[8] Shuqi Dai, Zheng Zhang, and Gus Xia. Music style transfer: A position paper. *CoRR*, abs/1803.06841, 2018.

[9] Jonathan Driedger, Thomas Prätzlich, and Meinard Müller. Let it Bee – towards NMF-inspired audio mosaicing. In *ISMIR*, 2015.

[10] Douglas Eck and Jürgen Schmidhuber. Finding temporal structure in music: blues improvisation with LSTM recurrent networks. In *NNSP*, 2002.

[11] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, 2001.

[12] Sebastian Flossmann and Gerhard Widmer. Toward a multilevel model of expressive piano performance. In *ISPS*, 2011.

[13] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, pages 2414–2423, 2016.

[14] Eric Grinstein, Ngoc Q. K. Duong, Alexey Ozerov, and Patrick Pérez. Audio style transfer. In *ICASSP*, pages 586–590, 2018.

[15] Gaëtan Hadjeres and François Pachet. DeepBach: a steerable model for Bach chorales generation. In *ICML*, 2017.

[16] Gaëtan Hadjeres, Jason Sakellariou, and François Pachet. Style imitation and chord invention in polyphonic music with exponential families. *CoRR*, abs/1609.05152, 2016.

[17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, pages 5967–5976, 2017.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[19] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. *CoRR*, abs/1312.6114, 2014.

[20] Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. *CoRR*, abs/1711.00043, 2017.

[21] Kyogu Lee and Malcolm Slaney. Acoustic chord transcription and key extraction from audio using key-dependent hmms trained on synthesized audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 16:291–301, 2008.

[22] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NIPS*, 2017.

[23] Wei-Tsung Lu and Li Su. Transferring the style of homophonic music using recurrent neural networks and autoregressive models. In *ISMIR*, 2018.

[24] Iman Malik and Carl Henrik Ek. Neural translation of musical style. *CoRR*, abs/1708.03535, 2017.

[25] Matthias Mauch and Simon Dixon. PYIN: A fundamental frequency estimator using probabilistic threshold distributions. In *ICASSP*, pages 659–663, 2014.

[26] Cory McKay. Automatic genre classification of MIDI recordings. M.A. Thesis, McGill University, 2004.

[27] Cory McKay and Ichiro Fujinaga. The Bodhidharma system and the results of the MIREX 2005 symbolic genre classification contest. In *ISMIR*, 2005.

[28] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. A universal music translation network. *CoRR*, abs/1805.07848, 2018.

[29] François Pachet and Pierre Roy. Non-conformant harmonization: the Real Book in the style of Take 6. In *ICCC*, 2014.

[30] Germán Ros, Laura Sellart, Joanna Materzynska, David Vázquez, and Antonio M. López. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, pages 3234–3243, 2016.

[31] Jason Sakellariou, Francesca Tria, Vittorio Loreto, and François Pachet. Maximum entropy models capture melodic styles. *Scientific Reports*, 2017.

[32] Justin Salamon, Rachel M. Bittner, Jordi Bonada, Juan J. Bosch, Emilia Gómez, and Juan Pablo Bello. An analysis/synthesis framework for automatic F0 annotation of multitrack datasets. In *ISMIR*, 2017.

[33] Ian Simon and Sageev Oore. Performance RNN: Generating music with expressive timing and dynamics. Magenta Blog, 2017. `https://magenta.tensorflow.org/performance-rnn`.

[34] Bob L. Sturm, João Felipe Santos, Oded Ben-Tal, and Iryna Korshunova. Music transcription modelling and composition using deep learning. *CoRR*, abs/1604.08723, 2016.

[35] Christopher J. Tralie. Cover song synthesis by analogy. In *ISMIR*, 2018.

[36] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *CVPR*, pages 4627–4635, 2017.

[37] Gerhard Widmer, Sebastian Flossmann, and Maarten Grachten. YQX plays Chopin. *AI Magazine*, 30:35–48, 2009.

[38] Xuexiang Xie, Feng Tian, and Seah Hock Soon. Feature guided texture synthesis (FGTS) for artistic style transfer. In *DIMEA*, 2007.

[39] Junbo Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, and Yann LeCun. Adversarially regularized autoencoders. In *ICML*, 2018.

[40] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, pages 2242–2251, 2017.

[41] Aymeric Zils and François Pachet. Musical mosaicing. In *COST G-6 Conference on Digital Audio Effects (DAFX-01)*, 2001.

DEEP MUSIC ANALOGY VIA LATENT REPRESENTATION DISENTANGLEMENT

Ruihan Yang[1] **Dingsu Wang**[1] **Ziyu Wang**[1] **Tianyao Chen**[1] **Junyan Jiang**[1,2] **Gus Xia**[1]

[1] Music X Lab, NYU Shanghai [2] Machine Learning Department, Carnegie Mellon University

[1]{ry649,dw1920,zz2417,tc2709,gxia}@nyu.edu, [2]junyanj@cs.cmu.edu

ABSTRACT

Analogy-making is a key method for computer algorithms to generate both natural and creative music pieces. In general, an analogy is made by partially transferring the music abstractions, i.e., high-level representations and their relationships, from one piece to another; however, this procedure requires *disentangling* music representations, which usually takes little effort for musicians but is non-trivial for computers. Three sub-problems arise: extracting latent representations from the observation, disentangling the representations so that each part has a unique semantic interpretation, and mapping the latent representations back to actual music. In this paper, we contribute an explicitly-constrained variational autoencoder (EC^2-VAE) as a unified solution to all three sub-problems. We focus on disentangling the *pitch* and *rhythm* representations of 8-beat music clips conditioned on chords. In producing music analogies, this model helps us to realize the imaginary situation of "*what if*" a piece is composed using a different pitch contour, rhythm pattern, or chord progression by borrowing the representations from other pieces. Finally, we validate the proposed disentanglement method using objective measurements and evaluate the analogy examples by a subjective study.

1 Introduction

For intelligent systems, an effective way to generate high-quality art is to produce analogous versions of existing examples [15]. In general, two systems are analogous if they share common abstractions, i.e., high-level representations and their relationships, which can be revealed by the paired tuples *A : B :: C : D* (often spoken as A is to B as C is to D). For example, the analogy "the hydrogen atom is like our solar system" can be formatted as *Nucleus : Hydrogen atom :: Sun : Solar system*, in which the shared abstraction is "a bigger part is the center of the whole system." For generative algorithms, a clever shortcut is to make analogies by solving the problem of "*A : B :: C : ?*". In the context of music generation, if A is the rhythm pattern of a very lyrical piece B, this analogy can help us realize the imaginary situation of "what if B is composed with a rather rapid and syncopated rhythm C" by preserving the pitch contours and the intrinsic relationship between pitch and rhythm. In the same fashion, other types of "what if" compositions can be created by simply substituting A and C with different aspects of music (e.g., chords, melody, etc.).

A great advantage of *generation via analogy* is the ability to produce both *natural* and *creative* results. Naturalness is achieved by reusing the representations (high-level concepts such as "image style" and "music pitch contour") of human-made examples and the intrinsic relationship between the concepts, while creativity is achieved by recombining the representations in a novel way. However, making meaningful analogies also requires *disentangling* the representations, which is effortless for humans but non-trivial for computers. We already see that making analogies is essentially transferring the abstractions, not the observations — simply copying the notes or samples from one piece to another would only produce a casual re-mix, not an analogous composition [11].

In this paper, we contribute an explicitly-constrained conditional variational autoencoder (EC^2-VAE), a conditional VAE with explicit semantic constraints on intermediate outputs of the network, as an effective tool for learning disentanglement. To be specific, the encoder extracts latent representations from the observations; the semantic constraints disentangle the representations so that each part has a unique interpretation, and the decoder maps the disentangled representations back to actual music while preserving the intrinsic relationship between the representations. In producing analogies, we focus on disentangling and transferring the *pitch* and *rhythm* representations of 8-beat music clips when chords are given as the condition (an extra input) of the model. We show that EC^2-VAE has three desired properties as a generative model. First, the disentanglement is *explicitly coded*, i.e., we can specify which latent dimensions denote which semantic factors in the model structure. Second, the disentanglement does not sacrifice much of the reconstruction. Third, the learning does not require any analogous examples in the training phase, but the model is capable of making analogies in the inference phase. For evaluation, we propose a new metric and conduct a survey. Both objective and subjective evaluations show that our model significantly outperforms the baselines.

2 Related Work

2.1 Generation Via Analogy

The history of generation via analogy can trace back to the studies of non-parametric "image analogies" [15] and "playing Mozart by analogy" using case-based reasoning [29]. With recent breakthroughs in artificial neural networks, we see a leap in the quality of produced analogous examples using deep generative models, including music and image style transfer [7, 13], image-to-image translation [18], attribute arithmetic [3], and voice impersonation [12].

Here, we distinguish between two types of analogy algorithms. In a *broad* sense, an analogy algorithm is any computational method capable of producing analogous versions of existing examples. A common and relatively easy approach is supervised learning, i.e., to directly learn the mapping between analogous items from labeled examples [18, 27]. This approach requires little representation learning but needs a lot of labeling effort. Moreover, supervised analogy does not generalize well. For example, if the training analogous examples are all between lyrical melodies (the source domain) and syncopated melodies (the target domain), it would be difficult to create other rhythmic patterns, much less the manipulation of pitch contours. (Though improvements [1, 21, 32] have been made, weak supervision is still needed to specify the source and target domains.) On the other hand, a *strict* analogy algorithm requires not only learning the representations but also disentangling them, which would allow the model to make domain-free analogies via the manipulation of any disentangled representations. Our approach belongs to this type.

2.2 Representation Learning and Disentanglement

Variational auto-encoders (VAEs) [22] and generative adversarial networks (GANs) [14] are so far the two most popular frameworks for music representation learning. Both use encoders (or discriminators) and decoders (or generators) to build a bi-directional mapping between the distributions of observation x and latent representation z, and both generate new data via sampling from $p(z)$. For music representations, VAEs [2,9,24,30] have been a more successful tool so far compared with GANs [31], and our model is based on the previous study [30].

The motivation of representation disentanglement is to better interpret the latent space generated by VAE, connecting certain parts of z to semantic factors (e.g., age for face images, or rhythm for melody), which would enable a more controllable and interactive generation process. InfoGAN [5] disentangles z by encouraging the mutual information between x and a subset of z. β-VAE [16] and its follow-up studies [4,20,30] imposed various extra constraints and properties on $p(z)$. However, the disentanglement above are still *implicit*, i.e., though the model separates the latent space into subparts, we cannot define their meanings beforehand and have to "check it out" via *latent space traversal* [3]. In contrast, the disentanglement in Style-based GAN [19], Disentangled Sequential Autoencoder [23], and our EC^2-VAE are *explicit*, i.e., the meanings of different parts of z are defined by the model structure, so that the controlled generation is more precise and straightforward. The study Disentangled Sequential Autoencoder [23] is most related to our work and also deals with sequential inputs. Using a partially time-invariant encoder, it can approximately disentangle dynamic and static representations. Our model does not directly constrain z but applies a loss to intermediate outputs associated with latent factors. Such an indirect but explicit constraint enables the model to further disentangle the representation into pitch, rhythm, and any semantic factors whose observation loss can be defined. As far as we know, this is the first disentanglement learning method tailored for music composition.

3 Methodology

In this section, we introduce the data representation and model design in detail. We focus on disentangling the latent representations of pitch and rhythm, the two fundamental aspects of composition, over the duration of 8-beat melodies. All data come from the Nottingham dataset [10], regarding a $\frac{1}{4}$ beat as the shortest unit.

3.1 Data Representation

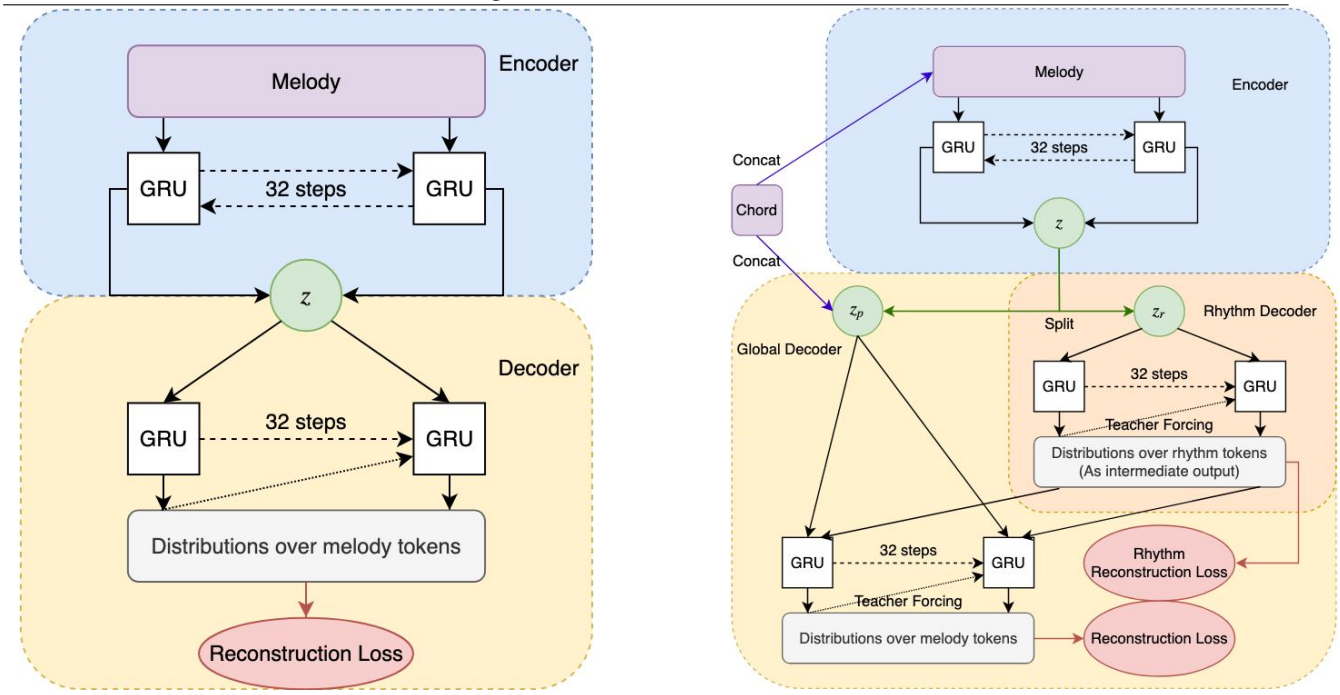
Each 8-beat melody is represented as a sequence of 32 one-hot vectors each with 130 dimensions, where each vector denotes a $\frac{1}{4}$-beat unit. As in [24], the first 128 dimensions denote the *onsets* of MIDI pitches ranging from 0 to 127 with one unit of duration. The 129^{th} dimension is the *holding* state for longer note duration, and the last dimension denotes *rest*. We also designed a rhythm feature to constrain the intermediate output of the network. Each 8-beat rhythm pattern is also represented as a sequence of 32 one-hot vectors. Each vector has 3 dimensions, denoting: an onset of any pitch, a holding state, and rest.

Besides, chords are given as a condition, i.e., an extra input, of the model. The chord condition of each 8-beat melody is represented as a chromagram with equal length, i.e., 32 multi-hot vectors each with 12 dimensions, where each dimension indicates whether a pitch class is activated.

3.2 Model Architecture

Our model design is based on the previous studies of [24, 30], both of which used VAEs to learn the representations of fixed-length melodies. Figure 1 shows a comparison between the model architectures, where Figure 1(a) shows the model designed in [30] and Figure 1(b) shows the model design in this study. We see that both use bi-directional GRUs [6] (or LSTMs [17]) as the encoders (in blue) to map each melody observation to a latent representation z, and both use uni-directional GRUs(LSTMs) (with teacher forcing [26] in the training phrase) as the decoders (in yellow) to reconstruct melodies from z.

The key innovation of our model design is to assign a part of the decoder (in orange) with a specific subtask: to disentangle the latent rhythm representation z_r from the overall z by explicitly encouraging the intermediate output of z_r to match the rhythm feature of the melody. The other



(a) Vanilla sequence VAE.

(b) EC^2-VAE model.

Figure 1: A comparison between vanilla sequence VAE [30] and our model with condition and disentanglement.

part of z is therefore everything but rhythm and interpreted as the latent pitch representation, z_p. Note that this explicitly coded disentanglement technique is quite flexible — *we can use multiple subparts of the decoder to disentangle multiple semantically interpretable factors of z simultaneously* as long as the intermediate outputs of the corresponding latent factors can be defined, and the model shown in Figure 1(b) is the simplest case of this family.

It is also worth noting that the new model uses chords as a condition for both the encoder and decoder. The advantage of chord conditioning is to free z from storing chord-related information. In other words, the pitch information in z is "detrended" by the underlying chord for better encoding and reconstruction. The cost of this design is that we cannot learn a latent distribution of chord progressions.

3.2.1 *Encoder*

A single layer bi-directional GRU with 32 time steps is used to model $Q_\theta(z|x,c)$, where x is the melody input, c is the chord condition, and z is the latent representation. Chord conditions are given by concatenating with the input at each time step.

3.2.2 *Decoder*

The global decoder models $P_\phi(x|z,c)$ by multiple layers of GRUs, each with 32 steps. For disentanglement, we split the latent representation z into two halves z_p and z_r, each being a 128-dimensional vector. As a subpart of the global decoder, the rhythm decoder models $P_{\phi_r}(r(x)|z)$ by a single layer GRU, where $r(x)$ is the rhythm feature of the melody. Meanwhile, the rhythm is concatenated with z_p and chord condition as the input of the rest of the global decoder to reconstruct the melody. We used cross-entropy loss for both rhythm and melody reconstruction. Note that the overall decoder is supposed to learn non-trivial relationships between pitch and rhythm, rather than naively cutting a pitch contour by a rhythm pattern.

3.3 Theoretical Justification of the ELBO Objective with Disentanglement

One concern about representation disentanglement techniques is that they sometimes sacrifice reconstruction power [20]. In this section, we prove that our model does not suffer much of the disentanglement-reconstruction paradox, and the likelihood bound of our model is close to that of the original conditional VAE, and in some cases, equal to it.

Recall the Evidence Lower Bound (ELBO) objective function used by a typical conditional VAE [8] constraint on input sample x with condition c:

$$\begin{aligned}\text{ELBO}(\phi,\theta) &= \mathbb{E}_Q[\log P_\phi(x|z,c)] \\ &\quad - \mathbb{KL}[Q_\theta(z|x,c)||P_\phi(z|c)] \leq \log P_\phi(x|c)\end{aligned}$$

For simplicity, $\mathcal{D}$ denotes $\mathbb{KL}[Q_\theta(z|x,c)||P_\phi(z|c)]$ in the rest of this section. If we see the intermediate rhythm output in Figure 1(b) as hidden variables of the whole network, the new ELBO objective of our model only adds the rhythm reconstruction loss based on the original one, resulting in a lower bound of the original ELBO. Formally,

$$\begin{aligned}&\text{ELBO}^{\text{new}}(\phi,\theta) \\ &= \mathbb{E}_Q[\log P_\phi(x|z,c)] - \mathcal{D} + \mathbb{E}_Q[\log P_{\phi_r}(r(x)|z_r)] \\ &= \text{ELBO}(\phi,\theta) + \mathbb{E}_Q[\log P_{\phi_r}(r(x)|z_r)]\end{aligned}$$

where ϕ_r denotes parameters of the rhythm decoder. Clearly, ELBO$^{\text{new}}$ is a lower bound of the original ELBO because $\mathbb{E}_Q[\log P_{\phi_r}(r(x)|z_r)] \leq 0$.

Moreover, if the rest of global decoder takes the original rhythm rather than the intermediate output of rhythm decoder as the input, the objective can be rewritten as:

$$\begin{aligned}
&\text{ELBO}^{\text{new}}(\phi,\theta)\\
&=\mathbb{E}_Q[\underbrace{\log P_\phi(x|r(x),z_p,c)+\log P_\phi(r(x)|z_r,c)}_{\text{with } x \perp\!\!\!\perp z_r|r(x),c \text{ and } r(x)\perp\!\!\!\perp z_p}]-\mathcal{D}\\
&=\mathbb{E}_Q[\log P_\phi(x,r(x)|z,c)]-\mathcal{D}\\
&=\mathbb{E}_Q[P_\phi(x|z,c)+\log P_\phi(r(x)|x,z,c)]-\mathcal{D}\\
&=\text{ELBO}(\phi,\theta)
\end{aligned}$$

The second equal sign holds for a perfect disentanglement, and the last equal sign holds since $r(x)$ is decided by x, i.e., $P_\phi(r(x)|x,z,c)=1$. In other words, we show that under certain assumptions ELBO$^{\text{new}}$ with disentanglement is identical to the ELBO.

4 Experiments

We present the objective metrics to evaluate the disentanglement in Section 4.1, show several representative examples of generation via analogy in Section 4.2, and use subjective evaluations to rate the artistic aspects of the generated music in Section 4.3.

4.1 Objective Measurements

Upon a successful pitch-rhythm disentanglement, any changes in pitch of the original melody should not affect the latent rhythm representation much, and vice versa. Following this assumption, we developed two measurements to evaluate the disentanglement: 1) Δz after transposition, which is more qualitative, and 2) F-score of an augmentation-based query, which is more quantitative.

4.1.1 Visualizing Δz after transposition

We define F_i as the operation of transposing all the notes by i semitones, and use the L_1-norm to measure the change in z. Figure 2 shows a comparison between $\Sigma|\Delta z_p|$ and $\Sigma|\Delta z_r|$ when we apply F_i to a randomly chosen piece (where $i \in [1,12]$) while keeping the rhythm and underlying chord unchanged.

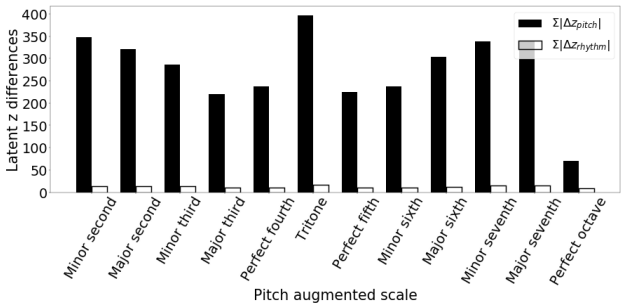


Figure 2: A comparison between Δz_p and Δz_r after transposition.

Here, the black bars stand for $\Sigma|\Delta z_p|$ and the white bars stand for the $\Sigma|\Delta z_r|$. It is conspicuous that when augmenting pitch, the change of z_p is much larger than the change of z_r, which well demonstrates the success of the disentanglement.

It is also worth noting that the change of z_p to a certain extent *reflects human pitch perception*. Given a chord, the change in z_p can be understood as the "burden" (or difficulty) to memorize (or encode) a transposed melody. We see that such burden is large for tritone (very dissonant), relatively small for major third, perfect fourth & fifth (consonant), and very small for perfect octave.

Due to the space limit, we only show the visualization of the latent space when changing the pitch. According to the data representation in Section 3.1, changing the rhythm feature of a melody would inevitably affect the pitch contour, which would lead to complex behavior of the latent space hard to interpret visually. We leave the discussion for future work but will pay more attention to the effect of the rhythm factor in Section 4.3.

4.1.2 F-score of Augmentation-based Query

The explicitly coded disentanglement enables a new evaluation method from an *information-retrieval* perspective. We regard the pitch-rhythm split in z defined by the model structure as the *reference* (the ground truth), the operation of factor-wise data augmentation (keeping the rhythm and only changing pitch randomly, or vice versa) as a *query* in the latent space, and the actual latent dimensions having the largest variance caused by augmentation as the *result set*. In this way, we can quantitatively evaluate our model in terms of precision, recall, and F-score.

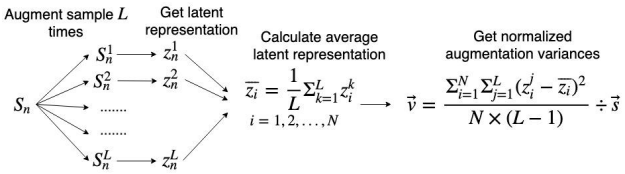


Figure 3: Evaluating the disentanglement by data augmentation.

	Pitch			Rhythm		
	pre.	rec.	F-s.	pre.	rec.	F-s.
EC^2-VAE	**0.88**	**0.88**	**0.88**	**0.80**	**0.80**	**0.80**
Random	0.5	0.5	0.5	0.5	0.5	0.5

Table 1: The evaluation results of pitch- and rhythm-wise augmentation-based query.

Figure 3 shows the detailed query procedure, which is a modification of the evaluation method in [20]. After pitch or rhythm augmentation for each sample, $\vec{v}$ is calculated as the average (across the samples) variance (across augmented versions) of the latent representations, normalized by the total sample variance $\vec{s}$. Then, we choose the first half (128 dimensions) with the largest variances as the result set. This precision, recall and F-score of this augmentation-based query result is shown in Table 1. (Here, precision and recall are identical since the size of the result set equals the dimensionality of z_p and z_r.) As this is the first tailored metric for explicitly coded disentanglement, we use random guess as our baseline.

4.2 Examples of Generation via Analogy

We present several representative "what if" examples by swapping or interpolating the latent representations of different pieces. Throughout this section, we use the following example (shown in Figure 4), an 8-beat melody from the Nottingham Dataset [10] as the source, and the target rhythm or pitch will be borrowed from other pieces. (MIDI demos are available at `https://github.com/cdyrhjohn/Deep-Music-Analogy-Demos`.)

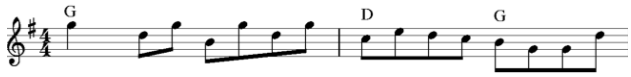
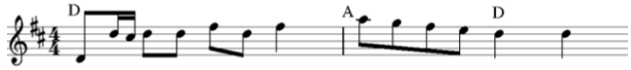


Figure 4: The source melody.

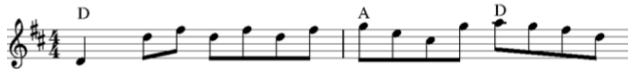
4.2.1 Analogy by replacing z_p

Two examples are presented. In both cases, the latent pitch representation and the chord condition of the source melody are replaced with new ones from other pieces. In other words, the model answers the analogy question: *"source's pitch : source melody :: target's pitch : ?"*

Figure 5 shows the first example, where Figure 5(a) shows the piece from which the pitch and chords are borrowed, and Figure 5(b) shows the generated melody. From Figure 5(a), we see the target melody is in a different key (D major) with a larger pitch range than the source and a big pitch jump in the beginning. From Figure 5(b), we see the generated new melody captures such pitch features while keeping the rhythm of the source unchanged.

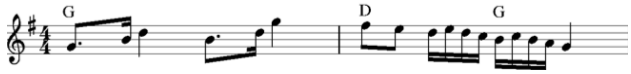


(a) Target's pitch and chord.

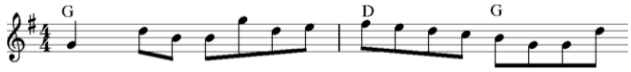


(b) The generated target music, using the pitch and chord from (a) and the rhythm from the source.

Figure 5: The 1st example of analogy via replacing z_p.



(a) Target's pitch and chord.



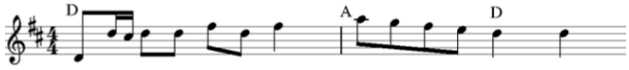
(b) The generated target, using the pitch and chord from (a) and the rhythm from the source.

Figure 6: The 2nd analogy example via replacing z_p.

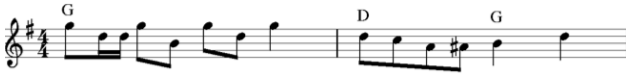
Figure 6 shows another example, whose subplots share the same meanings with the previous one. From Figure 6(a), we see the first measure of the target's melody is a broken chord of Gmaj, while the second measure is the G major scale. From Figure 6(b), we see the generated new melody captures these pitch features. Moreover, it retains the source's rhythm and ignores the dotted eighth and sixteenth notes in Figure 6(a).

4.2.2 Analogy by replacing z_r

Similar to the previous section, this section shows two example answers to the question: "*source's rhythm : source melody :: target's rhythm : ?*" by replacing z_r. Figure 7 shows the first example, where Figure 7(a) contains the new rhythm pattern quite different from the source, and Figure 7(b) is the generated target. We see that Figure 7(b) perfectly inherited the new rhythm pattern and made minor but novel modifications based on the source's pitch.

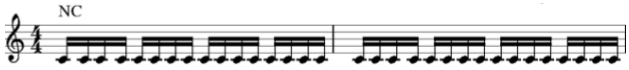


(a) Target's rhythm pattern.

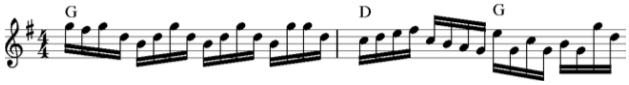


(b) The generated target music, using the rhythm of (a) while keeping source's pitch and chord.

Figure 7: The 1st example of analogy via replacing z_r.



(a) Target's rhythm pattern.

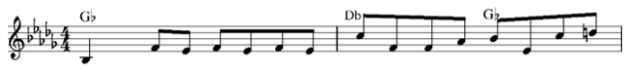


(b) The generated target music, using the rhythm of (a) while keeping source's pitch and chord.

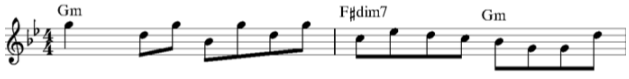
Figure 8: The 2nd analogy example via replacing z_r.

Figure 8 shows a more extreme case, in which Figure 8(a) contains only 16th notes of the same pitch. Again, we see the generated target in Figure 8(b) maintains the source's pitch contour while matching the given rhythm pattern.

4.2.3 Analogy by Replacing Chord



(a) Changing all the chords down a semitone, resulting in the key change from G major to Bb minor.



(b) Changing the key from G major to G minor.

Figure 9: Two examples of replacing the original chord.

Though chord is not our main focus, here we show two analogy examples in Figure 9 to answer "what if" the source melody is composed using some other chord progressions. Figure 9(a) shows an example where the key is Bb minor. An interesting observation is the new melody

contour indeed adds some reasonable modification (e.g. flipping the melody) rather than simply transposing down all the notes. It brings us a little sense of Jazz. Figure 9(b) shows an example where the key is changed from G major to G minor. We see melody also naturally transforms from major mode to minor mode.

4.2.4 *Two-way Pitch-Rhythm Interpolation*

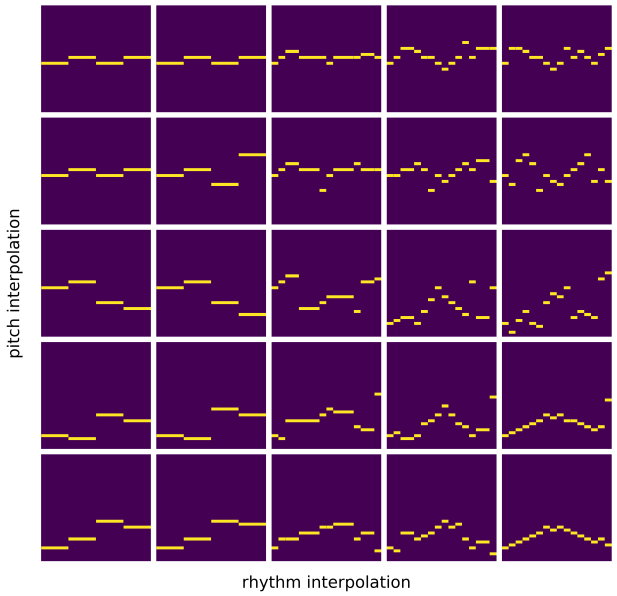


Figure 10: An illustration of two-way interpolation.

The disentanglement also enables a smooth transition from one music to another. Figure 10 shows an example of two-way interpolation, i.e., a traversal over a subspace of the learned latent representations z_r and z_p along 2 axes respectively, while keeping the chord as NC (no chord). Here, each square is a piano-roll of an 8-beat music. The top-left (source) and bottom-right (target) squares are two samples created manually and everything else is generated by interpolation using SLERP [28]. Note that the rhythmic changes are primarily observed moving along the "rhythm interpolation" axis, and likewise for pitch and the vertical "pitch interpolation" axis.

4.3 Subjective Evaluation

Besides objective measurement, we conducted a subjective survey to evaluate the quality of generation via analogy. We focus on changing the rhythm factors of existing music since this operation leads to an easier identification of the source melodies.

Each subject listened to two groups of five pieces each. All the pieces had the same length (64 beats at 120 bpm). Within each group, one piece was an original, human-composed piece from the Nottingham dataset, having a lyrical melody consisting of longer notes. The remaining four pieces were variations upon the original with more rapid rhythms consisting of 8^{th} and 16^{th} notes. Two of the variations were produced in a rule-based fashion by naively cutting the notes in the original into shorter subdivisions, serving as the *baseline*. The other two variations were generated with our EC^2-VAE by merging the z_p of the original piece and the z_r decoded from two pieces having the same rhythm pattern as the baselines but with all notes replaced with "do" (similar to Figure 8(a)). The subjects always listened to the original first, and the order of the variations was randomized. In sum, we compare three versions of music: 1) the original piece, 2) the variation created by the baseline, and, 3) the variation created by our algorithm. The subjects were asked to rate each sample on a 5-point scale from 1 (very low) to 5 (very high) according to three criteria:

1. *Creativity*: how creative the composition is.
2. *Naturalness*: how human-like the composition is.
3. *Overall musicality*.

A total of 30 subjects (16 female and 14 male) participated in the survey. Figure 11 shows the results, where the heights of bars represent means of the ratings the and error bars represent the MSEs computed via within-subject ANOVA [25]. The result shows that our model performs significantly better than the rule-based baseline in terms of creativity and musicality ($p < 0.05$), and marginally better in terms of naturalness. Our proposed method is even comparable to the original music in terms of creativity, but remains behind human composition in terms of the other two criteria.

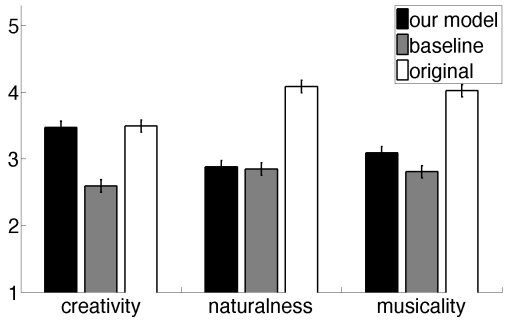


Figure 11: Subjective evaluation results.

5 Conclusion

In conclusion, we contributed an explicitly-constrained conditional variational autoencoder (EC^2-VAE) as an effective disentanglement learning model. This model generates new music via making analogies, i.e., to answer the imaginary situation of "what if" a piece is composed using different pitch contours, rhythm patterns, and chord progressions via replacing or interpolating the disentangled representations. Experimental results showed that the disentanglement is successful and the model is able to generate interesting and musical analogous versions of existing music. We see this study a significant step in music understanding and controlled music generation. The model also has the potential to be generalized to other domains, shedding light on the general scenario of generation via analogy.

6 Acknowledgement

We thank Yun Wang, Zijian Zhou and Roger Dannenberg for the in-depth discussion on music disentanglement and analogy. This work is partially supported by the Eastern Scholar Program of Shanghai.

7 References

[1] Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[2] Gino Brunner, Andres Konrad, Yuyi Wang, and Roger Wattenhofer. Midi-vae: Modeling dynamics and instrumentation of music with applications to style transfer. *arXiv preprint arXiv:1809.07600*, 2018.

[3] Shan Carter and Michael Nielsen. Using artificial intelligence to augment human intelligence. *Distill*, 2(12):e9, 2017.

[4] Ricky TQ Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in vaes. NIPS, 2018.

[5] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[7] Shuqi Dai, Zheng Zhang, and Gus G Xia. Music style transfer: A position paper. *arXiv preprint arXiv:1803.06841*, 2018.

[8] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[9] Philippe Esling, Axel Chemla-Romeu-Santos, and Adrien Bitton. Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR*, pages 23–27, 2018.

[10] E. Foxley. Nottingham database, 2011.

[11] Y Gao. *Towards neural music style transfer*. PhD thesis, Master Thesis, New York University. https://github.com/821760408-sp/the . . . , 2017.

[12] Yang Gao, Rita Singh, and Bhiksha Raj. Voice impersonation using generative adversarial networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2506–2510. IEEE, 2018.

[13] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[15] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001.

[16] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, volume 3, 2017.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[19] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018.

[20] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

[21] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1857–1865. JMLR. org, 2017.

[22] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[23] Yingzhen Li and Stephan Mandt. Disentangled sequential autoencoder. *arXiv preprint arXiv:1803.02991*, 2018.

[24] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. *arXiv preprint arXiv:1803.05428*, 2018.

[25] Henry Scheffe. *The analysis of variance*, volume 72. John Wiley & Sons, 1999.

[26] Nikzad Benny Toomarian and Jacob Barhen. Learning a trajectory using adjoint functions and teacher forcing. *Neural Networks*, 5(3):473–484, 1992.

[27] Christopher J Tralie. Cover song synthesis by analogy. *arXiv preprint arXiv:1806.06347*, 2018.

[28] Alan Watt and Mark Watt. Advanced animation and bendering techniques. 1992.

[29] Gerhard Widmer and Asmir Tobudic. Playing mozart by analogy: Learning multi-level timing and dynamics strategies. *Journal of New Music Research*, 32(3):259–268, 2003.

[30] Ruihan Yang, Tianyao Chen, Yiyi Zhang, and Gus Xia. Inspecting and interacting with meaningful music representations using vae. *arXiv preprint arXiv:1904.08842*, 2019.

[31] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[32] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

QUERY BY VIDEO: CROSS-MODAL MUSIC RETRIEVAL

Bochen Li
University of Rochester
Rochester, NY, USA

Aparna Kumar
Spotify
New York, NY, USA

ABSTRACT

Cross-modal retrieval learns the relationship between the two types of data in a common space so that an input from one modality can retrieve data from a different modality. We focus on modeling the relationship between two highly diverse data, music and real-world videos. We learn cross-modal embeddings using a two-stream network trained with music-video pairs. Each branch takes one modality as the input and it is constrained with emotion tags. Then the constraints allow the cross-modal embeddings to be learned with significantly fewer music-video pairs. To retrieve music for an input video, the trained model ranks tracks in the music database by cross-modal distances to the query video. Quantitative evaluations show high accuracy of audio/video emotion tagging when evaluated on each branch independently and high performance for cross-modal music retrieval. We also present cross-modal music retrieval experiments on Spotify music using user-generated videos from Instagram and Youtube as queries, and subjective evaluations show that the proposed model can retrieve relevant music. We present the music retrieval results at: `http://www.ece.rochester.edu/~bli23/projects/query.html`.

1. INTRODUCTION

Music retrieval has been explored for many cross-domain inputs such as text [27], image [5], location [41], video [32], vocal imitation [42], and sheet music [29]. To our knowledge there are few reports focusing on cross-modal music retrieval given videos from unconstrained sources. With the proliferation of smart phones, people capture short videos to communicate moments from their everyday lives. Learning relationships between music and real-world videos has many applications including novel music query scenarios where a playlist is recommended to fit user's surrounding scenes, or automatically soundtrack selection to complement and enhance visual messages on social media, e.g., Snapchat, Instagram, and Facebook.

Real-world videos can contain any form of video including edited or raw content, and music is an inherently diverse content as well [8]. Thus associating music with such videos is more challenging than common sounds and objects, e.g., barking to a dog [1], which has explicit connection on semantic level. One way to bridge real-world videos and music is via elicited emotions. Previous work addresses this problem after recognizing each modality independently as hand-labeled emotion features [5, 32], but this is not sufficient since hand-labeled emotions (e.g., several human-defined emotion tags) are prone to bias and subjectivity [39], and bottleneck each modality into a limited non-learnable space. Thus no scalable solution has been proposed to query from large music databases. Later a two-stream network structure is proposed for music query by music videos [12], where cross-modal embeddings are learned directly from music-video pairs. However, it requires intensive training on music videos (MV) where the videos were originally created for specific songs, and the music retrieval performances given videos from more varieties of sources are not systematically evaluated.

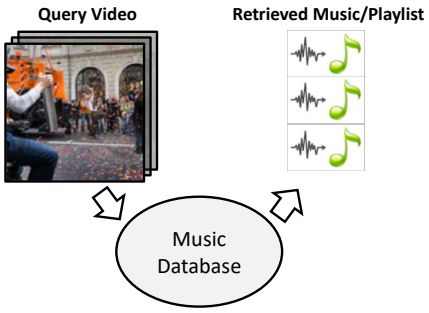


Figure 1. A large music database can be queried by real-world videos from unconstrained sources.

In this paper, we address the music retrieval task on videos in the wild (Figure 1). Different from previous work that models each modality independently, we propose a two-stream network structure to learn the cross-modal distance in an end-to-end fashion using music-video pairs, while emotion tags are applied on each branch to form latent emotion space. Each branch is pre-trained as audio/video emotion tagging sub-network before feeding music-video pairs to both for cross-modal distance learning. This strategy requires fewer music-video pairs for training, and makes it possible to collect crowdsourced pairs of music and videos from independent sources. Note that the tags are only used during training phase to facilitate the convergence of cross-modal distance learning, and not necessary during model inference of music query. When a video queries the system, the model ranks every item of

an existing music database by Euclidean distance to the input video on the cross-modal embedding. The top ranked results represent the best matches to the input.

The main contributions of this paper are :

- A first system to address music retrieval from videos in the wild via learnable emotion space.
- A two-stream network structure and training strategy with emotion tags as joint constraints to learn cross-modal embeddings from fewer music-video pairs.
- Subjective evaluations showing promising retrieval results on real-world datasets.

2. RELATED WORK

2.1 Music, Videos and Emotions

The emotions associated with music and videos have been thoroughly studied. It has been suggested that emotions are one of the primary reasons people engage with music [15], and psychological studies reveal that people have emotional reactions on visual stimuli as well [7]. Therefore a natural way to retrieve music for videos is through the associations with emotion.

Categorical and dimensional representations have been used to represent emotion in music [18]. Discrete categorical tags include terms such as *calmness*, *sadness*, *anger*, and more. Gracenote [1] has performed a major effort around tagging the mood in music and provides mood taxonomies consisting of over 300 categories organized hierarchically. One work finds that the number of mood categories does not reflect the richness of emotions perceived by humans, or the taxonomy is inherently ambiguous [15]. Dimensional labels typically represent music on a 2-D plane of valence and arousal [30]. This continuous representation does not have the taxonomy problem, but has trouble distinguishing some psychology and emotion concepts such as *nostalgia*.

Emotion associated with images and videos have been also represented categorically [44] and dimensionally [24], similar to music. Seven emotion tags have been associated with videos of facial expressions [16, 17]. Eight basic emotion tags, with 3 variations on each tag are introduced for labeling unconstrained videos [38]. Movie scenes have been characterized in the valence-arousal space [3]. In [11], "Dominance" is introduced as an additional dimension for characterizing video emotions.

2.2 Cross-modal Audio-Visual Retrieval

Cross-modal retrieval has received increasing attention in the recent years. One work proposes a two-stream network structure for audio-vision cross-modal retrieval of common objects and their respective sounds, such as an image of a clock paired with the sound of an alarm [1]. This work has curated a large training dataset of common objects and sounds from publicly available sources. The cross-modal correspondence is learned from audio-visual pairs. Similar work has been described with additional modalities of text [2] and speech [26].

Related work for music includes cross-modal localization [43], association [20, 22], and generation [21] of music performances. Earlier work for cross-modal music retrieval involves extracting distance measurement between low-level features from video and music segments [40]. Some approaches synchronize video and music after representing each modality as sequence of 2-D valence-arousal features [23, 31, 32]. One work uses stochastic emotion space to bridge video and music [36], and another reports recommended music for still photo albums by defining a cross-modal graph on which synsets of mood tags from images and music are associated [5]. Pairing in these approaches recognizes each modality as explicit symbolic representations independently (e.g., hand-labeled emotions) before learning the association. Also, these systems mostly emphasize temporal inter-dependence, focusing on pairing a soundtrack to match the visual event with less demand on learning deep semantic representations on emotions.

Learning cross-modal embeddings end-to-end using cross-modal pairs could result in a deep representations of the relationships and improved performance at scale in a music retrieval setting. As presented in [12], music/video cross-modal retrieval has been modeled by learning from music-video pairs and presented on music and their respective music videos, where the videos were intentionally created as MV. Without constraining the cross-modal learning space, it requires intensive training on music-video pairs, e.g., existing music videos, and is not systematically evaluated on the retrieval results for videos from more sources. In this paper we also learn the embedding space in an end-to-end fashion using pairs of video and music, but constrain the learning space with emotion tags to form latent emotion space for each modality.

3. APPROACH

3.1 Network Architecture

3.1.1 Video Branch

The video branch consists of a feature extraction module followed by fully-connected layers for emotion tagging, left stream in Figure 2. We use pre-trained Inflated-3D model (I3D) [4] as the visual feature extractor. I3D was originally proposed for human action recognition from videos and was trained on the Kinetics dataset [4]. This pre-trained network has been successfully used for other video understanding tasks such as video captioning and audio-visual localization [34].

We input only the RGB frames and ignore optical flow. The system outputs a concatenation of the inception modules. We take a global average pool resulting in a 1024-D feature vector for each whole video of any duration. Next, we add fully connected layers. Each layer is followed by a ReLU nonlinearity, except that the output layer is followed by a sigmoid nonlinearity. Input video frames are resized to 224×224, and the RGB values are normalized to [-1,

[1] www.gracenote.com

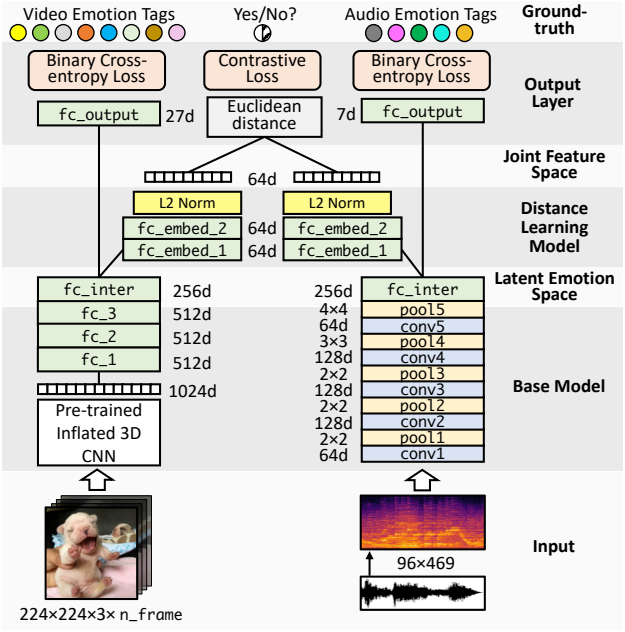


Figure 2. The two-stream network architecture with emotion constraints on each branch. The dimensions of fully-connected layer (fc), channel numbers of convolutional layer (conv), and pool sizes are marked aside each block. The kernel size for all the convolutional layers is 3×3.

1]. The video network is pre-trained with 27 emotion tags using binary cross-entropy loss.

3.1.2 Audio Branch

The audio branch consists of a ConvNet structure as a general audio tagging framework analogue to [6], right stream in Figure 2. Each convolutional layer is followed by a batch normalization and ReLU nonlinearity, and the output fully-connected layer has a sigmoid nonlinearity. Audio is trimmed to 10-sec with a sample rate of 12 kHz. Log mel-spectrograms are computed from input audio with a frame length of 42.7 ms (with 50% overlapping) and 96 mel-scale filter banks. The audio network is pre-trained with 7 emotion tags using binary cross-entropy loss.

3.1.3 Cross-modal distance learning

The cross-modal distance learning network is designed to embed the video and audio into the cross-modal embeddings (i.e., the joint feature space in Figure 2) so they can be directly compared as vector distance. This network takes the two 256-D penultimate layers from the video and audio branches to predict if the input music-video pair match, as a binary classification problem. The 256-D layers represent latent emotion space that is learned from the training pairs. The classifier is trained with the contrastive loss [10] on the Euclidean distance between each modality's 64-D cross-modal embedding, after L2 normalization.

3.2 Training

We first pre-train the audio and video branches independently as multi-label classifiers to predict emotion tags for each modality. The training stops when validation loss does not decrease for 5 consecutive epochs. Then the cross-modal distance learning framework is trained jointly while each branch predicts emotion tags. The network is jointly constrained and the three loss functions are equally weighted. This strategy constrains the learning space using emotion tags, and enables cross-modal distance learning from fewer music-video pairs. We use Adam optimizer [19], a stochastic gradient descent method, to minimize all the loss functions. When the model is trained, in practice all tracks in a music catalog are indexed by the embedding vector from the cross-modal joint feature space. Given any query video, the tracks in the database can be ranked by the Euclidean distance to the embedding vector calculated from the video. This creates a fast retrieval setup for large catalogs.

4. DATA

4.1 Training Data

The audio and video branches are pre-trained on independent music and video datasets with emotion labels. To train the cross-modal network we reuse the data from different modalities to create music-video pairs according to crowdsourced annotations about how well each pair matches.

4.1.1 AudioSet

We use the AudioSet [9] to pre-train the audio branch. AudioSet has human-labeled 10-second sound clips drawn from YouTube videos. We use data from "Music Mood" Ontology which contains music excerpts that are labeled with one of 7 music mood categories. We use AudioSet's official *Unbalanced Train* data where we randomly sample roughly 800 clips from each category for a total of 5.6K samples. We split it randomly where 80% is used for training and 20% is used for validation. We use AudioSet's official *Eval* data as the test set which consists of 354 samples, roughly 60 for each class, barring invalid download links. We do not use the videos from AudioSet for cross-modal distance learning because most contain a specific type of edited content, which is not suitable for the objective of retrieving music for videos from unconstrained sources.

4.1.2 Cowen2017

We use the Cowen2017 dataset [7] to pre-train the video branch. The dataset includes over 2K data samples including video clips from daily life, movies, cartoons, game scenes, artistic work, and more. Each video is annotated by several subjects who could select up to 27 emotion tags for each video. The annotations are aggregated so that each video's label is 27 emotion tags with a confidence value between 0 and 1. We split it randomly where and 80% is used for training and 20% is used to create the test set.

4.1.3 Music-video Pairs

To our knowledge there are no publicly available datasets that connect diverse videos with music, and contain the

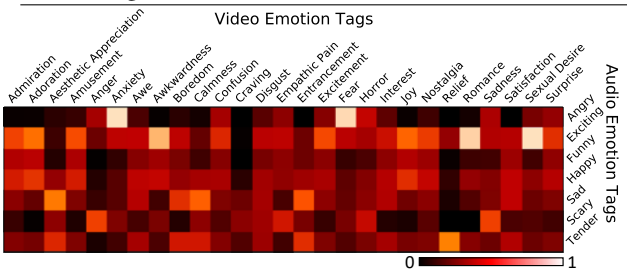


Figure 3. Visualizations of the emotion tags for the annotated music-video pairs from crowdsourced annotations. The color shows normalized counts.

respective emotion labels. Our goal is to construct a generalizable dataset that matches samples from one modality to the other while all samples have emotion labels. So we apply crowdsourcing to create music-video pairs from the same samples in AudioSet and Cowen2017. We follow the respective splits of training and test set as before so clips from one set do not appear in the other.

Training a cross-modal network requires positive and negative music-video pairs. The positive pairs are created by collecting binary crowdsourced judgments for randomly paired music and video clips until we have 1000 matches. Detailed crowdsourcing setup is described in Section 4.3. The negative samples are created from random un-annotated music-video pairs. In total we have class-balanced training set of 2000 pairs. We then perform the same process to collect another 1000 pairs on the test set. In Figure 3, we present a heatmap visualization of the relationship between the emotion tags of the positive audio-video pairs from the two modalities.

4.2 Real-world Data

To estimate how the system will perform on real-world data we curate videos and music from popular social media and music streaming platforms.

4.2.1 Spotify's Popular Music

We create a dataset of popular music from Spotify, an international music streaming platform. We identify popular Gracenote level 1 worldwide genres where at least 1000 tracks are streamed per day on Spotify. From each of the 30 most popular genres we select 40 of the most popular songs. The audio is downloaded from Spotify, which results in 1195 music clips.

4.2.2 The Moments in Time Dataset

We use video clips from the *Moments in Time* dataset [28] where each clip is a 3-second video snippet. The dataset was created for the tasks of action recognition and event understanding. We pick the first 100 moment categories (sorted alphabetically) from the *Moments in Time Mini* (a subset). From each category we select the first 5 video samples, totaling 500 video clips as the query videos.

4.2.3 Instagram Videos

Instagram is a social media platform for sharing photos and short videos. From a new account without search history we curate the top 20 videos from common photo post categories [13] : *Friends*, *Food*, *Gadget*, *Pets*, *Activities*, *Selfies*, *Fashion*, and we exclude *Captioned Photos* because the text may bias annotators' judgments and the system is currently not trained to process text. This results in 140 user uploaded short video clips.

4.3 Crowdsourcing Setup

Crowdsourced judgments are collected to create music-video pairs in the training and test datasets for cross-modal distance learning, and for subjective evaluations of music retrieval performance on real-world datasets. Experiments are run on the Figure-eight [2] platform which minimizes malicious activity during annotations and ensure high quality judgments for researchers.

Annotators are sourced from an international pool and each annotator is allowed to answer at most 10 questions, so that relevance judgments would not be overfit to any small group. Every question is randomly presented to at least 3 annotators. If the agreement among annotations is less than 65% per question, the number of annotators is dynamically increased up to 5 or until there is at least 65% agreement. Annotators are instructed to "listen in a quiet place, wear headphones, and watch the entire clip". The instructions for each audio-video pair are: "Please tell us if there is a common emotion theme in the video and the music, try not to focus on whether you like the music or the video." Possible responses are: "yes they match", "no they do not match", and "I am not sure".

To avoid biasing the pool of contributors we do not use gold standard screening questions that resemble the annotation questions, a common practice on Figure-eight. Instead to monitor annotation quality and attentiveness, we monitor whether any annotator's responses consistently deviate from the responses of other annotators. If an annotator's responses are different from the average annotation in more than 3 questions we flag that individual to analyze their contributions. We do not find any annotators fall into this group. In total we have collected thousands of annotations from subjects from 12 countries. Approximately 71% of the questions have greater than 66% agreement on "yes" and "no" responses. The remaining has responses split between "yes", "no", and "not sure", and for this work we consider the responses as "no" because annotators do not perceive relevance.

5. EXPERIMENTS

The model performance is evaluated numerically on the tasks of 1) predicting emotion tags from each branch independently, 2) predicting if the input audio-video pairs match, and 3) cross-modal music retrieval, on the annotated datasets. The music retrieval performances are also

[2] https://www.figure-eight.com

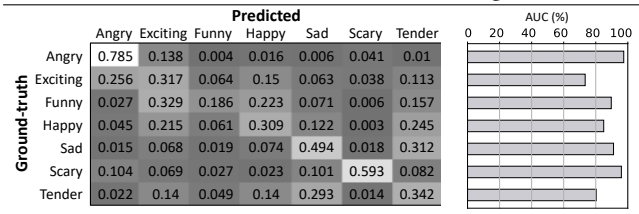


Figure 4. Performance of the audio branch for predicting emotion tags after pre-training. Results are presented as confusion matrix (left), and AUC on each category (right).

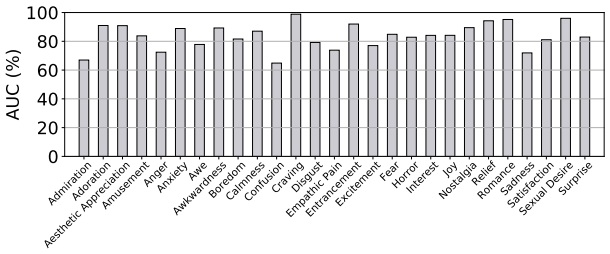


Figure 5. Performance of the video branch for predicting emotion tags after pre-training. AUCs are calculated using a score margin at 0.25.

evaluated on real-world data using crowdsourced subjective judgments.

5.1 Predicting Audio Emotion Tags

We evaluate the pre-trained audio branch using the labeled heldout data from AudioSet. The music emotion tagging result is evaluated using multi-class classification metrics. Figure 4 presents the confusion matrix, where *Angry* and *Scary* are likely to get high true positive rates, while the boundaries between *Exciting*, *Funny*, and *Happy* are blurred. We observe that angry music is generally noisy with strong percussion, while scary music has strong inharmonic components, as more distinct characteristics.

We also evaluate the model using the area under the receiver operating characteristic curve (AUC), a statistical metric that summarizes model's performance regardless of classification threshold. Equivalently, it measures the performance of binary classifiers (measuring each tag independently) by ranking scores, i.e., the probability that a randomly chosen positive is ranked ahead of a randomly chosen negative. The performance on each emotion tag is shown in Figure 4, with an average of **87.88**%.

5.2 Predicting Video Emotion Tags

The pre-trained video tagging branch is evaluated using heldout data from Cowen2017. Similar to scored AUCs [35, 37], we set a score margin on soft ground truth labels to report the performance. This metric assesses how well relative differences between video samples in the dataset can be predicted. It compares the sign of the differences between any two predictions to sign of the differences of the respective ground truth ones. Performance is measured only when the two data points have sufficiently large ground truth differences, e.g., 0.25 as used here. The average AUC for all tags is **83.79**%, as shown in figure 5 on each tag.

Method	2-stream	emotion [32]	emtoion [5]	proposed
Result	38.1%	36.0%	46.8%	**68.0**%

Table 1. Music retrieval performances on the labeled datasets compared with three baseline methods.

5.3 Predicting Audio-video Pairs

We also evaluate the cross-modal network to understand overall performance on cross-modal distance learning and the effects of the emotion tags which constrain the video and audio branches during cross-modal training. The cross-modal network predicts the input audio-video pairs as either positive or negative (matched or not) on the 1000 pairs from test set, and is evaluated as a binary classifier with a threshold of 0.5. We create a baseline model with the same two-stream network structure but without pre-training the branches on emotion tags or joint loss functions. The two models are trained and evaluated with the same dataset, described in Section 4.1.3. The accuracy of the proposed model is **79.00**% while the baseline achieves **63.30**%. Note that this baseline system is a general two-stream cross-modal distance learning network, e.g., [1], which usually requires intensive training on a large number of training pairs. The results indicate that pre-training and joint constraints on emotion tags is important for cross-modal distance learning when the training data is limited and the task includes data with highly diverse.

5.4 Cross-modal Music Retrieval

We reuse the heldout data from Cowen2017 and Audioset as query videos and the pool of music, respectively, where for each query video there are on-average 16.2 music samples from the pool are annotated as ground-truth retrieval. The music retrieval performance is evaluated by counting the number of videos that can retrieve a relevant song. For each query video only the top retrieval is considered, after ranking all the 354 music tracks. The proposed model retrieves relevant music for **68.0**% of query videos.

We also compare the performance to three baseline models, as presented in Table 1. The first baseline model is the same baseline as illustrated in Section 5.3, which share the same network structure but without emotion tags and pre-traning to form latent emotion space. It only achieves satisfactory retrieval for **38.1**% of videos.

The second baseline models each modality as continuous emotion representation on the valence-arousal (V-A) space, analogue to [32]. We implement this by adapting the proposed model structure, where the output layer from each branch is replaced with 2-D states to represent valence and arousal constrained by mean squared error (MSE) from the ground-truth values as a regression problem, without the two-stream structure for learning the cross-modal embeddings. The audio model achieves R^2

Video source	Moments in Time	Instagram
Result	58.2%	64.3%

Table 2. Music retrieval performance on Spotify music using query videos collected from new sources.

statistics of 53%/36% for arousal/valence when trained and evaluated on the *1000Song* dataset [33]. The video model achieves R^2 of 75.04%/60.28% for arousal/valence when trained and evaluated on the AudioSet videos using annotated labels. Both models achieve higher performance than the original work on emotion prediction [32]. We map the two modalities in the A-V space and the model achieves relevant retrievals for **36.0**% videos.

The third baseline matches modalities according to hand-labeled emotion tags, analogue to [5]. To implement this we modify the model structure to build the cross-modal distance learning structure from the predicted emotion tags from each branch, instead of the 256-D latent emotion space. This model achieves relevant retrievals for **46.8**% videos.

These experiments show that the proposed approach outperforms three baseline solutions. Two of the baselines join the modalities directly on predicted emotion states: arousal-valence values or explicit emotion tags. It suggests that our model learns deeper relationships between the modalities in the cross-modal space. Comparing to the other baseline, the results indicate that when the model is not constrained with emotion tags, only 2000 audio-video pairs as training set is too small for the network to learn the cross-modal embeddings to represent underlying the relationships between the cross-modal inputs.

5.5 Performance on Real-world Data

We assess how well our proposed model works on real-world data by collecting human judgments using the crowdsourcing setup in section 4.3. We use the 500 samples from the *Moments in Time* dataset and 140 user-generated videos from Instagram to retrieve music from a pool of music clips downloaded from Spotify. The model can successfully retrieve music for **58.2**% and **64.3**% videos, respectively. Music retrieval performance is better on Instagram than *Moments in Time*.

Note that Instagram videos are uploaded by users to visually share an experience or a mood that incites an emotion [14]. Instead, *Moments in Time* was created to capture different actions objectively without capturing sentiment [28]. This difference may explain why performance is higher on the Instagram videos.

5.6 Qualitative Analyses

We qualitatively analyze the latent emotion space learned from the two-stream model. We take the latent emotion space from the audio branch and create a t-SNE visualization [25], as plotted in Figure 6, to study how the matched videos localize in this 2-D space. Each dot represents a music sample from AudioSet, and we color the ones with the original label "Tender" in grey. Among these samples, we randomly select some from different regions and they are presented using different colors and with the thumbnails of the paired videos from model output. It indicates that samples close together with similar granular emotions are usually associated with similar videos. For example, samples in yellow represent music that sounds serene or soothing, and associated with outdoor nature scenes. This suggests that the proposed framework constrained with emotion tags enables the model to learn a interpretable emotion space and cross-modal correspondence including nuances that are no represented in the original tags.

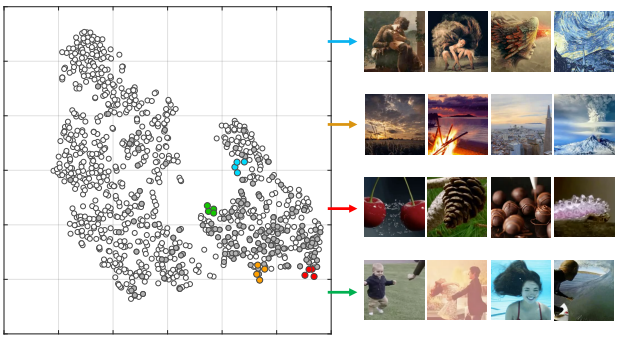


Figure 6. The t-SNE visualization of the latent emotion space from the audio branch. Samples from the "Tender" tag are in gray. Four randomly selected regions for tender music are presented in colors representing different emotion concepts: gloomy, ambient, delicate, sweet, each with the thumbnails of the paired videos displayed.

We also investigate some failure retrieval cases and find most are due to incorrect predictions of video emotions. Also, several retrievals are matched on emotions but mismatched on cultural signals. For example, a video with people bowing to Beyonce as she wears a crown is paired with music that sounds "mystical" or "heavenly", which is annotated as "mismatch" likely because annotators recognize Beyonce and they are expecting her music. If cultural signals are ignored these retrievals may have been reasonable matches. Overall, the results indicate that the system is effective for music retrieval, and we expect improvements by incorporating more signals such as culture or genres.

6. CONCLUSION

We have addressed the problem of music retrieval using real-world videos from unconstrained sources. We haved proved that emotion tags can constrain the learning space and enable cross-modal distance learning from fewer annotated cross-modal pairs. Experiments show that our model can retrieve promising results for user-generated query videos. As an application, this model can offer novel music query solutions for daily life videos which can enhance visual messages to make sharing more enjoyable. We also expect this work to have product implications in the music streaming business. In the future we plan to personalize the music that is retrieved for users' tastes.

7. REFERENCES

[1] Relja Arandjelović and Andrew Zisserman. Objects that sound. In *Proc. European Conference on Computer Vision (ECCV)*, 2018.

[2] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. See, hear, and read: Deep aligned representations. *arXiv preprint arXiv:1706.00932*, 2017.

[3] Yoann Baveye, Emmanuel Dellandrea, Christel Chamaret, and Liming Chen. Liris-accede: A video database for affective content analysis. *IEEE Transactions on Affective Computing*, 6(1):43–55, 2015.

[4] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] Jiansong Chao, Haofen Wang, Wenlei Zhou, Weinan Zhang, and Yong Yu. Tunesensor: A semantic-driven music recommendation service for digital photo albums. In *Proc. International Semantic Web Conference (ISWC)*, 2011.

[6] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. In *Proc. International Society for Music Information Retrieval (ISMIR)*, 2016.

[7] Alan S Cowen and Dacher Keltner. Self-report captures 27 distinct categories of emotion bridged by continuous gradients. *National Academy of Sciences*, 114(38):E7900–E7909, 2017.

[8] J. Stephen Downie. Music information retrieval. *Annual Review of Information Science and Technology*, 37:295–340, 2003.

[9] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780, 2017.

[10] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1735–1742, 2006.

[11] Alan Hanjalic and Li-Qun Xu. Affective video content representation and modeling. *IEEE Transactions Multimedia*, 7(1):143–154, 2005.

[12] Sungeun Hong, Woobin Im, and Hyun S Yang. Cbvmr: Content-based video-music retrieval using soft intra-modal structure constraint. In *Proc. ACM International Conference on Multimedia Retrieval*, pages 353–361, 2018.

[13] Yuheng Hu, Lydia Manikonda, Subbarao Kambhampati, et al. What we instagram: A first analysis of instagram photo content and user types. In *Proc. International Conference on Weblogs and Social Media (ICWSM)*, pages 595–598, 2014.

[14] Christina A Jackson and Andrew F Luchner. Self-presentation mediates the relationship between self-criticism and emotional response to instagram feedback. *Personality and Individual Differences*, 133:1–6, 2018.

[15] Patrik N Juslin and Petri Laukka. Expression, perception, and induction of musical emotions: A review and a questionnaire study of everyday listening. *Journal of New Music Research*, 33(3):217–238, 2004.

[16] Samira Ebrahimi Kahou, Xavier Bouthillier, Pascal Lamblin, Caglar Gulcehre, Vincent Michalski, Kishore Konda, Sébastien Jean, Pierre Froumenty, Yann Dauphin, Nicolas Boulanger-Lewandowski, et al. Emonets: Multimodal deep learning approaches for emotion recognition in video. *Journal on Multimodal User Interfaces*, 10(2):99–111, 2016.

[17] Heysem Kaya, Furkan Gürpınar, and Albert Ali Salah. Video-based emotion recognition in the wild using deep transfer learning and score fusion. *Image and Vision Computing*, 65:66–75, 2017.

[18] Youngmoo E Kim, Erik M Schmidt, Raymond Migneco, Brandon G Morton, Patrick Richardson, Jeffrey Scott, Jacquelin A Speck, and Douglas Turnbull. Music emotion recognition: A state of the art review. In *Proc. International Society for Music Information Retrieval (ISMIR)*, pages 255–266, 2010.

[19] D Kinga and J Ba Adam. A method for stochastic optimization. In *Proc. International Conference on Learning Representations (ICLR)*, volume 5, 2015.

[20] Bochen Li, Karthik Dinesh, Zhiyao Duan, and Gaurav Sharma. See and listen: Score-informed association of sound tracks to players in chamber music performance videos. In *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2906–2910. IEEE, 2017.

[21] Bochen Li, Akira Maezawa, and Zhiyao Duan. Skeleton plays piano: Online generation of pianist body movements from midi performance. In *Proc. International Society for Music Information Retrieval (ISMIR)*, 2018.

[22] Bochen Li, Chenliang Xu, and Zhiyao Duan. Audio-visual source association for string ensembles through multi-modal vibrato analysis. In *Proc. Sound and Music Computing (SMC) Conference*, pages 159–166, 2017.

[23] Jen-Chun Lin, Wen-Li Wei, and Hsin-Min Wang. Emv-matchmaker: emotional temporal course modeling and

matching for automatic music video generation. In *Proc. ACM International Conference on Multimedia*, pages 899–902, 2015.

[24] Xin Lu, Poonam Suryanarayan, Reginald B Adams Jr, Jia Li, Michelle G Newman, and James Z Wang. On shape and the computability of emotions. In *Proc. ACM International Conference on Multimedia*, pages 229–238, 2012.

[25] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[26] Gou Mao, Yuan Yuan, and Lu Xiaoqiang. Deep cross-modal retrieval for remote sensing image and audio. In *Proc. IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS)*, pages 1–7. IEEE, 2018.

[27] Brian McFee and Gert RG Lanckriet. The natural language of playlists. In *Proc. International Society for Music Information Retrieval (ISMIR)*, pages 537–541, 2011.

[28] Mathew Monfort, Bolei Zhou, Sarah Adel Bargal, Alex Andonian, Tom Yan, Kandan Ramakrishnan, Lisa Brown, Quanfu Fan, Dan Gutfruend, Carl Vondrick, et al. Moments in time dataset: one million videos for event understanding. *arXiv preprint arXiv:1801.03150*, 2018.

[29] Meinard Mueller, Andreas Arzt, Stefan Balke, Matthias Dorfer, and Gerhard Widmer. Cross-modal music retrieval and applications: An overview of key methodologies. *IEEE Signal Processing Magazine*, 36(1):52–62, 2019.

[30] James A Russell. A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161, 1980.

[31] Shoto Sasaki, Tatsunori Hirai, Hayato Ohya, and Shigeo Morishima. Affective music recommendation system based on the mood of input video. In *International Conference on Multimedia Modeling*, pages 299–302. Springer, 2015.

[32] Ki-Ho Shin and In-Kwon Lee. Music synchronization with video using emotion similarity. In *Proc. International Conference on Big Data and Smart Computing (BigComp)*, pages 47–50, 2017.

[33] Mohammad Soleymani, Micheal N Caro, Erik M Schmidt, Cheng-Ya Sha, and Yi-Hsuan Yang. 1000 songs for emotional analysis of music. In *Proc. ACM international workshop on Crowdsourcing for multimedia*, pages 1–6. ACM, 2013.

[34] Yapeng Tian, Jing Shi, Bochen Li, Zhiyao Duan, and Chenliang Xu. Audio-visual event localization in unconstrained videos. In *Proc. European Conference on Computer Vision (ECCV)*, 2018.

[35] Stijn Vanderlooy and Eyke Hüllermeier. A critical analysis of variants of the auc. *Machine Learning*, 72(3):247–262, 2008.

[36] Ju-Chiang Wang, Yi-Hsuan Yang, I-Hong Jhuo, Yen-Yu Lin, Hsin-Min Wang, et al. The acousticvisual emotion guassians model for automatic generation of music video. In *Proc. ACM international conference on Multimedia*, pages 1379–1380, 2012.

[37] Shaomin Wu, Peter Flach, and Cèsar Ferri. An improved model selection heuristic for auc. In *Proc. European Conference on Machine Learning*, pages 478–489. Springer, 2007.

[38] Baohan Xu, Yanwei Fu, Yu-Gang Jiang, Boyang Li, and Leonid Sigal. Video emotion recognition with transferred deep feature encodings. In *Proc. ACM on International Conference on Multimedia Retrieval*, pages 15–22, 2016.

[39] Yi-Hsuan Yang and Homer H Chen. Machine recognition of music emotion: A review. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):40, 2012.

[40] Jong-Chul Yoon, In-Kwon Lee, and Siwoo Byun. Automated music video generation using multi-level feature-based segmentation. *Multimedia Tools and Applications*, 41(2):197, 2009.

[41] Yi Yu, Zhijie Shen, and Roger Zimmermann. Automatic music soundtrack generation for outdoor videos from contextual sensor information. In *Proc. ACM international conference on Multimedia*, pages 1377–1378, 2012.

[42] Yichi Zhang, Bryan Pardo, and Zhiyao Duan. Siamese style convolutional neural networks for sound search by vocal imitation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(2):429–441, 2019.

[43] Hang Zhao, Chuang Gan, Andrew Rouditchenko, Carl Vondrick, Josh McDermott, and Antonio Torralba. The sound of pixels. *arXiv preprint arXiv:1804.03160*, 2018.

[44] Sicheng Zhao, Yue Gao, Xiaolei Jiang, Hongxun Yao, Tat-Seng Chua, and Xiaoshuai Sun. Exploring principles-of-art features for image emotion recognition. In *Proc. ACM International Conference on Multimedia*, pages 47–56, 2014.

INVESTIGATING CNN-BASED INSTRUMENT FAMILY RECOGNITION FOR WESTERN CLASSICAL MUSIC RECORDINGS

Michael Taenzer[1], Jakob Abeßer[1], Stylianos I. Mimilakis[1], Christof Weiß[2], Meinard Müller[2], and Hanna Lukashevich[1]

[1] Fraunhofer IDMT, Ilmenau, Germany
[2] International Audio Laboratories Erlangen, Germany
michael.taenzer@idmt.fraunhofer.de

ABSTRACT

Western classical music comprises a rich repertoire composed for different ensembles. Often, these ensembles consist of instruments from one or two of the families woodwinds, brass, piano, vocals, and strings. In this paper, we consider the task of automatically recognizing instrument families from music recordings. As one main contribution, we investigate the influence of data normalization, pre-processing, and augmentation techniques on the generalization capability of the models. We report on experiments using three datasets of monotimbral recordings covering different levels of timbral complexity: isolated notes, isolated melodies, and polyphonic pieces. While data augmentation and the normalization of spectral patches turned out to be beneficial, pre-processing strategies such as logarithmic compression and channel-energy normalization did not lead to substantial improvements. Furthermore, our cross-dataset experiments indicate the necessity of further optimization routines such as domain adaptation.

1. INTRODUCTION

In classical music, there are compositions for a variety of distinct instrumentations. Chamber music, for example, comprises ensembles of different size, which often consist of instruments from a specific instrument family such as woodwinds, brass, piano, vocal, or strings. Typical examples are brass quintets, piano duos, choirs, or string quartets. While the automatic classification of such *monotimbral* recordings w.r.t. the instrument family is a simplification of general instrument recognition, it still constitutes a challenging task. Successfully tackling this problem could help to organize and browse classical music collections. Furthermore, recognizing instrument families from monotimbral recordings constitutes the first step towards handling more complicated scenarios such as orchestra recordings, where we often find passages featured by specific instrument families.

In this paper, we approach the task of music instrument family recognition from classical music recordings. For our experiments, we consider three scenarios using datasets of monotimbral recordings with different levels of timbral complexity. We start with analyzing recordings of isolated notes (`IN`) played by an individual instrument. Furthermore, we test on isolated, monophonic melodies (`IM`) with a natural variety of note durations. For the third scenario, we consider monotimbral, mostly polyphonic music recordings (`MP`), where one or more instruments of the same instrument family are playing simultaneously. In Section 3, we describe these datasets in detail.

To approach the instrument family recognition task, we make use of a state-of-the-art instrument recognition algorithm [8] based on convolutional neural networks using spectrogram segments as input. As our main contributions, we apply this method to the instrument family scenario. For the `MP` scenario, we investigate how the model performance can be improved using strategies for data augmentation and pre-processing. We systematically test the generalization capability of the trained models to previously unseen datasets in a sequence of cross-dataset experiments.

2. RELATED WORK

Traditional algorithms for automatic instrument recognition (AIR) rely on audio features measuring instrument-specific timbral properties of music signals. Fuhrmann [4] provides a comprehensive overview of such techniques. As an example with a focus on classical music, Eggink & Brown [3] propose a system to recognize five wind and string instruments based on partial frequency and magnitude features combined with a Gaussian classifier.

Due to the rapid proliferation of deep-learning techniques, most recent publications mainly focus on data-driven algorithms, which are the focus of this literature review. These algorithms are trained to learn a direct mapping from low-level signal representations such as mel-spectrograms to higher-level attributes such as instrument labels. While data-driven approaches require less domain knowledge, they usually need large amounts of training data in order to learn models that generalize well to unseen

datasets. However, popular AIR datasets such as MedleyDB [1], IRMAS [2], or MusicNet [21] are still of limited size. As a consequence, authors often apply data augmentation techniques such as pitch shifting [7] to virtually enlarge the number of audio files.

Concerning the generalization capability, AIR approaches based on deep neural networks (DNNs) show good performance on particular datasets, but cross-dataset experiments (as we present in Section 5.2) are rarely performed. Such experiments are crucial for better understanding to which extent DNN models generalize to unseen datasets with different characteristics. In the related field of audio event recognition, researchers often observe this limitation of data-driven algorithms and suggest additional domain adaptation steps [5]. Another challenge is the entanglement between perceptual attributes such as pitch and timbre in spectrogram representations. Lostanlen et al. [14] propose weight-sharing strategies for DNN models in order to derive pitch-invariant representations, which still maintain good timbre discriminability.

Typical model architectures used in recently proposed AIR systems are convolutional neural networks (CNNs) [6–9, 12, 17, 20] and hybrid convolutional-recurrent neural networks (CRNNs) [7]. Most of the CNN architectures comprise several convolutional layers for feature learning and a set of dense layers for classification. Han et al. [8] proposed such a CNN architecture to recognize the predominant instrument in polyphonic and multitimbral recordings. The authors evaluate different late-fusion techniques to aggregate frame-level model predictions in order to obtain song-level instrument labels. This model has been used and extended in recent AIR literature [6, 20]. Takahashi et al. [20] show in a comparative experiment that using horizontal and vertical filter shapes instead of symmetrical ones improves recognition performance, but requires more training time. Hung & Yang [9] tested an alternative CNN model, which includes residual blocks with additional skip connections to allow for reducing the vanishing-gradient problem during training.

Concerning the input representation, most DNN-based AIR systems process mel-spectrogram segments (*patches*). As alternative, Hung & Yang test constant-Q spectrograms and harmonic-series features as input to the models [9]. Li et al. [12] propose an end-to-end-learning approach using a CNN architecture that directly processes raw audio data. Hung & Yang [9] show that using score information as an additional cue leads to small improvements in the frame-level recognition of seven classical instruments.

3. DATASETS

In this section, we describe three datasets that we use for our instrument family recognition experiments. Regarding the level of difficulty, the isolated-note scenario (`IN`) constitutes the simplest task represented by the Studio On Line Dataset (`DB-SOL`) presented in Section 3.1. A scenario with increased level of difficulty comprises isolated melodies (`IM`), represented by the University of Rochester Multi-modal Music Performance Dataset (`DB-URMP`) described in Section 3.2. As our most complicated scenario, we consider monotimbral polyphonic recordings of classical music realized in the Monotimbral Classical Dataset (`DB-MTC`), which comprises recordings of monotimbral, mostly polyphonic classical pieces (`MP`, see Section 3.3). Table 1 summarizes the properties of the three datasets.

Table 1: Number of audio files, spectral patches, and average patches per file for each dataset and experiment.

Dataset	Files	Patches	Patches / file (avg)
Original Datasets (with silent patches)			
`DB-MTC`	50	38078	762
`DB-MTC`$^+$	400	304624	762
`DB-URMP`	149	33693	226
`DB-SOL`	20604	225273	11
Experiment 1 (silent patches removed)			
`DB-MTC`	50	34163	683
`DB-MTC`$^+$	400	281841	705
Experiment 2 (3 classes, silent patches removed)			
`DB-MTC`	30	20900	697
`DB-URMP`	149	31236	210
`DB-SOL`	20604	202486	10
`DB-M/U/S`	20783	254622	12

3.1 Studio On Line Dataset (`DB-SOL`)

The Studio On Line dataset [1], recorded in 2002 at IRCAM (Paris), comprises over 25000 isolated note recordings from 16 different instruments covering the instrument families woodwinds, brass, and strings. Recognizing the instrument family of such isolated note recordings (`IN`) constitutes a relatively simple scenario since there is no spectral overlap of multiple notes. However, the large variety of instrument playing techniques—in particular, frequency modulation techniques such as vibrato and trill—makes the recognition task more complex. For our experiments, we discarded recordings from `DB-SOL` that only comprise mechanical instrument sounds without a clear pitch as well as breathing and speaking sounds.

3.2 University of Rochester Multi-modal Music Performance Dataset (`DB-URMP`)

The University of Rochester Multi-modal Music Performance (URMP) Dataset [11] was originally published to study audio-visual music performance analysis. The dataset comprises 44 ensemble pieces including duets, trios, quartets, and quintets, most of which are arrangements of popular classical pieces. For all pieces, multitrack recordings are available with a total of 149 isolated instruments tracks. Within each track, one melody instrument from the families woodwinds, brass, and strings is recorded in isolation. We use these individual tracks as the basis for our isolated-melodies (`IM`) scenario.

[1] Freely available as part of the Orchids software at `http://forumnet.ircam.fr/product/orchids-en/`. In [13], the dataset was used for evaluating an instrument recognition system.

3.3 Monotimbral Classical Dataset (DB-MTC)

To test the instrument family recognition task on a realistic scenario, we compiled a dataset consisting of 50 tracks from commercial recordings. The data comprises mostly polyphonic classical pieces composed for instruments of one family. For each of the five families, we included ten audio files, each from a different CD. The total duration in minutes for each instrument family is 63.2 (woodwinds), 37.6 (brass), 75.1 (piano), 51.4 (vocal), and 82.8 (strings).

The woodwind class mainly comprises chamber music works such as wind quintets by Cambini, Danzi, Hindemith, Nielsen, and Reicha, as well as a quartet by Rossini and a sextet by Janacek. Furthermore, we consider a serenade by W. A. Mozart, an excerpt from Dvořak's Ninth symphony (*New World*), and a partita for wind ensemble by Krommer. We are aware of the problem that wind ensembles often include a french horn, which is a brass instrument. While this is a typical situation in classical music, it might influence our recognition experiments. For the brass selection, we use brass ensemble music for five to ten players. We consider pieces by ten different composers, played by Canadian Brass, German Brass, Mnozil Brass, and other ensembles. Into the piano class we placed solo sonatas and fugues by Beethoven, Berg, C. P. E. Bach, and others, played by different pianists. Regarding vocal music, we include music for solo voice—such as Berio's *Sequenza III*—and choirs. The choir pieces comprise a renaissance composition by Allegri, romantic pieces by Bruckner and Janacek, modern pieces by Ligeti and Scelsi, and more. The strings class consists of several string orchestra pieces by Barber, Hindemith, Lutosławski, Penderecki, and Rawsthorne. Additionally, we include chamber music such as string quartets, a quintet by Schubert and a sextet by Brahms. [2]

4. SYSTEM OVERVIEW

In the following, we present our system for instrument family recognition, which consists of three main components. The first component (Section 4.1) transforms the audio signal of a music recording into a mel-based time–frequency representation. The second component (Sections 4.2 and 4.3) applies pre-processing techniques such as normalization or compression to the time–frequency representation. The third component (Section 4.4) consists of a CNN that outputs class probabilities and is trained in a supervised fashion. Figure 1 summarizes the main processing steps together with additional details regarding the network architecture (second and third columns of the figure).

4.1 Mel-spectogram Representation

For computing the time–frequency representation of the recordings, we follow the work by Han et al. [8]. We resample the audio signals to a sample rate of f_s = 22050 Hz and compute the mel-spectrogram [3] using 128 mel-bands, a hop size of 512 samples, and a window size of 1024 samples. Then, we normalize the magnitude in each frequency band of the mel-spectogram via dividing by the number of mel-bands. For each recording, we further segment the resulting mel-spectrogram representation into time–frequency patches with a length of 43 frames (approx. one second) with an overlap of 21 frames (approx. 0.5 seconds). This results in a tensor $X \in \mathbb{R}^{N \times 43 \times 128}$, where N indicates the total number of computed patches. In order to remove potential silent parts in the recordings, we discard a patch as soon as the mean of its magnitude values is below 5% of the entire file's maximal magnitude.

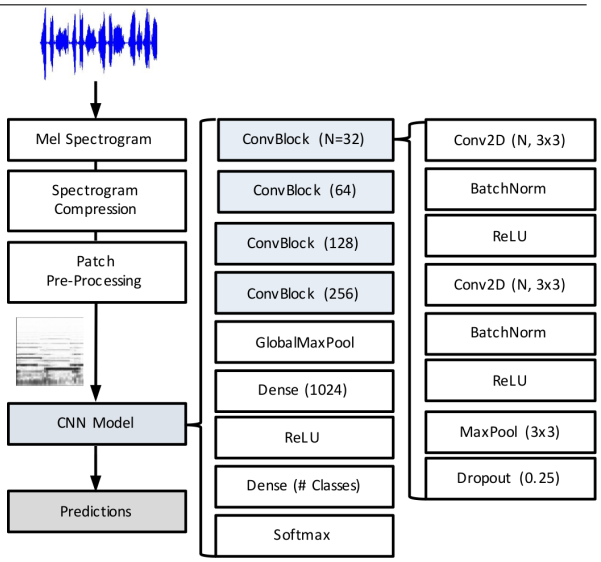


Figure 1: Reference model proposed by Han et al. [8] with slight modifications as discussed in Section 4.4. Spectrogram patches are processed by successive pairs of convolutional layers followed by batch normalization and ReLU activation function, max pooling (MaxPool), and global max-pooling (GlobMaxPool).

4.2 Spectrogram Dynamic Range Compression

Classical music recordings commonly exhibit a large dynamic range. To account for this, we investigate the effect of pre-processing strategies for compressing the dynamic range of the mel-spectrograms. In the following, we compare four different approaches for dynamic compression.

The first strategy, denoted as `NO`, does not apply any dynamic range compression. In this case, we directly use the mel-spectrogram as input to the model. The second strategy applies logarithmic compression defined by $X \leftarrow \log(1 + \gamma X)$. For our experiments, we consider two settings with $\gamma = 1$ (denoted as `LC 1`) and $\gamma = 10000$ (denoted as `LC 10000`), respectively. As the fourth strategy, we employ Per-Channel Energy Normalization (`PCEN`) proposed in [22] and further studied in [15]. `PCEN` ap-

[2] Due to copyright issues, we cannot publish the audio files. Instead, we publish the spectrogram patch tensors and corresponding targets to allow for reproducibility of our experiments under `https://doi.org/10.5281/zenodo.3258829`.

[3] We use the implementation from `librosa` (`https://librosa.github.io/librosa/`), version 0.6.2.

plies a first-order Infinite Impulse Response (IIR) filter, which controls the gain of the spectral representation, followed by dynamic range compression [22]. In principle, PCEN enhances prominent spectral characteristics (such as onsets), while attenuating low-energy frequency bands that are correlated with reverberation or corrupted by noise [22]. As the main benefit, the resulting spectral representation is robust against effects of reverberation and additive noise. For our experiments, we use PCEN as implemented in the librosa[3] Python library with default parameters (gain=0.98, bias=2, power=0.5, time_constant=0.4).

4.3 Patch Pre-processing

After the dynamic range compression step of the mel-spectrogram, we apply another pre-processing technique in order to normalize the spectral patches before we feed them to the CNN model. For this *patch pre-processing* step (not to be confused with the batch normalization within the CNN), we compare four different approaches. Let mean$(\cdot)$ and std$(\cdot)$ denote the computation of the average and standard deviation, respectively. $X_{:,:,f}$ denotes a slice of the given tensor $X \in \mathbb{R}^{N\times 43\times 128}$ for a fixed frequency index $f \in \{1,\ldots,128\}$. Similarly, $X_{p,:,:}$ denotes a slice of the tensor X for a fixed patch index $p \in \{1,\ldots,N\}$. Based on this, we define the four approaches as follows:

The first approach (A) performs frequency-based Zero-Mean and Unit-Variance (ZMUV) normalization following early approaches for efficiently training DNNs [10]. For each $f \in \{1,\ldots,128\}$, it is computed via:

$$X_{:,:,f} \leftarrow \frac{X_{:,:,f} - \text{mean}(X_{:,:,f})}{\text{std}(X_{:,:,f}) + \epsilon}. \quad (1)$$

The second approach (B) applies global ZMUV normalization to the tensor X, following the work presented in [18]:

$$X \leftarrow \frac{X - \text{mean}(X)}{\text{std}(X) + \epsilon}. \quad (2)$$

The third approach (C) employs local patch pre-processing, where each patch $p \in \{1,\ldots,N\}$ is normalized individually in the following way:

$$X_{p,:,:} \leftarrow \frac{X_{p,:,:} - \text{mean}(X_{p,:,:})}{\text{std}(X_{p,:,:}) + \epsilon}. \quad (3)$$

The fourth approach (denoted as "–") does not apply any pre-processing: The mel-spectral representation is provided directly to the CNN model.

For the approaches A and B, we apply ZMUV normalization to the validation and test set using mean and standard deviation as computed from the training set.

4.4 CNN Model

For our experiments, we adopt a CNN architecture proposed by Han et al. [8], illustrated in Figure 1. The model is based on a VGG-type architecture [19] and consists of four blocks that perform convolution operations. Each block contains a pair of 2D convolutional layers, each comprising K kernels of size 3×3. After each convolutional layer, we apply batch normalization (BatchNorm) followed by the Rectified Linear Unit (ReLU) activation function. At the end of each convolutional block, we use 3×3 max-pooling and dropout (with probability 0.25). Between subsequent convolutional blocks, we increase the number of channels K by a factor of two.

After the fourth convolutional block, we use a global max-pooling layer in order to flatten the latent representation. We give the flattened representation to a fully-connected feed-forward layer with 1024 units, followed by the final feed-forward layer that uses a soft-max activation function. We extend the architecture presented by Han et al. [8] using additional batch normalization layers after each convolutional layer. The batch normalization layers perform ZMUV normalization across each batch of mel-spectrogram patches. We train the model using categorical cross-entropy loss, the Adam optimizer with a learning rate of 10^{-4}, and a batch size of 128. In order to reduce overfitting, we implement early stopping during model training with a patience of 20 epochs. Since the audio files in DB–MTC substantially differ in length, we use a class-weighting scheme during training to compensate for class imbalance, which is computed as an inverse proportion of the number of training items per class.

5. EXPERIMENTS

In this section, we present our experiments on instrument family recognition. For Experiment 1 (Section 5.1), we consider polyphonic, monotimbral recordings using the DB–MTC dataset and test the improvement strategies discussed in Section 4. In Experiment 2 (Section 5.2), we investigate the generalization capabilities of the trained models in a cross-dataset experiment using the three datasets described in Section 3. As evaluation measure, we report the micro-average F-score. The F-score is computed as the harmonic mean between precision and recall on a patch-level and is not affected by potential class imbalance.

5.1 Experiment 1: Instrument Family Classification in Monotimbral Classical Music Recordings

In this experiment, we evaluate whether data augmentation, spectrogram compression, and patch pre-processing techniques lead to an improved classification performance. We focus on the MP scenario using the DB–MTC dataset.

5.1.1 Data Augmentation

Due to the small number of 10 audio files per instrument family in DB–MTC (see Section 3.3), we enlarge the dataset using the data augmentation techniques shown in Table 2. We apply algorithms taken from the Audio Degradation Toolbox [16] that implement brown and white noise, two room impulse responses, dynamic range compression, and two kinds of signal attenuation. In total, we create seven augmented versions of each audio file in the dataset, thus increasing the size of the dataset from 50 to 400 files. We refer to this augmented dataset as DB–MTC$^+$.

Table 2: Overview of applied data augmentation methods. *(def)* indicates where default presets from [16] are used.

Abbr.	Augmentation Type	Approach
Noi	Noise	Brown noise, SNR -6 dB
		White noise, SNR +22 dB
Imp	Impulse Response	Great Hall *(def)*
		Classroom *(def)*
Dyn	Dynamic Range Compression	*(def)*
Att	Attenuation	-3 dB
		-6 dB

Table 3: Confusion matrix for the best parameter configuration in Experiment 1 including all augmentations (DB-MTC$^+$), averaged over three folds. The overall F-score is 0.89.

True \ Predicted	woo	bra	pia	voc	str
woodwinds (woo)	**0.92**	0.06	0.02	0.00	0.00
brass (bra)	0.20	**0.70**	0.07	0.00	0.03
piano (pia)	0.01	0.01	**0.97**	0.01	0.01
vocal (voc)	0.09	0.00	0.01	**0.82**	0.08
strings (str)	0.00	0.02	0.02	0.02	**0.94**

5.1.2 Evaluation Procedure

We systematically evaluate 96 combinations of data augmentation, spectrogram compression, and patch pre-processing methods as listed in Table 4. For each configuration, we perform three validation runs and report the mean F-score. In each run, we randomly split the dataset on file level into training (40%), validation (30%), and test set (30%). We use the additional augmented versions of the files for the training and validation sets and test only on the clean, non-augmented signals.

5.1.3 Results

In Table 4, we show the results of Experiment 1. We observe the highest F-score of 0.89 for a system using no compression of the mel-spectrogram (NO), frequency-based patch normalization (A), and a fully augmented training set. Independent of the applied data augmentation, the results show that a normalization of spectral patches before model training is very beneficial if no (NO) or only mild spectrogram compression (LC 1) is applied. In contrast, strong compression (LC 10000) elevates the results for systems without patch pre-processing (−) much closer to the regions with pre-processing: Strong spectrogram compression and the patch normalization techniques have a similar effect, with the latter tending to have an even greater impact. At least one of the methods should be considered for usage. The simple logarithmic compression strategies outperform the PCEN strategy, which we apply using default parameters. We assume that using a trainable PCEN front-end instead seems to be more promising for future work.

In Table 3, we show a confusion matrix for this experiment using the ideal parameter combination. While the piano class is recognized best, confusions mainly occur between vocal and woodwinds or strings, and between woodwinds and brass. Concerning the latter confusion, both families are wind instruments and, therefore, exhibit certain timbral similarity. Furthermore, the presence of french horns in both classes (as discussed in Section 3.3) might be problematic.

5.1.4 Baseline System

To compare the CNN-based results to a simple baseline system relying on standard audio features, we extract 20 mel-frequency cepstral coefficients (MFCC) per spectral patch $X_{p,:,:}$ and average them over the patch duration. This way, each spectral patch is represented by a 20-dimensional MFCC feature vector. We train a random forest classifier with 50 estimators obtaining an F-score of 0.75. This result—which could be further improved by using data augmentation and more diverse audio features—indicates that the benefit of our deep learning strategy over standard approaches is only weak when using datasets of limited size such as DB-MTC.

5.2 Experiment 2: Cross-Dataset Evaluation

In this experiment, we evaluate how well the CNN model generalizes to unseen datasets that represent different levels of timbral complexity. Ideally, we expect the model to learn spectro-temporal patterns that are unique to particular instrument families so that these patterns are recognized independent of a dataset's acoustic characteristics.

5.2.1 Evaluation Procedure

We split all three datasets DB-MTC, DB-SOL, DB-URMP and, additionally, a combination of them called DB-M/U/S, into individual training, validation, and test sets and perform cross-dataset evaluations. Concretely speaking, we train a model using training and validation sets taken from one dataset and evaluate using the test set of another dataset. Due to differences between the datasets, we restrict ourselves in this experiment to the three instrument families woodwinds, brass, and strings which are consistently present over all datasets. Consequently, we discard piano and vocal recordings from DB-MTC. Data augmentation and a comparison with the MFCC-based baseline system are not part of this experiment as we solely focus on the cross-dataset performance of the models.

We compare two approaches for splitting datasets into training, validation, and test sets. We either randomly select patches (patch-based) or split patches based on files (file-based) in order to avoid overfitting due to patches from the same file ending up in both the training and test sets. For the patch-based approach, we identify the smallest amount of available patches per class among the datasets. The smallest class is the brass class in DB-MTC with 4397 patches. Therefore, for the patch-based evaluation, we sample the same amount of patches from all other classes and datasets. We then use a split ratio of 40%–30%–30% to create training, validation, and test sets. Hence, each dataset finally consists of

Table 4: Mean F-scores for the parameter optimization on the DB-MTC dataset (Experiment 1) described in Section 5.1. The abbreviations for data augmentation methods (first four columns) are introduced in Table 2. The spectrogram compression methods LC 1, LC 10000, and PCEN, as well as the patch pre-processing methods −, A, B, C in the remaining columns are described in Section 4.2 and Section 4.3, respectively. The optimal parameter configuration (NO, A, full data augmentation) is highlighted using gray background color. The average standard deviation between F-scores over all three validation runs is 0.03 (min: 0.003, max: 0.15).

Augmentation Types				NO				LC 1				LC 10000				PCEN			
Noi	Imp	Dyn	Att	−	A	B	C	−	A	B	C	−	A	B	C	−	A	B	C
-	-	-	-	0.40	0.78	0.82	**0.85**	0.34	0.76	0.84	0.84	0.66	0.76	0.82	0.75	0.70	0.67	0.67	0.65
✓	✓	✓	✓	0.46	**0.89**	0.86	0.86	0.49	0.88	0.87	0.85	0.86	0.86	0.85	0.83	0.79	0.80	0.79	0.80
✓	-	-	-	0.49	**0.87**	0.86	0.85	0.49	**0.87**	0.84	0.86	0.81	0.82	0.82	0.82	0.75	0.76	0.75	0.75
-	✓	-	-	0.42	**0.86**	0.85	0.85	0.42	0.84	0.85	0.85	0.84	0.83	0.81	0.83	0.79	0.77	0.79	0.78
-	-	✓	-	0.55	0.81	0.83	0.83	0.45	0.83	0.83	**0.85**	0.78	0.83	0.79	0.81	0.75	0.74	0.72	0.76
-	-	-	✓	0.35	0.81	0.84	**0.87**	0.36	0.81	0.83	0.82	0.77	0.78	0.78	0.81	0.74	0.72	0.71	0.74

13191 patches. We create the fourth dataset DB-M/U/S by equally sampling patches from the other datasets.

For the file-based approach, we split each dataset using the same ratio of 40%–30%–30% on a file level, i. e., patches from one file will exclusively end up in one of the subsets. Here, no further steps are taken to balance out the amount of patches. Since this procedure leads to class imbalance, we use class weights as discussed in Section 4.4. The file-based version of DB-M/U/S is generated by accumulating all subsets over all datasets. Table 1 summarizes all datasets used in this experiment. For model training, we pick the parameters that lead to the best result in Experiment 1, namely no compression (NO) and patch pre-processing method C. We do not include any augmentations in this experiment.

5.2.2 Results

Table 5 shows the results for the cross-dataset evaluation on the patch-based data split. Due to the split strategy, the classifier overfits to the training set and naturally achieves high F-scores on the corresponding test set when patches are randomly mixed. This overfitting effect is supported by the fact that adding additional training data from a different dataset in DB-M/U/S even degrades the performance for testing on DB-MTC and DB-SOL.

Table 6 shows the results for the file-based data split. Due to its large size, DB-SOL has the highest impact on the model performance in the mixed dataset DB-M/U/S. When comparing the performance for training and testing on DB-MTC, the F-score drops by 0.12 for the file-based split strategy. This confirms our expectations since the DB-MTC dataset has a small number of files per class and a large variance of file durations.

As a general observation for both split strategies, the CNN approach for instrument family recognition shows only a limited capability to generalize well towards unseen data, which becomes apparent for all cross-dataset combinations in both tables (high values on the diagonal).

Table 5: Resulting F-scores for cross-dataset evaluation using patch-based dataset split. The rows and columns of the table indicate the training and test sets for each configuration, respectively.

Training \ Test	DB-MTC	DB-URMP	DB-SOL	DB-M/U/S
DB-MTC	**0.96**	0.70	0.62	0.77
DB-URMP	0.49	**0.96**	0.61	0.70
DB-SOL	0.64	0.78	**0.95**	0.79
DB-M/U/S	0.95	**0.97**	0.91	0.94

Table 6: Resulting F-scores for cross-dataset evaluation using file-based dataset split.

Training \ Test	DB-MTC	DB-URMP	DB-SOL	DB-M/U/S
DB-MTC	**0.84**	0.60	0.68	0.68
DB-URMP	0.51	**0.92**	0.69	0.70
DB-SOL	0.69	0.74	**0.99**	0.94
DB-M/U/S	0.89	0.95	**0.99**	0.98

6. CONCLUSIONS

In this paper, we investigated a state-of-the-art convolutional neural network model for automatic instrument family recognition in Western classical music recordings. Focusing on monotimbral, polyphonic recordings, we showed that increasing the amount of training data via augmentation techniques leads to improved classification performance. We also found that pre-processing is of central importance for achieving a good system. Combining patch normalization with dynamic compression or per-channel energy normalization does not further improve the results, but these techniques may compensate the effect of patch normalization to some degree. Given that a simple MFCC-based baseline system already achieves good performance in Experiment 1, the possible superiority of more complex data-driven methods such as CNNs needs to be assessed carefully. In a cross-dataset experiment, we further tested how well the CNN model generalizes towards unseen data. Our results indicate that current CNN models lack generalization capability across different datasets, thus indicating the need for applying further optimization methods such as domain adaptation [5].

Acknowledgments: This work has been supported by the German Research Foundation (AB 675/2-1, MU 2686/11-1). The International Audio Laboratories

Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

7. REFERENCES

[1] Rachel M. Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. MedleyDB: A Multitrack Dataset for Annotation-Intensive MIR Research. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, pages 155–160, Taipei, Taiwan, 2014.

[2] Juan Bosch, Jordi Janer, Ferdinand Fuhrmann, and Perfecto Herrera. A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 559–564, Porto, Portugal, 2012.

[3] Jana Eggink and Guy J. Brown. Instrument recognition in accompanied sonatas and concertos. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 217–220, 2004.

[4] Ferdinand Fuhrmann. *Automatic musical instrument recognition from polyphonic music audio signals*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2012.

[5] Shayan Gharib, Konstantinos Drossos, Emre Cakir, Dmitriy Serdyuk, and Tuomas Virtanen. Unsupervised adversarial domain adaptation for acoustic scene classification. In *Proceedings of the 3rd Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, pages 138–142, Surrey, UK, 2018.

[6] Juan Gómez, Jakob Abeßer, and Estefanía Cano. Jazz solo instrument classification with convolutional neural networks, source separation, and transfer learning. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 577–584, Paris, France, 2018.

[7] Siddharth Gururani, Cameron Summers, and Alexander Lerch. Instrument activity detection in polyphonic music using deep neural networks. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 569–576, Paris, France, 2018.

[8] Yoonchang Han, Jae-Hun Kim, and Kyogu Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *CoRR*, abs/1605.09507, 2016.

[9] Yun-Ning Hung and Yi-Hsuan Yang. Frame-level instrument recognition by timbre and pitch. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 135–142, Paris, France, 2018.

[10] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade, this book is an outgrowth of a 1996 NIPS Workshop*, pages 9–50, London, UK, 1998. Springer-Verlag.

[11] Bochen Li, Xinzhao Liu, Karthik Dinesh, Zhiyao Duan, and Gaurav Sharma. Creating a musical performance dataset for multimodal music analysis: Challenges, insights, and applications. *CoRR*, abs/1612.08727, 2016.

[12] Peter Li, Jiyuan Qian, and Tian Wang. Automatic instrument recognition in polyphonic music using convolutional neural networks. *CoRR*, abs/1511.05520, 2015.

[13] Vincent Lostanlen, Joakim Andén, and Mathieu Lagrange. Extended playing techniques: the next milestone in musical instrument recognition. In *Proceedings of the 5th International Conference on Digital Libraries for Musicology (DLfM)*, pages 1–10, Paris, France, 2018.

[14] Vincent Lostanlen and Carmine-Emanuele Cella. Deep convolutional networks on the pitch spiral for music instrument recognition. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 612–618, New York, NY, USA, 2016.

[15] Vincent Lostanlen, Justin Salamon, Mark Cartwright, Brian McFee, Andrew Farnsworth, Steve Kelling, and Juan Pablo Bello. Per-Channel Energy Normalization: Why and How. *IEEE Signal Processing Letters*, 26(1):39–43, 2019.

[16] Matthias Mauch and Sebastian Ewert. The audio degradation toolbox and its application to robustness evaluation. In *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, Curitiba, Brazil, 2013.

[17] Taejin Park and Taejin Lee. Musical instrument sound classification with deep convolutional neural network using feature fusion approach. *CoRR*, abs/1512.07370, 2015.

[18] Jordi Pons and Xavier Serra. Designing efficient architectures for modeling temporal features with convolutional neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2472–2476, New Orleans, USA, 2017.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[20] Takumi Takahashi, Satoru Fukayama, and Masataka Goto. Instrudive: A music visualization system based on automatically recognized instrumentation. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 561–568, Paris, France, 2018.

[21] John Thickstun, Zaid Harchaoui, and Sham M. Kakade. Learning features of music from scratch. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, pages 1–14, Palais des Congrès Neptune, Toulon, France, 2017.

[22] Yuxuan Wang, Pascal Getreuer, Thad Hughes, Richard F. Lyon, and Rif A. Saurous. Trainable frontend for robust and far-field keyword spotting. *CoRR*, abs/1607.05666, 2016.

A BI-DIRECTIONAL TRANSFORMER FOR MUSICAL CHORD RECOGNITION

Jonggwon Park **Kyoyun Choi** **Sungwook Jeon**
Dokyun Kim **Jonghun Park**
Department of Industrial Engineering & Center for Superintelligence,
Seoul National University, Seoul, Republic of Korea
{jayg996, andyhome1907, wookee3, kdk01, jonghun}@snu.ac.kr

ABSTRACT

Chord recognition is an important task since chords are highly abstract and descriptive features of music. For effective chord recognition, it is essential to utilize relevant context in audio sequence. While various machine learning models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have been employed for the task, most of them have limitations in capturing long-term dependency or require training of an additional model.

In this work, we utilize a self-attention mechanism for chord recognition to focus on certain regions of chords. Training of the proposed bi-directional Transformer for chord recognition (BTC) consists of a single phase while showing competitive performance. Through an attention map analysis, we have visualized how attention was performed. It turns out that the model was able to divide segments of chords by utilizing adaptive receptive field of the attention mechanism. Furthermore, it was observed that the model was able to effectively capture long-term dependencies, making use of essential information regardless of distance.

1. INTRODUCTION

The goal of chord recognition task is to output a sequence of time-synchronized chord labels when a raw audio recording of music is given as input. Chords are highly abstract and descriptive features of music that can be used for a variety of musical purposes, including automatic lead-sheet creation for musicians, cover song identification, key classification and music structure analysis [4, 24, 26]. Since manual chord annotation is labor intensive, time consuming and requires expert knowledge, automatic chord recognition system has been an active research area within the music information retrieval community.

Automatic chord recognition is challenging due to the fact that 1) not all the notes played are necessarily related to the chord of the moment and 2) simple one-hot encoding of chord labels cannot represent the inherent relationship between different chords. Most traditional automatic chord recognition systems consist of three parts: feature extraction, pattern matching and chord sequence decoding. The most common strategy was to rely on hidden Markov models (HMMs) [3] for sequence decoding. Recently, many studies have explored various deep neural networks such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs) [23] for chord recognition.

Recently, a novel attention-based network architecture named Transformer was proposed in [33]. It performs well without any recurrence or convolution and the use of Transformer has become popular in various domains. For example, a bi-directional Transformer model called BERT achieved state-of-the-art results on eleven natural language processing (NLP) tasks [10]. In the domain of music, [13] applied Transformer to a music generation task and succeeded in creating music with complex and repetitive structure.

In this paper we propose BTC (Bi-directional Transformer for Chord recognition). In contrast to the other chord recognition models that depend on training of separate feature extractors or adopting additional decoders such as HMMs or Conditional Random Fields (CRFs) [22], BTC requires only a single training phase while being able to obtain results comparable to them. We also visualize how the model works through attention maps. The attention maps demonstrate that BTC is able to 1) divide segments of chords by utilizing its adaptive receptive field and 2) capture long-term dependencies.

2. RELATED WORK

2.1 Automatic Chord Recognition

In the past, most automatic chord recognition systems were divided into three parts: feature extraction, pattern matching and chord sequence decoding. After applying transformation such as short-time Fourier transform or constant-q transform (CQT) to an input audio signal, features are extracted from the resulting time-frequency domain. Some examples of such hand-crafted features include chroma

vectors and the "Tonnetz" [11] representation. For pattern matching and chord sequence decoding, Gaussian mixture models with feature smoothing [6, 7] and HMMs [28, 32] have been the most popular choices, respectively.

With the recent wide acceptance of deep learning in research communities, there have been many studies applying it to chord recognition task in various ways. The very first deep-learning-based chord recognition system was proposed by [14] where they trained a CNN for major-minor chord classification. Attempts to apply deep learning to feature extraction include [16] and [19], where the former employed a CNN to extract Tonnetz features from audio data and the latter adopted a deep neural network (DNN) to compute chroma features. CNN and HMM were combined for chord recognition in [15] and [35].

In addition to CNN, another popular network architecture for chord recognition is RNN. [5] and [31] explored an RNN as chord sequence decoding method, relying on deep belief network and a DNN, respectively. Another branch of RNN-based chord recognition systems utilize a language model which predicts only the sequence of chords without considering their durations. This might be helpful when the number of chord labels is large (e.g. large vocabulary type, explained in Section 4.1). A large-scale study of language models for chord prediction was conducted in [18]. Without audio data, the authors trained just a language model with the chord progression data only and showed that RNNs outperformed N-gram models. In their succeeding work [21], they combined the RNN-based harmonic language model with a chord duration model to complete the chord recognition task.

Another RNN-based approach is presented in [34] which trained a CNN feature extractor with large MIDI (Musical Instrument Digital Interface) data and combined BLSTM (Bi-directional Long-Short Term Memory) with CRF for sequence decoder. This BLSTM-CRF model achieved good performance but has a drawback that its training procedure involves complex MIDI pre-training. The model that we propose, on the other hand, is much simpler to train.

2.2 Attention-based Models

The attention mechanism, first introduced by [2], can be described as computing an output vector when query, key and value vectors are given. In sequence modelling tasks such as machine translation, query and key correspond to certain elements of the target sequence and the source sequence respectively. Each key has its own value. The output is computed as a weighted sum of the values where the weights are computed from the query and key. Self-attention refers to the case when query, key and value are computed from the same input.

Transformer is an attention-based network that relies on attention mechanism only and does not include recurrent or convolutional architecture. Utilizing multi-head attention together with position-wise fully-connected feed-forward network, it showed significantly faster training speed and achieved better performance than recurrent or convolutional networks for translation tasks.

Transformer used scaled dot-product as an attention function:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_K}})V \quad (1)$$

where Q, K and V are matrices of query, key and value vectors respectively, and d_K is the dimension of key.

The use of Transformer has become very popular, achieving the state-of-the-art results in various domains. A well-known example is bi-directional encoder representations from Transformers (BERT) [10]. BERT is a pre-training model based on masked language model for language representations that achieved state-of-the-art results on eleven NLP tasks. In the domain of music, [13] proposed music Transformer for symbolic music generation. Music Transformer employed relative attention to capture long-term structure effectively, which resulted in music compositions that are both qualitatively and quantitatively better structured than existing music generation models.

3. BI-DIRECTIONAL TRANSFORMER FOR CHORD RECOGNITION

3.1 Bi-directional Transformer

Making use of appropriate surrounding frames is essential for successful chord recognition [7, 8]. This context-dependent characteristic of the task is the motivation for applying the self-attention mechanism. With some modification to the original Transformer architecture, we present a bi-directional Transformer for chord recognition (BTC). [1]

The structure of BTC is shown in Figure 1. The model consists of bi-directional multi-head self-attentions, position-wise convolutional blocks, a positional encoding, layer normalization [1], dropout [30] and fully-connected layers. The model takes a CQT feature of 10 second audio signal (Section 4.1) as input. The results of adding positional encoding are given as input to two self-attention blocks with different masking directions, indicated as dotted boxes in Figure 1(b). The outputs are concatenated and are fed into a fully-connected layer so that the output size is the same as the original input. A stack of N bi-directional self-attention layers is followed by another fully-connected layer that outputs logit values. The size of the logit values is the same as the number of chord labels. These logits are used to predict the chord and calculate the loss.

The loss function is a negative log-likelihood and all the model parameters are trained to minimize the loss given by the following equation (2).

$$L = -\sum_{t=1}^{T}\sum_{c\in V} y_c(t) log(\hat{y_c}(t)) \quad (2)$$

T is the number of total time frames and V is the chord label set. $y_c(t)$ is 1 if the reference label at time t is c and 0 otherwise. $\hat{y_c}(t)$ is the output of the model, representing the probability of the chord at time t being c.

[1] https://github.com/jayg996/BTC-ISMIR19

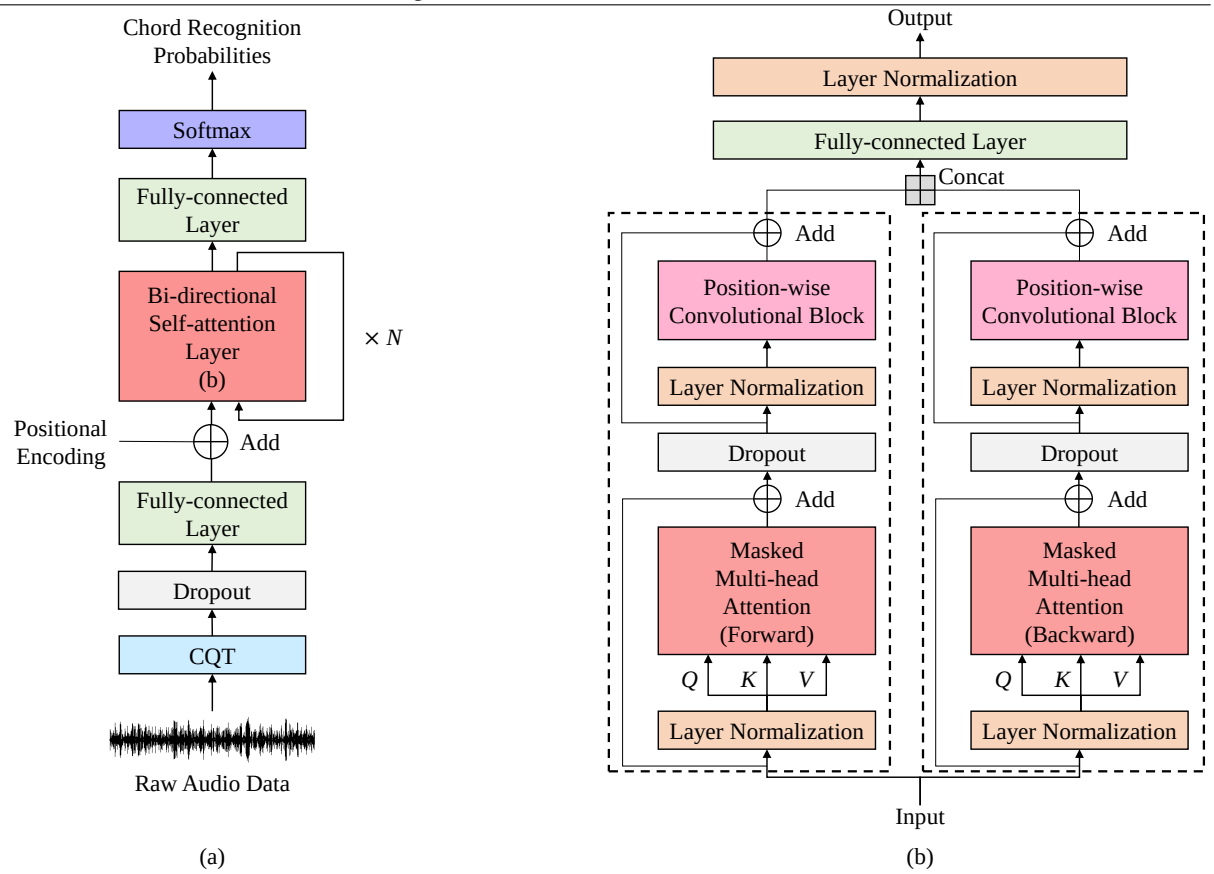


Figure 1. Structure of BTC. (a) shows the overall network architecture and (b) describes the bi-directional self-attention layer in detail. Dotted boxes indicate self-attention blocks.

3.1.1 Bi-directional Multi-head Self-attention

BTC employs multi-head self-attention as in the original Transformer. For each time frame, the input features are split into n_h pieces and provided as input to the multi-head self-attention with the number of heads, n_h. Given I as an input matrix, the multi-head self-attention can be computed as (3):

$$Multihead = Concat(head_1, ..., head_{n_h})W_O \quad (3)$$

$Q_j = (IW_Q)_j, K_j = (IW_K)_j$ and $V_j = (IW_V)_j$ are given as input to the attention function (1) to produce $head_j$ for $j = 1, ..., n_h$. W_Q, W_K and W_V are fully-connected layers that project the input to the dimension of Q, K and V, respectively. W_O is also a fully-connected layer that projects the concatenated output of dimension $(n_h \times d_{V_j})$ to the dimension of the final output. Dropout is applied to the softmax output weights when computing each $head_j$.

In BTC, self-attention can be interpreted as determining how much attention to apply to the value of the key time frame when inferring the chord of the query time frame. To prevent the loss of information due to the attention being performed to the entire input at once, we employed bi-directional masking. The forward / backward direction refers to masking all the preceding / succeeding time frames. The same masked multi-head attention module as the Transformer decoder was adopted. The bi-directional structure enables BTC to fully utilize the context before and after the target time frame.

Since the multi-head attention is performed on every time frame in the sequence, information about the order of the sequence is lost. We employed the same solution proposed by Transformer to address this issue: adding positional encoding results to the input, which are obtained by applying sinusoidal functions to each position. Since relative positions between two frames can be expressed as a linear function of the encodings, positional encoding helps the model learn to apply attention via relative positions.

3.1.2 Position-wise Convolutional Block

To utilize the adjacent feature information in a time frame, we replaced the position-wise fully-connected feed-forward network from the original Transformer architecture with a position-wise convolutional block. The position-wise convolutional block consists of a 1D convolution layer, a ReLU (Rectified Linear Unit) activation function and a dropout layer, where the whole sequence of layers is repeated n_C times. Input and output channel size were identical to keep the feature size and sequence length constant. With the position-wise convolutional block, we anticipate to search the boundary and smooth the chord sequence by exploring adjacent information at each time frame.

3.2 Self-attention in Chord Recognition

For chord recognition, it is important to utilize not only the information from the target time frame but also from other related frames, which we call the context. The network architectures such as CNNs or RNNs can also explore the context, but self-attention is more suitable for the task because of the following reasons.

First, self-attention has selective usage of attention. In other words, the receptive field can be adaptive unlike CNNs where the kernel size is fixed. For example, assume that the labels for 16 frames are Cs for the first four frames, Gs and Fs for the next eight frames and Cs for the last four frames (see Figure 2). Consider the situation of recognizing Gs in frames 5 to 8. As for a CNN with kernel size of 3, when recognizing the chord of frame 7, the receptive field (frame 6 to 8) would be informative enough since all the frames contain the same chord. However, when inferring frame 5, the receptive field of frame 4 to 6 contains not only G but also C. With self-attention, on the other hand, the model can pay attention to the section of frame 5 to 8 regardless of the target frame's position.

Another advantage of attention mechanism is its ability to capture long-term dependency effectively. RNNs can also utilize distant information but direct access is not possible. For CNNs, there are two ways to access distant frames: by stacking layers in depth or by increasing the kernel size. The former has the same drawback as RNNs and the latter has the disadvantage that the weight sharing becomes less effective. Unlike these, self-attention has direct access to other frames no matter how far they are. Specifically, when recognizing the chord of frame 13, performing attention to first four frames would be helpful since they all contain C. With RNNs or deep CNNs, information that the first four frames were C would inevitably be diluted while passing through frames 5 to 12.

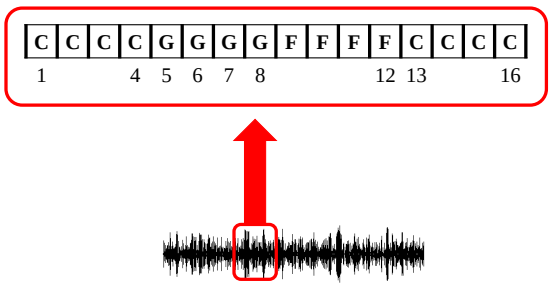


Figure 2. Chord sequence example

4. EXPERIMENTS

4.1 Data and Preprocessing

BTC and other baseline models were evaluated on the following datasets. A subset of 221 songs from Isophonics [2]: 171 songs by the Beatles, 12 songs by Carole King, 20 songs by Queen and 18 songs by Zweieck; Robbie Williams [12]: 65 songs by Robbie Williams; and a subset of 185 songs from UsPop2002 [3]. These datasets consist of label files that specify the start time, end time and type of the chord. Due to copyright issue, these datasets do not include audio files. The audio files used in this work were collected from online music service providers (e.g. Melon [4]), which do not always provide the same audio files corresponding to the songs in the datasets. Since it was not possible to get exactly the same audio files, there were subtle differences in the chord start time of the label file and audio file. Accordingly we manually matched the labels to the audio file by shifting the whole label file back and forth, which resulted in no more than adding or deleting some "No chord" labels.

Each 10-second-long audio signal (consecutive signals overlapping 5 seconds) was processed at the sampling rate of 22,050Hz using CQT with 6 octaves starting from C1, 24 bins per octave, and the hop size of 2048 [34]. The CQT features were transformed to log amplitude with $S_{log} = ln(S + \epsilon)$ where S represents the CQT feature and ϵ is an extremely small number. After that, global z-normalization was applied with mean, variance from the training data.

Pitch augmentation was also employed to the audio file with pyrubberband [5] package and labels were changed with pitch variation. Pitch augmentation between -5 $\sim$ +6 semitones were applied to all the training data.

Two different label types were used: maj-min and large vocabulary. The maj-min label type consists of 25 chords (12 semitones $\times$ {maj, min} and "No chord") [20]. The large vocabulary label type consists of 170 chords (12 semitones $\times$ {maj, min, dim, aug, min6, maj6, min7, minmaj7, maj7, 7, dim7, hdim7, sus2, sus4} and "X chord : the unknown chord", "No chord") [25]. From the label files, we extracted the chord that matches the time frame of input feature and transformed it to the appropriate label type.

4.2 Evaluation Metric

The evaluation metric was weighted chord symbol recall (WCSR) score and 5-fold cross validation was applied to the entire data. When separating the evaluation data from the training data, there was no song included in both. The WCSR score can be computed as (4), where t_c is the duration of correctly classified chord segments and t_a is the duration of the entire chord segments.

$$WCSR = \frac{t_c}{t_a} \times 100(\%) \tag{4}$$

Scores were computed with mir_eval [27]. Root and Maj-min scores were used for the maj-min label type. Root, Thirds, Triads, Sevenths, Tetrads, Maj-min and MIREX scores were used for the large vocabulary label type. To calculate the score with mir_eval, the chord recognition results were converted into label files.

[2] http://isophonics.net/datasets

[3] https://github.com/tmc323/Chord-Annotations

[4] http://www.melon.com

[5] https://github.com/bmcfee/pyrubberband

Model	maj-min label type		large vocabulary label type						
	Root	Maj-min	Root	Thirds	Triads	Sevenths	Tetrads	Maj-min	MIREX
CNN	83.6±1.3	81.8±1.2	83.5±1.4	80.4±1.2	75.5±0.6	71.5±1.9	65.2±1.0	81.9±1.4	79.8±0.7
CNN+CRF [20]	**84.0**±1.3	**83.1**±1.4	**83.7**±1.5	**81.1**±1.4	**76.3**±0.8	71.3±1.9	**65.7**±1.6	82.1±1.5	**81.8**±1.1
CRNN [25]	83.4±0.8	82.3±0.9	82.9±1.1	80.1±1.0	75.3±0.7	71.3±1.9	65.2±0.9	81.5±1.3	79.9±0.8
CRNN+CRF	83.3±0.8	82.3±1.0	82.7±1.2	79.7±0.9	74.8±0.5	69.5±2.0	63.9±1.0	80.7±1.4	80.2±1.0
BTC	83.8±1.0	82.7±1.0	83.5±1.2	80.8±1.0	75.9±0.5	**71.8**±1.7	65.5±0.9	**82.3**±1.2	80.8±0.9
BTC+CRF	83.9±1.0	**83.1**±1.1	83.5±1.2	80.7±1.1	75.7±0.5	70.7±2.0	64.8±1.1	81.7±1.4	81.4±0.9

Table 1. WCSR scores averaged over the same 5 folds. Numbers next to the scores denote the standard deviations.

4.3 Results

Specific hyperparameters of BTC are summarized in Table 2. The hyperparameters with the best validation performance were obtained empirically after applying in 5-fold cross validation. Adam optimizer [17] was used with initial learning rate of 10^{-4}. Learning rate was decayed with rate 0.95 when validation accuracy did not increase. Training was stopped if the validation accuracy did not improve for over 10 epochs.

Since existing studies of chord recognition were evaluated on different datasets, it is difficult to say that a particular model is the state-of-the-art. Among the models that were trainable with our datasets, we chose three baseline models with good performance: CNN, CNN+CRF and CRNN. CNN is a VGG [29]-style CNN and CNN+CRF has an additional CRF decoder [20]. CRNN is a combination of CNN and gated recurrent unit [9], named "CR2" in [25]. The input was preprocessed as mentioned in Section 4.1 for BTC and CRNN. For CNN+CRF and CNN, a single label was estimated with a patch of 15 time frames, in a similar way to [20].

Table 1 shows the performance comparison results of the baseline models and BTC for two label types. The best value for each metric is represented in bold. Among the models without a CRF decoder, BTC showed the best performance for all metrics. Including models with a CRF decoder, CNN+CRF obtained the best result in most of the metrics. Still, BTC shows comparable performance to CNN+CRF, performing better in Sevenths and Maj-min metrics for the large vocabulary label type.

The main purpose of training a CRF decoder is to smooth the predicted chord sequences that are often fragmented. The performances of CRNN+CRF and BTC+CRF are also presented in Table 1 for comparison. Performance improvements due to the introduction of CRFs are evident in CNN but not in BTC and CRNN. This indicates that outputs of CNN were fragmented and an additional decoder training is necessary for better performance. On the other hand, BTC and CRNN can be trained with only CQT features and chord labels. That is, BTC requires only a single training phase while achieving the performance comparable to that of CNN+CRF.

Bi-directional self-attention layer	layer repetition (N)	{1, 2, 4, **8**, 12}
	self-attention heads (n_h)	{1, 2, **4**}
	dimension of Q, K, V and all the hidden layers	{64, **128**, 256}
Position-wise convolutional block	block repetition (n_C)	**2**
	kernel size	**3**
	stride	**1**
	padding size	**1**
Dropout	dropout probability	{**0.2**, 0.3, 0.5}

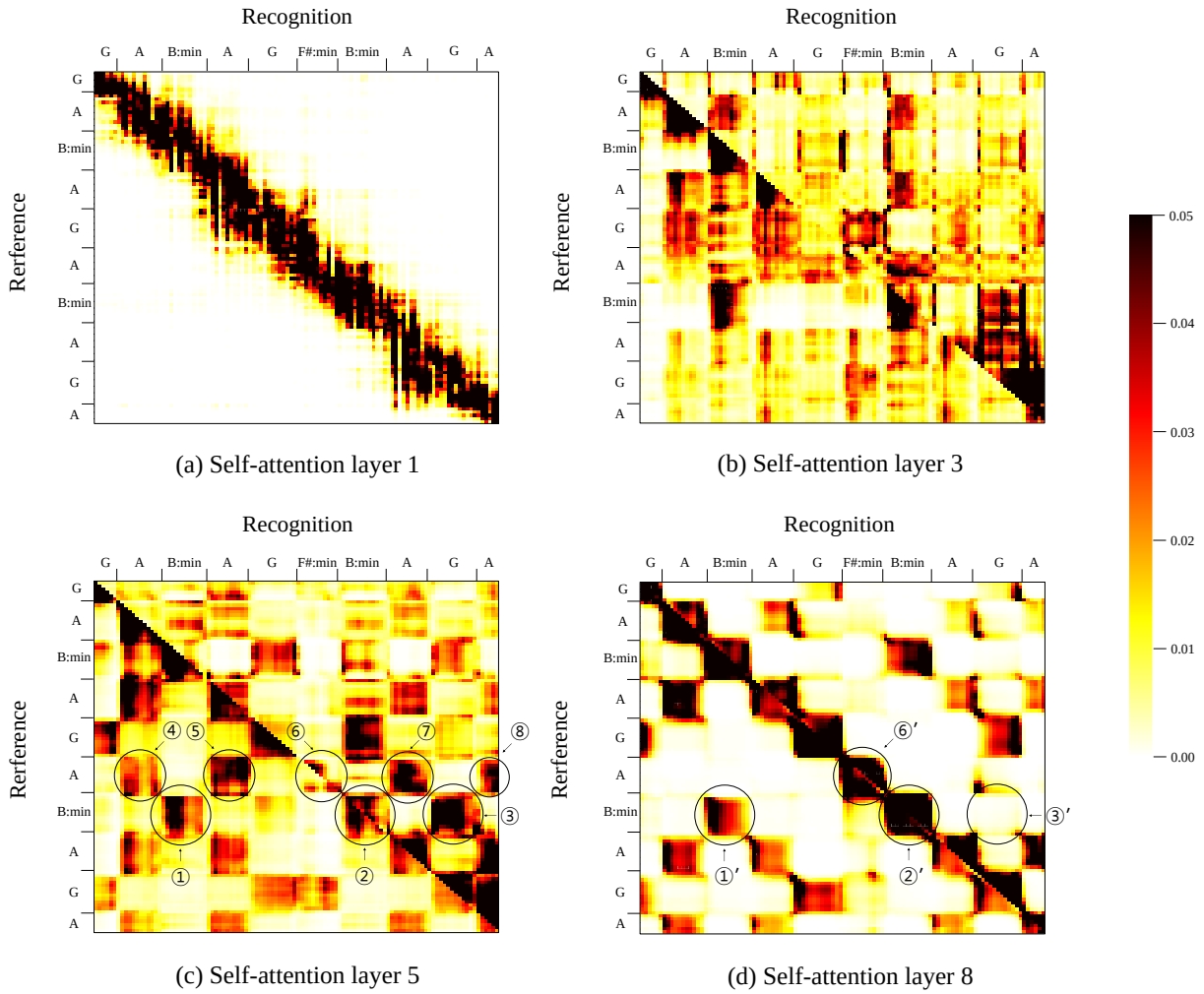
Table 2. Hyperparameters of BTC. Hyperparameters with the best validation performance are shown in bold.

4.4 Attention Map Analysis

Attention maps demonstrate that each self-attention layer has different characteristics. Figure 3 shows the attention map of self-attention layers 1, 3, 5 and 8, trained with the maj-min label type. The lower / upper triangle of each attention map represents the attention probability of the forward / backward direction self-attention layer. The labels of the vertical axis and the horizontal axis are the reference chord and the chord recognition result of the target time frame, respectively. The cell of i-th row and j-th column represents the attention probability to the j-th time frame when inferring the chord of the i-th time frame.

At the first self-attention layer, only neighboring frames are used to construct the representation of the target frame. For the third layer, the attention is widely spread over all time frames, yet still with higher probabilities for nearby frames than distant frames. At the fifth layer, several adjacent time frames form a group, which appears in a rectangular region in the attention map. This means that the model divides the whole input into some sections, which is possible due to the adaptive receptive field. The network focuses only on a few important sections to identify the target frame, regardless of the distance between section and the frame. Unlike the fifth layer, attention is more dense in certain regions at the eighth layer. In particular, the boundary of the high probability region matches that of the final recognition result.

Specifically, at the fifth layer in Figure 3(c), the reference chord for region ② is B:min. Region ① shares the same reference chord B:min and the network assigns high attention probabilities to region ① for time frames in region ②. This phenomenon is similar in layer 8 between ①′ and ②′(Figure 3(d)), which results in the correct final chord recognition of B:min. In contrast, for region ③ where the reference chord is G, the attention probability is



(a) Self-attention layer 1

(b) Self-attention layer 3

(c) Self-attention layer 5

(d) Self-attention layer 8

Figure 3. The figures represent the probability values of the attention of self-attention layers 1, 3, 5 and 8 respectively. The layers that best represent the different characteristics were chosen. The input audio is the song "Just A Girl" (0m30s ∼ 0m40s) by No Doubt from UsPop2002, which was in evaluation data.

high at layer 5 but not for region ③′ at layer 8. This can be attributed to G and B:min sharing two notes in common, since G and B:min consist of (G,B,D) and (B,D,F#) respectively. In other words, attention at layer 5 can be seen as attention to partial features of chords sharing the same notes. None the less, the final recognition result after the last layer is not G but B:min. This is possible because of the multi-head attention structure: the other heads might lower the attention probability even if the attention to a wrong chord is active, leading to the correct result.

On the other hand, there are cases where the recognition results are wrong in a similar situation. The reference chord for regions ⑥ and ⑥′ is A. At layer 5, the attention mechanism seems to work well with high attention probabilities to region ④,⑤,⑦ and ⑧, where the reference chords are all As. However, the attention to those regions cannot be seen at the last layer, and the final recognition result is not A but F#:min. This recognition failure can be regarded as a result of two notes of F#:min (F#,A,C#) overlapping with A (A,C#,E).

To summarize, for each target frame in the input audio, the model uses only neighboring frames at first. At the middle layers, the model gradually broadens the receptive field and selectively focuses on time frames with characteristics similar to that of the target frame. Finally, at the last layer, the attention is performed on only essential information for chord recognition.

5. CONCLUSION

In this paper, we presented bi-directional Transformer for chord recognition (BTC). To the best of our knowledge, this paper was the first attempt to apply Transformer to chord recognition. The self-attention mechanism was appropriate for the task that attempts to capture long-term dependency by effectively exploring relevant sections. BTC has an advantage in that its training procedure is simple and it showed results competitive to other models in most of the evaluation metrics. Through the attention map analysis, it turned out that each self-attention layer had different characteristics and that the attention mechanism was effective in identifying sections of chords that were crucial for chord recognition.

6. ACKNOWLEDGEMENTS

This work was supported by Kakao and Kakao Brain corporations.

7. REFERENCES

[1] L. J. Ba, R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint*, arXiv:1607.06450, 2016.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR), Conference Track Proc.*, San Diego, CA, USA, 2015.

[3] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

[4] J. P. Bello. Chord segmentation and recognition using em-trained hidden markov models. In *Proc. of the 8th International Society for Music Information Retrieval Conference (ISMIR)*, pages 239–244, Vienna, Austria, 2007.

[5] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Audio chord recognition with recurrent neural networks. In *Proc. of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, pages 335–340, Curitiba, Brazil, 2013.

[6] T. Cho. *Improved Techniques for Automatic Chord Recognition from Music Audio Signals*. PhD thesis, New York University, 2014.

[7] T. Cho and J. P. Bello. A feature smoothing method for chord recognition using recurrence plots. In *Proc. of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, pages 651–656, Miami, Florida, USA, 2011.

[8] T. Cho and J. P. Bello. On the relative importance of individual components of chord recognition systems. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 27(2):477–492, 2014.

[9] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint*, arXiv:1412.3555, 2014.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*, arXiv:1810.04805, 2018.

[11] L. Euler. *Tentamen novae theoriae musicae ex certissimis harmoniae principiis dilucide expositae*. ex typographia Academiae scientiarum, 1739.

[12] B. Di Giorgi, M. Zanoni, A. Sarti, and S. Tubaro. Automatic chord recognition based on the probabilistic modeling of diatonic modal harmony. In *Proc. of the 8th International Workshop on Multidimensional Systems*, Erlangen, Germany, 2013.

[13] C.-Z. Anna Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman, and D. Eck. Music transformer: Generating music with long-term structure. *arXiv preprint*, arXiv:1809.04281, 2018.

[14] E. J. Humphrey and J. P. Bello. Rethinking automatic chord recognition with convolutional neural networks. In *11th International Conference on Machine Learning and Applications(ICMLA)*, pages 357–362, Boca Raton, FL, USA, 2012.

[15] E. J. Humphrey and J. P. Bello. Four timely insights on automatic chord estimation. In *Proc. of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 673–679, Málaga, Spain, 2015.

[16] E. J. Humphrey, T. Cho, and J. P. Bello. Learning a robust tonnetz-space transform for automatic chord recognition. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing(ICASSP)*, pages 453–456, Kyoto, Japan, 2012.

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR), Conference Track Proc.*, San Diego, CA, USA, 2015.

[18] F. Korzeniowski, D. R. W. Sears, and G. Widmer. A large-scale study of language models for chord prediction. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing(ICASSP)*, pages 91–95, Calgary, AB, Canada, 2018.

[19] F. Korzeniowski and G. Widmer. Feature learning for chord recognition: The deep chroma extractor. In *Proc. of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 37–43, New York City, USA, 2016.

[20] F. Korzeniowski and G. Widmer. A fully convolutional deep auditory model for musical chord recognition. In *26th IEEE International Workshop on Machine Learning for Signal Processing, (MLSP)*, pages 1–6, Vietri sul Mare, Salerno, Italy, 2016.

[21] F. Korzeniowski and G. Widmer. Improved chord recognition by combining duration and harmonic language models. In *Proc. of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 10–17, Paris, France, 2018.

[22] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of*

the 18th International Conference on Machine Learning (ICML 2001), Williams College, pages 282–289, Williamstown, MA, USA, 2001.

[23] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[24] K. Lee. Identifying cover songs from audio using harmonic representation. *MIREX 2006*, pages 36–38, 2006.

[25] B. McFee and J. P. Bello. Structured training for large-vocabulary chord recognition. In *Proc. of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 188–194, Suzhou, China, 2017.

[26] J. Pauwels, F. Kaiser, and G. Peeters. Combining harmony-based and novelty-based approaches for structural segmentation. In *Proc. of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, pages 601–606, Curitiba, Brazil, 2013.

[27] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis. Mir_eval: A transparent implementation of common mir metrics. In *Proc. of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, pages 367–372, Taipei, Taiwan, 2014.

[28] A. Sheh and D. P. W. Ellis. Chord segmentation and recognition using em-trained hidden markov models. In *Proc. of the 4th International Society for Music Information Retrieval Conference (ISMIR)*, Baltimore, Maryland, USA, 2003.

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations (ICLR), Conference Track Proc.*, San Diego, CA, USA, 2015.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[31] S.Sigtia, N. Boulanger-Lewandowski, and S.Dixon. Audio chord recognition with a hybrid recurrent neural network. In *Proc. of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 127–133, Málaga, Spain, 2015.

[32] Y. Ueda, Y. Uchiyama, T. Nishimoto, N. Ono, and S. Sagayama. Hmm-based approach for automatic chord detection using refined acoustic features. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing(ICASSP)*, pages 5518–5521, Dallas, Texas, USA, 2010.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 6000–6010, Long Beach, CA, USA, 2017.

[34] Y. Wu and W. Li. Automatic audio chord recognition with midi-trained deep feature and BLSTM-CRF sequence decoding model. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 27(2):355–366, 2019.

[35] X. Zhou and A. Lerch. Chord detection using deep learning. In *Proc. of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 52–58, Málaga, Spain, 2015.

SAMBASET: A DATASET OF HISTORICAL SAMBA DE ENREDO RECORDINGS FOR COMPUTATIONAL MUSIC ANALYSIS

Lucas S. Maia[1,2], **Magdalena Fuentes**[3,2], **Luiz W. P. Biscainho**[1], **Martín Rocamora**[4], **Slim Essid**[2]
[1] Federal University of Rio de Janeiro, Brazil
[2] LTCI, Télécom Paris, Institut Polytechnique de Paris, France
[3] L2S, CNRS–Université Paris-Sud–CentraleSupélec, France
[4] Universidad de la República, Uruguay
lucas.maia@smt.ufrj.br

ABSTRACT

In the last few years, several datasets have been released to meet the requirements of "hungry" yet promising data-driven approaches in music technology research. Since, for historical reasons, most investigations conducted in the field still revolve around music of the so-called "Western" tradition, the corresponding data, methodology and conclusions carry a strong cultural bias. Music of non-"Western" background, whenever present, is usually underrepresented, poorly labeled, or even mislabeled, the exception being projects that aim at specifically describing such music. In this paper we present SAMBASET, a dataset of Brazilian *samba* music that contains over 40 hours of historical and modern *samba de enredo* commercial recordings. To the best of our knowledge, this is the first dataset of this genre. We describe the collection of metadata (e.g. artist, composer, release date) and outline our semiautomatic approach to the challenging task of annotating beats in this large dataset, which includes the assessment of the performance of state-of-the-art beat tracking algorithms for this specific case. Finally, we present a study on tempo and beat tracking that illustrates SAMBASET's value, and we comment on other tasks for which it could be used.

1. INTRODUCTION

Machine-learning-based systems in music information retrieval (MIR) are becoming increasingly complex to cope with the also expanding number of tasks and the challenges they propose. In turn, estimating the parameters of these large models requires more and better data [29], especially because such data must often be separated into training, test, and validation sets. Although data augmentation can be used to alleviate this bottleneck [29], this kind of strategy is not able to solve the cultural bias still present in existing MIR data, methodologies, and conclusions [38].

Indeed, a great part of the research in this field focuses on musical traditions usually labeled as "Western". This is worrying, since by doing so we risk not being able to fully evaluate and reproduce specific musical properties found in some cultures [38]. Some datasets attempt to be universal and to cover a large number of music styles, but end up sacrificing the very representation of what they are trying to portray. This is the case, for example, of the well-known Ballroom and Extended Ballroom datasets, whose "Samba" class contains a mixture of songs of different origins, of which only a few examples correspond to Brazilian rhythms, specifically identifiable as *bossa-nova*, *pagode*, and others [28]. In other datasets, music from non-"Western" traditions is given generic labels as "Latin", or "World" [28]. This underscores the importance of increasing the efforts towards the study of non-"Western" traditions found throughout the multicultural world we live in.

1.1 Other Culture-Specific Datasets

Here we review some of the existing datasets devoted to non-"Western" music traditions. One of the biggest projects today is CompMusic [38], which focuses on five particular music cultures: Arab-Andalusian, Beijing Opera, Turkish-makam, Hindustani, and Carnatic. Several annotations are provided, including melody (e.g. singer tonics, pitch contours), rhythm and structure (e.g. tala cycles), scores (e.g. for percussion patterns), and lyrics.

There are also some datasets of Latin-American music launched with MIR in mind. For instance, the dataset released in [32] comprises annotated audio recordings of Uruguayan *candombe* drumming, suited for beat/downbeat tracking. Aimed at music genre classification, the Latin Music Database [39] has Brazilian rhythms—*axé*, *forró*, *gaúcha*, *pagode*, and *sertaneja*—and music from other traditions: *bachata*, *bolero*, *merengue*, *salsa*, and *tango*. Closer to the topic of this article, two datasets focus exclusively on Brazilian music: intended for music genre classification, the Brazilian Music Dataset [40] includes *forró*, rock, *repente*, MPB (Brazilian popular music), *brega*, *sertanejo*, and disco; meant for beat/downbeat tracking and rhythmic pattern analysis, the BRID [28] consists of typi-

cal rhythmic patterns of *samba*, *samba de enredo*, *partido-alto* and other styles played on Brazilian instruments.

1.2 Our Contributions

In this paper, we present SAMBASET, the first large dataset of annotated *samba de enredo* recordings. Besides describing the dataset contents and detailing the beat and downbeat annotation process, we highlight one possible (musicological) use of this dataset through a study on recordings' tempo across the years, and briefly discuss how its results agree with expert knowledge about the evolution of *sambas de enredo* over the last few decades. Finally, we draw our concluding remarks and point out other challenges that can be tackled with SAMBASET.

1.3 Notes on *Samba*

Samba plays a special role in Brazil's image overseas. And every year, the country receives millions of tourists for Carnival activities in cities such as Salvador and Rio de Janeiro. Being Brazil's quintessential rhythm, *samba*'s development is closely related to that of Brazil itself.

Samba's roots can be traced back to dance and religious practices from the Afro-Brazilian diaspora [1, 20] and, as Araujo [1] points out, to the accommodation efforts made by people of African descent to maintain their heritage and cultural identity despite slavery and persecution. In many of these cultural practices, participants would form a *roda* (circle) and accompany one or more dancers (positioned at the center of the *roda*) by clapping, singing, and occasionally playing instruments [1, 37]. These traditions gave origin to different cultural manifestations, collectively associated with the term *samba*, [1] for example: *coco*, *samba de roda*, *partido-alto*, *samba de terreiro*, *pagode*, among others. In the post-Abolition period, *samba* overcame prohibition to become Brazil's national rhythm.

In the 1930s, the genre evolved to the rhythmic framework that still defines it today—generally characterized by duple meter (i.e., binary division of the periodically perceived pulsations) and strong syncopation. However, the idea of syncope—momentary contradiction of the prevailing meter or pulse [36]—can only be adequately applied to "Western" music, creating a fundamental problem in music traditions where this disruption of the pulse is the norm, and not the exception. That is why some authors prefer to resort to the concepts of commetricity and contrametricity [37], which indicate respectively when the surface rhythm confirms or contradicts the underlying meter, a terminology more commonly used in African music studies [2, 24, 37]. Therefore, it is more appropriate to say that *samba* presents a strong tendency towards contrametricity.

Later developments in *samba* led to, arguably, the most internationally famous of all its subgenres, the *samba de enredo*. These are *sambas* subject to an *enredo* (plot) composed in the context of an *escola de samba*—popular association for the practice of *samba*, strongly connected to a specific community—, and presented at parades in an organized competition annually held along a so-called Sambadrome during Carnival. At the core of every *escola de samba* lies the *bateria* (percussion ensemble). During a performance, the rhythmic aura of a *bateria* is created by the superposition of several cyclical individual parts, assigned to each multi-piece instrument set, similarly to what is observed in percussion ensemble practices throughout sub-Saharan Africa [1]. The *bateria* sets the mood of *samba*, but recent studies have observed an increase in the average tempo of *bateria* performances, an effect attributable to stricter parading time constraints [12, 22, 34].

[1] Possibly a variation of *semba*, word for a kind of circle dance practice in the Angolan Kimbundu language [37].

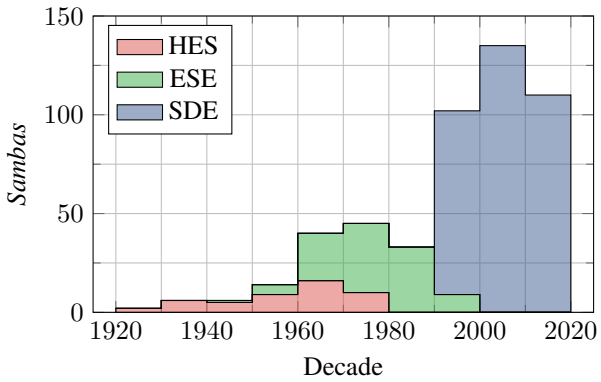


Figure 1: Recordings per decade of first performance.

2. DATASET OVERVIEW

Sambas de enredo are well documented in the phonographic industry. Apart from historical collections, since 1968 the yearly *sambas de enredo* that competing *escolas de samba* will perform at the carnival parade have been professionally recorded and marketed. Initially available as LP records, these official compilations began to appear as CDs in 1990. Since then, the amount of musicians (instrumentalists and choir) in each track has only increased.

Currently comprised of audio recordings, annotations and metadata, SAMBASET covers different eras, from later renditions of old classics to the most recent *sambas de enredo* just out of the Sambadrome. Figure 1 indicates the distribution of *sambas* w.r.t. the year they were first performed (typically, the parading year). Three major collections make up the dataset; in chronological order:

História das Escolas de Samba (HES): a collection of historical *sambas*, composed between 1928 and 1974, from four major *escolas de samba*, arranged and interpreted by the instrumentalists of each *escola*. Recorded in 1974, published in four LPs by Discos Marcus Pereira (redistributed as CDs in 2011), their 48 tracks also include a few *sambas de quadra/de terreiro* and *partidos-altos*.

Escolas de Samba – Enredos (ESE): a collection of historical *sambas*, composed between 1949 and 1993, from ten traditional *escolas de samba* in the voices of many idols from *samba*'s history, accompanied by a selected ensemble of instrumentalists and choir. There are a total of 100 tracks recorded and released in 1993 by Sony Music. This

	Genres			
Escola	SE	ST/SQ	OT	Total
Mangueira	45	3	1	49
Portela	41	5	2	48
Salgueiro	42	5	-	47
Império Serrano	31	5	-	36
Mocidade	35	-	1	36
Beija-Flor	34	1	-	35
Imperatriz	35	-	-	35
Vila Isabel	33	-	-	33
União da Ilha	27	-	-	27
Grande Rio	25	-	-	25
Unidos da Tijuca	24	-	-	24
Viradouro	18	-	-	18
Estácio	16	-	-	16
Porto Da Pedra	15	-	-	15
Caprichosos	12	-	-	12
São Clemente	12	-	-	12
Tradição	11	-	-	11
Other *escolas* (7)	14	-	-	14
Total	470	19	4	**493**

Table 1: Number of recordings in *SAMBASET* separated by *escolas* and by genres: *samba de enredo* (SE), *samba de terreiro/samba de quadra* (ST/SQ), and others (OT). [2]

```
<metadata dataset="SAMBASET"
          curator="John Doe"
          version="0.0.1">
...
  <album title="História das Escolas de Samba – Mangueira"
         arranger="Cartola"
         producer="J. C. Botezelli"
         instrumentalists="Various Artists"
         record_label="Discos Marcus Pereira"
         year_published="2011"
         length="00:29:48"
         total_tracks="12"
         album_code="HES1"
         barcode="7892141643634">
    ...
    <track track_number="6"
           title="Vale Do São Francisco"
           artist="Cartola"
           composer="Cartola and Carlos Cachaça"
           year_recorded="1974"
           year_first_performed="1948"
           genre="samba de enredo"
           length="02:49.226"
           samplerate="44100"
           bpm="78.4"
           start_time="00:07.895"
           end_time="02:49.226"
           checksum="d16974f135f0c374677c0e0db101cfea"/>
    ...
  </album>
...
</metadata>
```

Figure 2: Metadata file excerpt.

collection includes a couple of tracks from different subgenres (*samba de terreiro* and *samba-exaltação*).

Sambas de Enredo (SDE): official compilations of *sambas de enredo* recorded by members of the top *escolas* from Rio de Janeiro, for each carnival parade between 1994 and 2018. The 25 CDs gather 338 tracks, published by RCA/BMG/Sony BMG (1994–2006) and by Universal Music (after 2007), with one *samba de enredo* per track.

Table 1 gives the number of tracks for each *escola de samba* featured in the dataset, by genre. In total, there are 493 recorded sambas in 486 audio tracks, [3] resulting in over 40 hrs 30 min of content. All files are stereo with a sampling rate of 44.1 kHz and 16-bit resolution. Not only the three different collections allow for the coverage of different time periods, but they also have distinct sonorous characteristics. In HES, tracks feature only a few musicians playing very naturally and with great expression, as if they were in a *roda*. For several tracks in the official compilation (SDE), on the other hand, more than fifty instrumentalists play simultaneously while a choir of around the same size accompanies the main singer. Finally, ESE presents smaller ensembles and less expressiveness.

3. METADATA AND ANNOTATIONS

Metadata for albums and tracks were carefully curated and organized in an XML file. The information therein described was primarily obtained from CD booklets and later cross-checked with both the *União Brasileira de Compositores* [4] (lit. Brazilian Union of Composers, UBC) and the *Instituto Memória Musical Brasileira* [5] (Brazilian Musical Memory Institute, IMMuB). Whenever corresponding information was available, data was also checked against online database services such as FreeDB, MusicBrainz or Discogs. Finally, we consulted *samba*-oriented forums and websites for additional, conflicting or missing information.

All XML tags can be seen in Figure 2. While most of these labels are straightforward (e.g. title, composer), some require further clarification. First, the album_code refers to a unique code given to each album in the dataset. Albums from the HES and ESE collections were sequentially numbered, i.e., they are referred to by the codes HES1 to HES4 and ESE1 to ESE10, respectively. For SDE, albums were specified via the publishing year, which is also present in the album's title (i.e., SDE1994–SDE2018). The track_number is used with the album_code to name all audio files (e.g. the metadata in Figure 2 corresponds to file HES1.06.wav). Track's start_time and end_time indicate the time each *samba* starts and ends, respectively. This is invaluable since many *samba de enredo* recordings are preceded by a short introductory speech or song motivating the performance, or succeeded by a "farewell" shout after the music has already stopped. The checksum attributes were filled with the MD5 hash of the track's WAV file, to allow the verification of audio data integrity. Finally, mean bpm values were estimated from the beat annotations described in the following.

As of the writing of this paper, SAMBASET has annotations of beat and downbeat produced according to a semi-

[2] Some imbalance can be observed in the distributions of both genres and *escolas*. However, ST/SQ and OT tracks were only kept for the completeness of the dataset in regard to the CD collections, and the *escolas*' playing styles are not so heterogeneous as to make this imbalance critical.

[3] Some tracks in the ESE collection contain more than one *samba*.

[4] http://www.ubc.org.br/

[5] https://immub.org/

automatic procedure, after the results of the experiment described in Section 4. First, automatically-generated beat annotations were obtained for all audio files using the DBNBeatTracker system, which is available in the `madmom` package [3]—deemed a good candidate for providing reliable beat estimations (cf. Section 4). In a second step, these estimations were checked and manually corrected by one of the authors, who addressed eventual phase errors, and missing/extra beats. Since *samba de enredo* is always in duple meter, downbeats could be manually selected during this second phase. This two-step procedure greatly reduced the amount of manual work necessary to annotate beats and downbeats for this entire dataset.

4. ANALYSIS OF BEAT TRACKERS' PERFORMANCE

In this section, we provide a performance analysis of different state-of-the-art beat tracking systems, which were applied on a subset of short (30-second) excerpts from SAMBASET. Samples were selected according to a criterion based on the mean mutual agreement between beat estimation sequences generated by the algorithms under analysis, inspired by the approach of Holzapfel et al. [21]. This subset was manually annotated using Sonic Visualiser [11] by an expert with an engineering background, knowledgeable of audio technologies, and with many years of experience as a practicing musician, in particular of *samba*. Estimations were evaluated against this ground truth using three different types of metrics.

4.1 State-of-the-art Algorithms Considered

Fourteen algorithms (seen in Table 2) were used for sample selection and performance evaluation. We replaced eight of the algorithms originally featured in Holzapfel et al. [21] with six other algorithms released in following years, most notably those provided in the `madmom` package [3].

The algorithms are implemented in different programming languages and, in a few cases, require different operating systems. We used the Python implementations of AUB (version 0.4.9), ELL (provided by the librosa package [30] version 0.6.2), DEG and MFT (Essentia package [8] version 2.1-beta5-dev), BO1, BO2, and BO3 (madmom package [3] version 0.16.1); and the available releases of DIX (in Java, version 0.5.8), DAV (Vamp plugin in conjunction with the Sonic Annotator [10]), IB1 and IB2 (version 1.0 binaries). Finally, the C++ implementation of KLA was kindly provided by the author.

4.2 Evaluation Measures

Since there is currently no consensus on the evaluation metrics for the beat tracking task [13], with choices depending on the type of application, in this work we adopted the following methods [6]:

F-measure [17]: It is defined as the harmonic mean between precision (ratio between correctly detected and estimated beats) and recall (ratio between correctly detected and annotated beats). Generally, an estimated beat is considered correct if within ± 70 ms around an annotation.

Continuity-based measures [19]: Here, an estimated beat is considered correct if it is within a small tolerance around an annotation, the previous estimation has also been deemed correct, and the inter-beat interval is consistent with the inter-annotation interval within another tolerance—both generally set to 17.5% of the inter-annotation interval. CMLt ("correct metrical level") is the ratio between correct and annotated beats; accepting phase errors of half a beat period or octave errors in estimation yields the AMLt ("allowed metrical level").

Information Gain [14]: Defined as the Kullback-Leibler divergence between the observed beat error histogram (considering the timing errors of all estimated beats within a beat-length window around the annotations) and a uniform one (accounting for a pair of unrelated beat sequences), it spans the range $[0, \log_2(K)]$ bits, where K is the number of bins in the histograms (usually 40).

4.3 Selection of Ground Truth Excerpts

In [21], Holzapfel et al. presented a method for selecting challenging music examples for the beat tracking task without ground truth annotations. To do so, they first calculate the mean mutual agreement (MMA) between sequences estimated by a group of state-of-the-art beat trackers. The mutual agreement between two estimated beat sequences $\{i, j\}$ output by different beat tracking systems is given by the Information Gain in bits:

$$\mathrm{MA}_{i,j} = \mathrm{InfGain}(i, j), \quad i \neq j. \tag{1}$$

For a committee of N beat trackers, they then calculate the $N(N-1)/2$ different mutual agreements and average them all to obtain the MMA. The researchers show how a low mean mutual agreement coincides with perceptual and musical properties that make tapping difficult for humans. They build a challenging dataset by selecting samples with MMA < 1 bit, given a committee of five beat trackers.

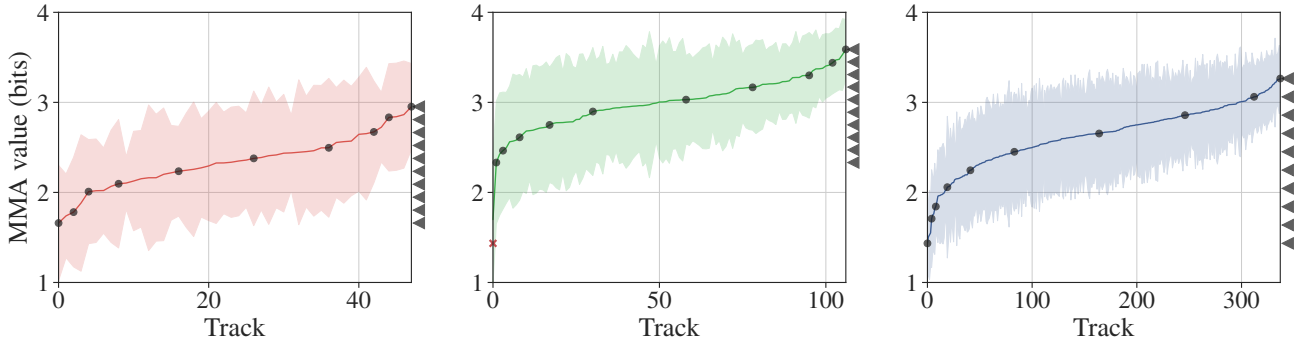
Later, in [42], they calculate the MaxMA, i.e., the algorithm whose output presents the maximum mutual agreement with the rest of the committee, showing that it provides the most reliable estimation for that given music example. They conduct subjective listening tests to determine a perceptual threshold for acceptable quality of this chosen output. This threshold is found to be 1.5 bits, for the same committee of five beat trackers. Their results also show a correlation between the test ratings and the MMA.

In this work, we followed a similar approach to select samples of various difficulties to the state-of-the-art algorithms. As mentioned in 4.1, we collected implementations of 14 beat tracking systems, removing some (the unavailable ones) featured in the original work [21], and adding others that were presented after its publication. We then extracted 30-second excerpts from all the different *sambas de enredo* in SAMBASET, and computed the MMA between estimations yielded by the beat trackers for all these 493 files. Figure 3 presents the ordered MMA values for excerpts from the three collections (HES, ESE, and SDE).

[6] Computed with standard settings using mir_eval [35] version 0.5.

Beat Tracker	CMLt (%)	AMLt (%)	F-meas. (%)	Inf. Gain (bits)
Aubio (AUB) [9]	59.4	65.6	61.9	2.30
BayesBeat-HMM (KR1) [25, 26]	42.7	65.6	67.6	2.27
BayesBeat-PF (KR2) [25, 27]	47.6	52.9	58.0	2.25
*BeatRoot (DIX) [17]	79.4	82.8	86.4	3.15
Davies (DAV) [15]	97.2	97.2	97.5	3.66
*Degara (DEG) [16]	88.3	91.2	89.7	3.40
*Ellis (ELL) [18]	76.9	76.9	78.7	3.35
IBT causal (IB1) [33]	83.4	83.4	86.2	2.45
*IBT non-causal (IB2) [33]	51.1	90.8	80.0	2.49
*Klapuri (KLA) [23]	61.3	63.7	63.1	3.09
BeatTracker (BO1) [5, 7]	98.1	98.1	98.6	3.78
DBNBeatTracker (BO2) [4, 26]	**99.5**	**99.5**	**99.5**	**3.80**
DBNDownBeatTracker (BO3) [6]	94.0	97.3	97.1	3.68
MultiFeature (MFT) [41]	86.4	86.4	86.8	3.55
Mean	76.1	82.2	82.2	3.09

Table 2: Ground truth performance of each beat tracking algorithm on the audio excerpts of SAMBASET. The best performance for each metric is highlighted in bold. The five-member committee proposed in [21] is indicated by an asterisk.



(a) HES collection (b) ESE collection (c) SDE collection

Figure 3: Collections sorted by MMA mean (solid line), with standard deviation (shaded region). Annotated samples (solid circles) were chosen as the closest to ten evenly spaced MMA values (solid triangles). One sample was treated as an outlier (cross) in (b).

For each collection, using the curve obtained from its excerpts, we determined P evenly spaced MMA values (including the maximum and minimum points), and selected the excerpts closest to each of these values to annotate. The reasons for this procedure are twofold: first, by selecting the same number of samples from each collection, we compensate for the large imbalance between them (e.g. in SDE there are nearly seven times more excerpts than in HES), while ensuring that their unique characteristics will be equally represented in the subset (recalling the variability in HES is much higher than that in ESE, or SDE); second, we guarantee that the algorithms will be compared within a group of samples to which they share different levels of consensus (and that would possibly provide a human annotator with a gamut of challenges). In total, thirty files were manually annotated ($P = 10$, i.e., ten from each collection), totalling just over 1 900 beats. It should be noted that a moderate number of annotated samples is sufficient, since we are dealing with a single music genre, which considerably limits the range of variations between them.

4.4 Discussion of Results

We see in Figure 3 that, in general, the fourteen beat tracking systems show more agreement in estimations for tracks in the ESE collection, followed by those in SDE, with HES in last. In fact, for over 50% of the tracks in ESE, the algorithms presented MMA > 3 bits, against slightly under 12% for SDE tracks and 0% in HES tracks in the same conditions. Considering an MMA > 2.5 bits, those percentages grow to 95%, 70% and 23%, respectively. This agrees with the authors' overall impression that the HES collection is the most "flavorful", whereas ESE is less expressive.

Regarding the test with the sampled subset, Table 2 gives the accuracy values for all algorithms, averaged over the thirty samples. Seven beat trackers perform better than the mean in all metrics, some of them outperforming the others by a large margin. In the end, the four best algorithms for our dataset are BO2, BO1, DAV, and BO3.

For the sake of comparison, we also evaluated the 493 excerpts with the five-member committee proposed in [21] and used in [42]: for 98.6% of the set the committee shows

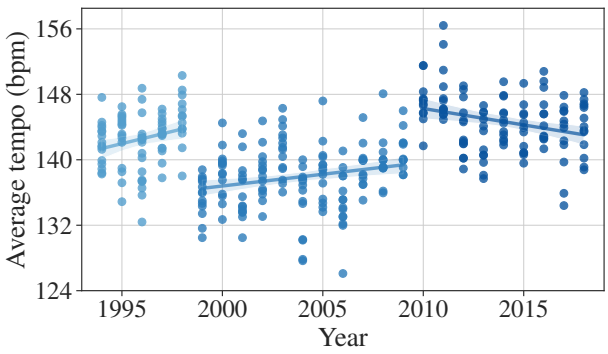


Figure 4: Variation of average tempo across the SDE collection with trend lines for three distinct regions and respective confidence intervals (shaded areas).

MMA > 1.5 bits; a single excerpt has MMA < 1 bit. This indicates that, overall, SAMBASET excerpts are not very challenging to the algorithms in this committee, which would provide a good number of acceptable estimations. Indeed, the good results shown by committee members in the ground truth performance suggests likewise. This analysis of state-of-the-art algorithms indicates a safe approach to semiautomatically annotating beats in this dataset.

5. MUSICOLOGICAL INSIGHTS

Here we investigate the evolution of average tempo in *samba de enredo* recordings across the years as represented in the SDE collection. For each excerpt, we compute the average tempo in beats per minute (bpm) as the inverse of the mean inter-beat interval, using the automatically detected beats. Figure 4 shows the average tempo for every track in SDE, plotted against the release year.

Although no clear trend is apparent from the whole data, we can readily verify the existence of local trends in three different regions of the graph. The first region accounts for the years of 1994 through 1998, and corresponds to the end of an era of "live" recordings in the *Teatro de Lona* (Barra da Tijuca), a large circus-like tent. As Moehn reveals on his essay "The Disc is not the Avenue" [31], by then the recordings were being made with a large number of musicians from each *escola* (around sixty) as well as large choirs from the respective community.

A radical change took place in the production of the 1999 disc: the entire process was moved to the studio and the number of *escola* members was reduced, not only to cut costs, but also to regain control over the sound organization [31]. Producers wanted the disc to sound "clear" and, thus, constrained the creative liberties of the *bateria*'s directors (e.g. they were not allowed to choose the tempo of the performance or to follow certain musical conventions that are common in a live performance). This was an attempt to recover the disc's marketability (sales had been dropping in previous years), despite distancing it from the actual phenomenon of the *samba de enredo* [31]. In 2010, "live" recordings were resumed, this time in the *Cidade do Samba* (Gambôa). Producers retreated in their interference on the soundscape creation, and the *escolas* were able to reclaim the final saying in some aspects of the recording, such as the tempo. The larger space provided by the *Cidade do Samba* also lead to an increase in the number of musicians taking part in the recordings: more than 8 000 for the 2014 CD against 1 500 in the 1998 recording [31].

Therefore, the first and third regions of Figure 4 more closely represent actual *samba de enredo* performances. In particular, notice that the average bpm in the third region is above the averages in the other two regions. This can be seen as a direct translation to the digital media of the decisions to accelerate the live performances (and the marching pace), so that the *escolas* satisfy changes in parading time limits, as reported by many specialists [12, 22, 34].

6. CHALLENGES

It would be very interesting to enrich this dataset with other types of annotations. In particular, one could think of generating ground truths for section boundaries (e.g. verses and the two different choruses that are very common in *samba de enredo* compositions), chord annotations (for instruments such as the *cavaquinho*), and instrument activity. As in the case of the CompMusic project [38], pitch contour annotations for soloist voices could be produced, as well as time-aligned lyrics and percussion transcription.

With these annotations, SAMBASET can provide many challenges to state-of-the-art algorithms in different MIR tasks. Tracks (specially in SDE) contain a plethora of simultaneous sounds of different qualities and textures, e.g. harmonic and percussive instruments, soloists and choirs. These could pose hard problems to vocal F0 or chord estimation systems. Also, singing voice annotation and lyrics could allow the study of soloist's interpretation as to phrasing, preferred ornaments, or characteristic syncopation; along with the metadata provided, it would be possible for example, to work on singer classification.

7. CONCLUSION

In this paper, we presented SAMBASET, a large *samba de enredo* dataset with rich metadata, beat and downbeat annotations. We provided a detailed overview of its contents [7] and reported a study on the performance of a group of state-of-the-art beat trackers over the set. We also motivated one musicological use of the dataset, i.e., the study of changes in *samba de enredo*'s rhythmic properties across several years. We expect that SAMBASET allows for technical improvements in traditional MIR tasks via new perspectives on problem solving that arise from contemplating cultures different from those to which we are accustomed.

8. ACKNOWLEDGEMENTS

Authors would like to thank CNPq and CAPES-ANII-CNRS (STIC AmSud program) for funding this work.

[7] Visit `http://www.smt.ufrj.br/~starel/sambaset/` for more information.

9. REFERENCES

[1] S. M. Araujo Junior. *Acoustic labor in the timing of everyday life: a critical contribution to the history of samba in Rio de Janeiro*. Phd thesis, University of Illinois, 1992.

[2] S. Arom. Structuration du temps dans les musiques d'afrique centrale: périodicité, mètre, rythmique et polyrythmie. *Revue de Musicologie*, 70(1):5–36, January 1984.

[3] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer. madmom: a new python audio and music signal processing library. In *24th ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, October 2016.

[4] S. Böck, F. Krebs, and G. Widmer. A multi-model approach to beat tracking considering heterogeneous music styles. In *15th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 603–608, Taipei, Taiwan, October 2014.

[5] S. Böck, F. Krebs, and G. Widmer. Accurate tempo estimation based on recurrent neural networks and resonating comb filters. In *16th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 625–631, Málaga, Spain, October 2015.

[6] S. Böck, F. Krebs, and G. Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *17th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 255–261, New York, USA, August 2016.

[7] S. Böck and M. Schedl. Enhanced beat tracking with context-aware neural networks. In *14th Int. Conf. on Digital Audio Effects (DAFx)*, pages 135–139, Paris, France, September 2011.

[8] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra. ESSENTIA: an audio analysis library for music information retrieval. In *14th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 493–498, Curitiba, Brazil, November 2013.

[9] P. M. Brossier. *Automatic Annotation of Musical Audio for Interactive Applications*. PhD thesis, Department of Electronic Engineering, Queen Mary University of London, 2006.

[10] C. Cannam, M. O. Jewell, C. Rhodes, M. Sandler, , and M. d'Inverno. Linked data and you: Bringing music research software into the semantic web. *Journal of New Music Research*, 39(4):313–325, 2010.

[11] C. Cannam, C. Landone, and M. Sandler. Sonic Visualiser: An open source application for viewing, analysing, and annotating music audio files. In *ACM Multimedia 2010 Int. Conf.*, pages 1467–1468, Florence, Italy, October 2010.

[12] F. L. Cunha. Samba locations: An analysis on the carioca samba, identities, and intangible heritage (Rio de Janeiro, Brazil). In F. L. Cunha, M. Santos, and J. Rabassa, editors, *Latin American Heritage*, The Latin America Studies Book Series, chapter 1, pages 3–20. Springer International Publishing, 2018.

[13] M. E. P. Davies, N. Degara, and M. D. Plumbley. Evaluation metrics for musical audio beat tracking algorithms. Technical Report C4DM-TR-09-06, Centre for Digital Music, Queen Mary University of London, London, UK, 2009.

[14] M. E. P. Davies, N. Degara, and M. D. Plumbley. Measuring the performance of beat tracking algorithms using a beat error histogram. *IEEE Signal Processing Letters*, 18(3):157–160, March 2011.

[15] M. E. P. Davies and M. D. Plumbley. Context-dependent beat tracking of musical audio. *IEEE Transactions on Audio, Speech and Language Processing*, 15(3):1000–1009, February 2007.

[16] N. Degara, E. A. Rúa, A. Pena, S. Torres-Guijarro, M. E. P. Davies, and M. D. Plumbley. Reliability-informed beat tracking of musical signals. *IEEE Transactions on Audio, Speech and Language Processing*, 20(1):290–301, January 2012.

[17] S. Dixon. Evaluation of the audio beat tracking system BeatRoot. *Journal of New Music Research*, 36(1):39–50, 2007.

[18] D. P. W. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[19] S. W. Hainsworth and M. D. Macleod. Particle filtering applied to musical tempo tracking. *EURASIP Journal on Advances in Signal Processing*, 15:2385–2395, November 2004.

[20] M. A. Hertzman. A Brazilian counterweight: Music, intellectual property and the African diaspora in Rio de Janeiro (1910s–1930s). *Journal of Latin American Studies*, 41(4):695–722, November 2009.

[21] A. Holzapfel, M. E. P. Davies, J. R. Zapata, J. L. Oliveira, and F. Gouyon. Selective sampling for beat tracking evaluation. *IEEE Transactions on Audio, Speech and Language Processing*, 20(9):2539–2548, November 2012.

[22] IPHAN. *Matrizes do Samba no Rio de Janeiro: partido-alto, samba de terreiro, samba-enredo*. National Institute of Historic and Artistic Heritage (IPHAN), Brasília, Brazil, 2014. IPHAN dossier 10.

[23] A. P. Klapuri, A. J. Eronen, and J. T. Astola. Analysis of the meter of acoustic musical signals. *IEEE Transactions on Audio, Speech and Language Processing*, 14(1):342–355, January 2006.

[24] M. Kolinski. A cross-cultural approach to metro-rhythmic patterns. *Ethnomusicology*, 17(3):494–506, September 1973.

[25] F. Krebs, S. Böck, and G. Widmer. Rhythmic pattern modelling for beat and downbeat tracking from musical audio. In *14th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 227–232, Curitiba, Brazil, November 2013.

[26] F. Krebs, S. Böck, and G. Widmer. An efficient state-space model for joint tempo and meter tracking. In *16th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 72–78, Málaga, Spain, October 2015.

[27] F. Krebs, A. Holzapfel, A. T. Cemgil, and G. Widmer. Inferring metrical structure in music using particle filters. *IEEE Transactions on Audio, Speech and Language Processing*, 23(5):817–827, May 2015.

[28] L. S. Maia, P. D. Tomaz Jr., M. Fuentes, M. Rocamora, L. W. P. Biscainho, M. V. M. Costa, and S. Cohen. A novel dataset of Brazilian rhythmic instruments and some experiments in computational rhythm analysis. In *2018 AES Latin American Congress of Audio Engineering*, pages 53–60, Montevideo, Uruguay, September 2018.

[29] B. McFee, E. J. Humphrey, and J. P. Bello. A software framework for musical data augmentation. In *16th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 248–254, Málaga, Spain, October 2015.

[30] Brian McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenbergk, and O. Nieto. librosa: Audio and music signal analysis in Python. In *14th Python in Science Conf. (SciPy)*, pages 18–24, Austin, USA, July 2015.

[31] F. J. Moehn. 'The disc is not the avenue': Schismogenetic mimesis in samba recording. In P. D. Green and T. Porcello, editors, *Wired for Sound: Engineering and Technologies in Sonic Cultures*, chapter 3, pages 47–83. Wesleyan University Press, 2005.

[32] L. Nunes, M. Rocamora, L. Jure, and L. W. P. Biscainho. Beat and downbeat tracking based on rhythmic patterns applied to the uruguayan candombe drumming. In *16th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 264–270, Málaga, Spain, October 2015.

[33] J. L. Oliveira, F. Gouyon, L. G. Martins, and L. P. Reis. IBT: A real-time tempo and beat tracking system. In *11th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 291–296, Utrecht, The Netherlands, August 2010.

[34] Y. Prado. Padrões musicais do samba-enredo na era do Sambódromo. *Música em Perspectiva*, 8(1):155–195, June 2015.

[35] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis. mir_eval: A transparent implementation of common MIR metrics. In *15th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 367–372, Taipei, Taiwan, October 2014.

[36] D. M. Randel. *The Harvard Dictionary of Music*. Belknap Press of Harvard University Press, Cambridge, USA, 4 edition, 2003.

[37] C. Sandroni. *Feitiço Decente: Transformações do samba no Rio de Janeiro (1917-1933)*. Jorge Zahar Ed./Ed. UFRJ, Rio de Janeiro, Brazil, 2 edition, 2008.

[38] X. Serra. A multicultural approach in music information research. In *12th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 151–156, Miami, USA, October 2011.

[39] C. N. Silla Jr., A. L. Koerich, and C. A. A. Kaestner. The Latin music database. In *9th Int. Conf. on Music Information Retrieval (ISMIR)*, pages 451–456, Philadelphia, USA, September 2008.

[40] J. M. Sousa, E. T. Pereira, and L. R. Veloso. A robust music genre classification approach for global and regional music datasets evaluation. In *2016 IEEE Int. Conf. on Digital Signal Processing*, pages 109–113, Beijing, China, October 2016.

[41] J. R. Zapata, M. E. P. Davies, and E. Gómez. Multi-feature beat tracking. *IEEE Transactions on Audio, Speech and Language Processing*, 22(4):816–825, April 2014.

[42] J. R. Zapata, A. Holzapfel, M. E. P. Davies, J. L. Oliveira, and F. Gouyon. Assigning a confidence threshold on automatic beat annotation in large datasets. In *13th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 157–162, Porto, Portugal, October 2012.

DEEP-RHYTHM FOR TEMPO ESTIMATION AND RHYTHM PATTERN RECOGNITION

Hadrien Foroughmand
IRCAM Lab - CNRS - Sorbonne Université
hadrien.foroughmand@ircam.fr

Geoffroy Peeters
LTCI - Télécom Paris - Institut Polytechnique de Paris
geoffroy.peeters@telecom-paris.fr

ABSTRACT

It has been shown that the harmonic series at the tempo frequency of the onset-strength-function of an audio signal accurately describes its rhythm pattern and can be used to perform tempo or rhythm pattern estimation. Recently, in the case of multi-pitch estimation, the depth of the input layer of a convolutional network has been used to represent the harmonic series of pitch candidates. We use a similar idea here to represent the harmonic series of tempo candidates. We propose the Harmonic-Constant-Q-Modulation which represents, using a 4D-tensors, the harmonic series of modulation frequencies (considered as tempo frequencies) in several acoustic frequency bands over time. This representation is used as input to a convolutional network which is trained to estimate tempo or rhythm pattern classes. Using a large number of datasets, we evaluate the performance of our approach and compare it with previous approaches. We show that it slightly increases Accuracy-1 for tempo estimation but not the average-mean-Recall for rhythm pattern recognition.

1. INTRODUCTION

Tempo is one of the most important perceptual elements of music. Today numerous applications rely on tempo information (recommendation, playlist generation, synchronization, dj-ing, audio or audio/video editing, beat-synchronous analysis). It is therefore crucial to develop algorithms to correctly estimate it. The automatic estimation of tempo from an audio signal has been one of the first research carried on in Music Information Retrieval (MIR) [11]. 25 years later it is still a very active research subject in MIR. This is due to the fact that tempo estimation is still an unsolved problem (outside the prototypical cases of pop or techno music) and that recent deep-learning approaches [3] [37] bring new perspectives to it.

Tempo is usually defined as (and annotated as) the rate at which people tap their foot or their hand when listening to a music piece. Several people can therefore perceive different tempi for the same piece of music. This is due to the hierarchy of the metrical structure in music (to deal with this ambiguity the research community has proposed to consider octave errors as correct) and due to the fact that without the cultural knowledge of the rhythm pattern(s) being played, it can be difficult to perceive "the" tempo (or even "a" tempo). This last point, of course, opens the door to data-driven approaches, which can learn the specificities of the patterns. In this work, we do not deal with the inherent ambiguity of tempo and consider the values provided by annotated datasets as ground-truth. The method we propose here belong to the data-driven systems in the sense that we learn from the data. It also considers both the tempo and rhythm pattern in interaction by adequately modeling the audio content. The tempo of a track can of course vary along time, but in this work we focus on the estimation of constant (global) tempi and rhythm patterns.

In the following section, we summarize works related to tempo and rhythm pattern estimation from audio. We refer the reader to [12, 32, 44] for more detailed overviews.

1.1 Related works

Tempo estimation. Early MIR systems encoded domain knowledge (audio, auditory perception and musical knowledge) by hand-crafting signal processing and statistical models (hidden Markov, dynamic Bayesian network). Data were at most used to manually tune some parameters (such as filter frequencies or transition probabilities). Early techniques for beat tracking and/or tempo estimation belong to this category. Their overall flowchart is a multi-band separation combined with an onset strength function which periodicity is measured. For example, Scheirer [36] proposes the use of band-pass filters combined with resonating comb filters and peak picking; Klapuri [18] uses the resonating comb filter bank which is driven by the bandwise accent signals, the main extension is the tracking of multiple metrical levels; Gainza and Cole [8] propose a hybrid multiband decomposition where the periodicities of onset functions are tracked in several frequency bands using autocorrelation and then weighted.

Since the pioneer works of [11], many audio datasets have been annotated into tempo. This therefore encourages researchers to develop data-driven systems based on machine learning and deep learning techniques. Such machine-learning models are K-Nearest-Neighbors (KNN) [40], Gaussian Mixture Model (GMM) [33, 43], Support Vector Machine (SVM) [4, 9, 34], bags of classifiers [20], Random Forest [38] or more recently deep learning models. The first use of deep learning for tempo estimation

was proposed by Böck et al. [3] who proposed a deep Recurrent Neural Network (bi-LSTM) to predict the position of the beats inside the signal. This output is then used as the input of a bank of resonating comb filters to detect the periodicity and so the tempo. This technique still achieves the best results today in terms of Accuracy2. Recently, Schreiber and Müller [37] proposed a "single step approach" for tempo estimation using deep convolutional networks. The network design is inspired by the flowchart of handcrafted systems: the first layer is supposed to mimic the extraction of an onset-strength-function. Their system uses as input Mel-spectrograms and the network is trained to classify the tempo of an audio excerpt into 256 tempo classes (from 30 to 285 BPM), it shows very good results in terms of Class-Accuracy and Accuracy1.

Rhythm pattern recognition. While tempo and rhythm pattern are closely interleaved, the recognition of rhythm pattern has received much less attention. This is probably due to the difficulty of creating datasets annotated in such rhythm pattern (defining the similarity between patterns — outside the trivial identity case — remains a difficult task). To create such a dataset, one may consider the equivalence between the rhythm pattern and the related dance (such as Tango): *Ballroom* [12], *Extended-ballroom* [24] and *Greek-dances* [14]. Systems to recognize rhythm pattern from the audio are all very different: Foote [7] defines a beat spectrum computed with a similarity matrix of MFCCs, Tzanetakis [42] defines a beat histogram computed from an autocorrelation function, Peeters [32] proposes a harmonic analysis of the rhythm pattern, Holzapfel [15] proposes the use of the scale transform (which allows to get a tempo invariant representation), Marchand [23, 25] extends the latter by combining it with the modulation spectrum and adding correlation coefficients between frequency bands. For a recognition task on the Extended-ballroom and Greek-dances, Marchand can be considered as the state-of-the-art.

1.2 Paper proposal and organization

In this paper we present a new audio representation, the Harmonic-Constant-Q-Modulation (HCQM), to be used as input to a convolutional network for global tempo and rhythm pattern estimation. We name it **Deep Rhythm** since it uses the **depth** of the input and a **deep** network. The paper is organized as follows. In §2, we describe the motivation for (§2.1) and the computation details of (§2.2) the HCQM. We then describe the architecture of the convolutional neural network we used (§2.3) and the training process (§2.4). In §3, we evaluate our proposal for the task of global tempo estimation (§3.1) and rhythm pattern recognition (§3.2) and discuss the results.

2. PROPOSED METHOD

2.1 Motivations

From Fourier series, it is known that any periodic signal $x(t)$ with period T_0 (of fundamental period $f_0 = 1/T_0$) can be represented as a weighted sum of sinusoidal components which frequencies are the harmonics of f_0: $\hat{x}_{f_0,\underline{a}}(t) = \sum_{h=1}^{H} a_h \sin(2\pi h f_0 t + \phi_h)$.

For the voiced part of speech or pitched musical instrument, this leads to the so-called "harmonic sinusoidal model" [26, 39] that can be used for audio coding or transformation. This model can also be used to estimate the pitch of a signal [21]: estimating the f_0 such that $\hat{x}_{f_0,\underline{a}}(t) \simeq x(t)$. The values a_h can be estimated by sampling the magnitude of the DFT at the corresponding frequencies $a_{h,f_0} = |X(hf_0)|$. The vector $\underline{a}_{f_0} = \{a_{1,f_0} \cdots a_{H,f_0}\}$ represents the spectral envelope of the signal and is closely related to the timbre of the audio signal, hence the instrument playing. For this reason, these values are often used for instrument classification [29].

For audio musical rhythm, Peeters [30] [31] [32] proposes to apply such an harmonic analysis to an onset-strength-function . The period T_0 is then defined as the duration of a beat. a_{1,f_0} then represents the DFT magnitude at the 4^{th}-note level, a_{2,f_0} at the 8^{th}-note level, a_{3,f_0} at the 8^{th}-note-triplet level, while $a_{\frac{1}{2},f_0}$ represent the binary grouping of the beats and $a_{\frac{1}{3},f_0}$ the ternary one. Peeters considers that the vector $\underline{a}$ is representative of the specific rhythm and that therefore $\underline{a}_{f_0}$ represents a specific rhythm played at a specific tempo f_0. He proposes the following harmonic series: $h \in \{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, 1, 1.25, 1.33, \ldots 8\}$ Using this, he shows - in [32] that given the tempo f_0, the vector $\underline{a}_{f_0}$ can be used to classify the different rhythm pattern; - in [30], that given manually-fixed prototype vectors $\underline{a}$, it is possible to estimate the tempo f_0 (looking for the f such that $\underline{a}_f \simeq \underline{a}$); - in [31] that the prototype vectors $\underline{a}$ can be learned (using simple machine-learning) to achieve the best tempo estimation f_0.

The method we propose in this paper is in the continuation of this last work (learning the values $\underline{a}$ to estimate the tempo or the rhythm class) but we adapted it to the deep learning formalism recently proposed by Bittner et al. [2] where the depth of the input to a convolutional network is used to represent the harmonic series $\underline{a}_f$. In [2], a constant-Q-transform (time τ and log-frequency f) is expanded to a third dimension which represent the harmonic series $\underline{a}_f$ of each f (with $h \in [\frac{1}{2}, 1, 2, 3, 4, 5]$). When $f = f_0$, $\underline{a}_f$ will represent the specific harmonic series of the musical instrument (plus an extra value at the $\frac{1}{2}f$ position used to avoid octave errors). When $f \neq f_0$, $\underline{a}_f$ will represent (almost) random values. In [2], the goal is to estimate the parameters of a filter such that when multiplied with this third dimension $\underline{a}_f$ it will provide very different values when $f = f_0$ or when $f \neq f_0$. This filter will then be convolved over all log-frequencies f and time τ to estimate the f_0's. This filter is trained using annotated data. In [2], there is actually several of such filter; they constitute the first layer of a convolutional network. In practice, in [2], the $a_{h,f}$ are not obtained as $|X(hf)|$; but by stacking in depth several CQTs each starting at different minimal frequencies $hf_{\min}$. The representation is denoted by Harmonic Constant-Q Transform (HCQT): $X_{hcqt}(f, \tau, h)$.

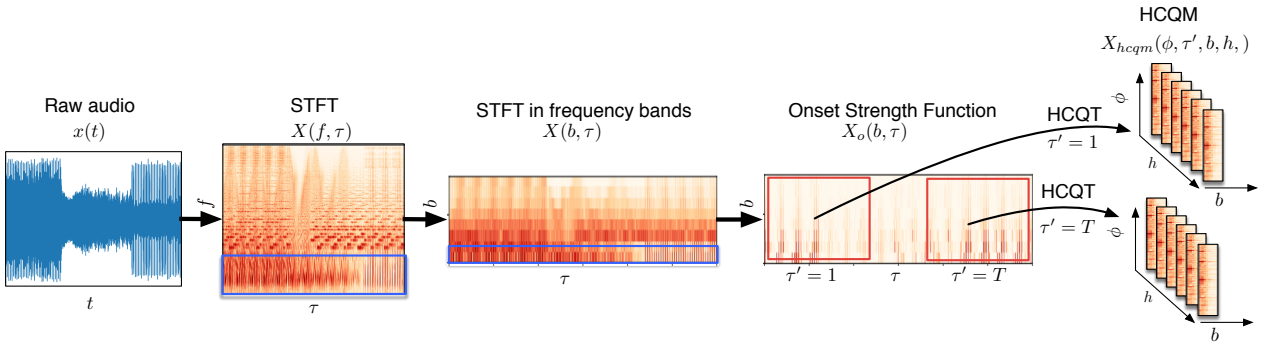


Figure 1. Flowchart of the computation of the Harmonic-Constant-Q-Modulation (HCQM). See text for details.

2.2 Input representation: the HCQM

As mentioned our goal is here to adapt the harmonic representation of the rhythm proposed in [30] [31] [32] to the deep learning formalism proposed in [2]. For this, the HCQT proposed by [2] is not applied to the audio signal, but to a set of Onset-Strength-Function (OSF) which represent the rhythm content in several acoustic frequency bands. The OSFs are low-pass signals which temporal evolution is related to the tempo and the rhythm pattern.

We denote our representation by **Harmonic-Constant-Q-Modulation (HCQM)**. As the Modulation Spectrum (MS) [1] it represents, using a time/frequency (τ'/ϕ) representation, the energy evolution (low-pass signal) within each frequency band b of a first time/frequency (τ/f) representation. However, while the MS uses two interleaved Short-Time-Fourier-Transforms (STFTs) for this, we use a Constant-Q transform for the second time/frequency representation (in order to obtain a better spectral resolution). Finally, as proposed by [2], we add one extra dimension to represent the content at the harmonics of each frequency ϕ. We denote it by $X_{hcqm}(\phi, \tau', b, h)$ where ϕ are the modulation frequencies (which correspond to the tempo frequencies), τ' are the times of the CQT frames, b are the acoustic frequency bands and h the harmonic numbers.

Computation. In Figure 1, we indicate the computation flowchart of the HCQM. Given an audio signal $x(t)$, we first compute its STFT, denoted by $X(f, \tau)$. The acoustic frequency f of the STFT are grouped into logarithmic-spaced acoustic-frequency-bands $b \in [1, B = 8]$. We denote it by $X(b, \tau)$. The goal of this is to reduce the dimensionality while preserving the information of the spectral location of the rhythm events (kick patterns tend to be in low frequencies while hit-hat patterns in high frequencies). For each band b, we then compute an Onset-Strength-Function over time τ, denoted by $X_o(b, \tau)$.

For a specific b, we now consider the signal $s_b(\tau) = X_o(b, \tau)$ and perform the analysis of its periodicities over time τ. One possibility would be to compute a time-frequency representation $S_b(\phi, \tau')$ over tempo-frequencies ϕ and time frame τ' and then sample $S_b(\phi, \tau')$ at the positions $h\phi$ with $h \in \{\frac{1}{2}, 1, 2, 3, 4, 5\}$ to obtain $S_b(\phi, \tau', h)$. This is the idea we used in [32]. However, in the present work, we use the idea proposed by [2]: we compute a set of CQTs [1], each one with a different starting frequency $h\phi_{\min}$. We set $\phi_{\min}$=32.7. Each of these CQTs gives us $S_{b,h}(\phi, \tau')$ for one value of h. Stacking them over h therefore provides us with $S_b(\phi, \tau', h)$. The idea proposed by [2] therefore allows to mimic the sampling at the $h\phi$ but provides the correct window length to achieve a correct spectral resolution. We finally stack the $S_b(\phi, \tau', h)$ over b to obtain the 4D-tensors $X_{hcqm}(\phi, \tau', b, h)$. The computation parameters (sample rate, hop size, window length and FFT size) are set such that the CQT modulation frequency ϕ represents the tempo value in BPM.

Illustration. For easiness of visualisation (it is difficult to visualize a 4D-tensor), we illustrate the HCQM $X_{hcqm}(\phi, \tau', b, h)$ for a given τ' (it is then a 3D-tensor). Figure 2 [Left] represent $X_{hcqm}(\phi, b, h)$ on a real audio signal with a tempo of 120 bpm. Each sub-figure represent $X_{hcqm}(\phi, b, h)$ for a different value of $h \in \{\frac{1}{2}, 1, 2, 3, 4, 5\}$. The y-axis and x-axis are the tempo frequency ϕ and the acoustic frequency band b. The dashed rectangle super-imposed to the sub-figures indicates the slice of values $X_{hcqm}(\phi = 120bpm, b, h)$ which corresponds to the ground-truth tempo. It is this specific pattern over b and h that we want to learn using the filters W of the first layer of our convolutional network.

2.3 Architecture of the Convolutional Neural Network

The architecture of our network is both inspired by the one of [2] (since we perform convolutions over an input spectral representation and use its depth) and the one of [37] (since we perform a classification). However, it differs in the definition of the input and output.

Input. In [2], the input is the 3D-tensor $X_{cqt}(f, \tau, h)$ and the convolution is done over f and τ (with filters of depth H). In our case, the input could be the 4D-tensors $X_{hcqm}(\phi, \tau', b, h)$ and the convolution could be done over ϕ, τ' and b (with filters of depth H). However, to simplify the computation, we reduce $X_{hcqm}(\phi, \tau', b, h)$ to a sequence over τ' of 3D-tensors $X_{hcqm}(\phi, b, h)$ [2]. The convolution is then done over ϕ and b with filters of depth H.

Our goal is to learn filters W narrow in ϕ and large in b which represents the specific shape of the harmonic con-

[1] For the STFT and CQT we used the librosa library [27].

[2] Future works will concentrate in performing the convolution directly using the 4D-tensors; which would allow to perform smoothing over time.

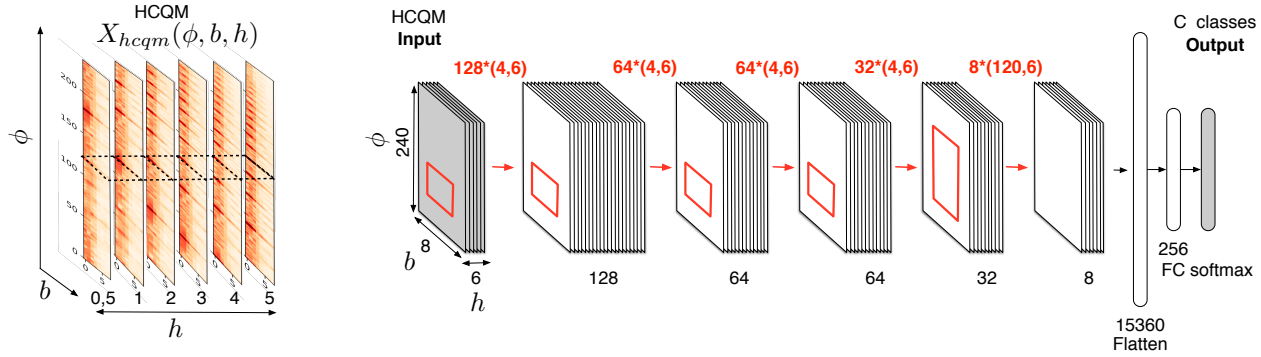


Figure 2. [Left] Example of the HCQM for a real audio with tempo 120bpm. [Right] Architecture of our CNN.

tent of a rhythm pattern. We do the convolution over b because the same rhythm pattern can be played with instrument transposed in acoustic frequencies.

Output. The output of the network proposed by [2] is a 2D representation which represents a saliency map of the harmonic content over time. In our case, the outputs are either the $C = 256$ classes of tempo (as in [37] we consider the tempo estimation problem as a classification problem into 256 tempo classes) or the $C = 13$ (for extended-ballroom) or $C = 6$ (for Greek-dances) classes of rhythm pattern. To do so, we added at the end of the network proposed by [2] two dense layers, the last one with C units and a softmax activation.

Architecture. In Figure 2 [Right], we indicate the architecture of our network. The input is a 3D-tensor $X_{hcqm}(\phi, b, h)$ for a given time τ'. The first layer is a set of 128 convolutional filters of shape $(\phi = 4, b = 6)$ (with depth H). As mentioned, the convolution is done over ϕ and b. The shape of these filters has been chosen such that they are narrow in tempo frequency ϕ (to precisely estimate the tempo) but cover multiple frequency acoustic bands b (because the information relative to the tempo/ rhythm cover several bands). As illustrated in Figure 2 [Left] the goal of the filters is to identify the pattern over b and h specific to $\phi =$ tempo frequency.

The first layer is followed by two convolutional layers of 64 filters of shape $(4, 6)$, one layer of 32 filters of shape $(4, 6)$, one layer of 8 filters of shape $(120, 6)$ (this allows to track down the relationships between the modulation frequencies ϕ). The output of the last convolution layer is then flattened and followed by a dropout with $p = 0.5$ (to avoid over-fitting [41]), a fully-connected layer of 256 units, a last fully-connected layer of C units with a softmax activation to perform the classification into C classes. The softmax activation vector is denoted by $\underline{y}(\tau')$. The Loss function to be minimized is a categorical cross entropy.

All layers are preceded by a batch normalization [16]. We used Rectified Linear Units (ReLU) [28] for all convolutional layers, and Exponential Linear Units (eLU) [5] for the first fully-connected layer.

2.4 Training

The inputs of our network are the 3D tensors HCQM $X_{hcqm}(\phi, b, h)$ computed for all time τ' of the music track. On the other side, the datasets we will use for our experiments only provide **global** tempi [3] or **global** rhythm classes as ground-truths. We therefore have several HCQMs for a given track which are all associated with the same ground-truth (multiple instance learning).

To fix the network hyper-parameters, we split the training set into a train (90%) and a validation part (10%). We used the ADAM [17] optimizer to find the parameters of the network with a constant learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1e - 8$. We used mini-batches of 256 HCQM with shuffle and a maximum of 20 epochs with early-stopping.

2.5 Aggregating decisions over time

Our network provides an estimation of the tempo or rhythm class at each time τ'. To obtain a **global** value we aggregate the softmax activation vectors $\underline{y}(\tau')$ over time by choosing the maximum of the vector $\underline{y}$ computed as the average over τ' of the $\underline{y}(\tau')$.

3. EVALUATION OF THE SYSTEM

We evaluate our proposed system for two tasks: - **global tempo estimation** (§3.1), - **rhythm pattern recognition** (§3.2). We used the same system (same input representation and same network architecture) for the two tasks. However, considering that the class definitions are different, we performed two independent trainings [4] .

3.1 Global tempo estimation

Training and testing sets. To be able to compare our results with previous works, we use the same paradigm (cross-dataset validation [5]) and the same datasets as [37] which we consider here as the state-of-the-art. The **training set** is the union of

[3] It should be noted that this does not always correspond to the reality of the track content since some of them have a tempo varying over time (tempo drift), have a silent introduction or a break in the middle.

[4] Future works, will consider training a single network for the two tasks or applying transfer learning of one task to the other.

[5] Cross-dataset validation uses separate datasets for training and testing; not only splitting a single dataset into a train and a test part.

- *LMD tempo*: it is a subset of the Lack MIDI dataset [35] annotated into tempo by [38]; it contains 3611 items of 30s excerpts of 10 different genres
- *MTG tempo* it is a subset of the GiantSteps MTG key dataset (MTG tempo) [6] annotated using a tapping method by [37]; it contains 1159 items of 2min excerpts of electronic dance music;
- *Extended Ballroom* [24]: it contains 3826 items of 13 genres (it should be noted that we removed from the Ballroom test-set the items existing in the Extended Ballroom training-set).

The total size of the training set is 8596. It covers multiple musical genres to favor generalization.

The **test-sets** are also the same as in [37] (see [38] for their details): - *ACM-Mirum* [33] (1410 items), - *ISMIR04* [12] (464 items), - *Ballroom* [12] (698 items), - *Hainsworth* [13] (222 items), - *GTzan-Rhythm* [22] (1000 items), - *SMC* [14] (217 items), - *Giantsteps Tempo* [19] (664 items). We also added the *RWC-popular* [10] (100 items) for comparison with [3]. As in [37], *Combined* denotes the union of all test-sets (except RWC popular).

Evaluation protocol. We measure the performances using the following metrics:

- **Class-Accuracy:** it measures the ability of our system to predict the correct tempo class (in our system we have 256 tempo classes ranging from 30 to 285bpm);
- **Accuracy1:** it measures if our estimated tempo is within $\pm 4\%$ of the ground-truth tempo
- **Accuracy2:** is the same as Accuracy1 but considering octave errors as correct

Results and discussions. The results are indicated in Tables 1, 2 and 3.

Validation of B and H. We first show that the separation of the signal into multiple acoustic-frequency-bands $b \in [1, B = 8]$ and the use of the harmonic depth H is beneficial. For this, we compare the results obtained using
- $B = 8$ and $h \in \{\frac{1}{2}, 1, 2, 3, 4, 5\}$ (column "new")
- $B = 8$ but $h \in \{1\}$ (column "h=1")
- $B = 1$ and $h \in \{\frac{1}{2}, 1, 2, 3, 4, 5\}$ (column "b=1").

Results are only indicated for the "Combined" test-sets. For all metrics (Class-Accuracy, Accuracy-1 and Accuracy-2), the best results are obtained using $B = 8$ and $h \in \{\frac{1}{2}, 1, 2, 3, 4, 5\}$.

Comparison with state-of-the-art. We now compare our results with the state-of-the-art represented by the 3 following systems: - **sch1** denotes the results published in [38], - **sch2** in [37] and - **böck** in [3].

According to these Tables, we see that our method allows an improvement for two test-sets: - *Ballroom* in terms of Class-Accuracy (73.8) - *Ballroom* and *Giantsteps* in terms of Accuracy1 (92.6 and 83.6) and Accuracy2 (98.7 and 97.9). This can be explained by the fact that the musical genres of these two test-sets are represented in our training set (by the *Extended-Ballroom* and *MTG tempo* respectively). It should be noted however that there is no intersection between the training and test sets. For the *ISMIR04* and *GTZAN* test-sets, our results are very close (but lower) to the one of **sch1**. For the *RWC popular* test sets, our results (73.0 and 98.0) largely outperforms the ones of **böck** in terms of Accuracy-1 and Accuracy-2. Finally, if we consider the *Combined* dataset, our method slightly outperforms the other ones in terms of Accuracy1 (74.4).

The worst results of our method were obtained for the *SMC* data-sets. This can be explained by the fact that the *SMC* data-sets contains rhythm patterns very different from the ones represented in our training-sets.

While our results for Accuracy2 (92.0) are of course higher than our results for Accuracy1 (74.4), they are still lower than the ones of **böck** (93.6). One reason for this is that our network (as the ones of sch1 and sch2) is trained to discriminate BPM classes, therefore it doesn't know anything about octave equivalences.

3.2 Rhythm pattern recognition

Training and testing sets. For the rhythm pattern recognition, it is not possible to perform cross-dataset validation (as we did above) because the definition of the rhythm pattern classes is specific to each dataset. Therefore, as in previous works [25], we perform a 10-fold cross-validation using the following datasets:

- *Extended Ballroom [24]:* it contains 4180 samples of C=13 rhythm classes: 'Foxtrot', 'Salsa', 'Viennesewaltz', 'Tango', 'Rumba', 'Wcswing', 'Quickstep', 'Chacha', 'Slowwaltz', 'Pasodoble', 'Jive', 'Samba', 'Waltz'
- *Greek-dances [15]:* it contains 180 samples of C=6 rhythm classes: 'Kalamatianos', 'Kontilies', 'Maleviziotis', 'Pentozalis', 'Sousta' and 'Kritikos Syrtos'

Evaluation protocol. As in previous works [25], we measure the performances using the average-mean-Recall R[6]. For a given fold f, the mean-Recall R_f is the mean over the classes c of the class-recall $R_{f,c}$. The average mean-Recall R is then the average over f of the mean-Recall R_f. We also indicate the standard deviation over f of the mean-Recall R_f.

Results and discussions. The results are indicated in Table 4. We compare our results with the ones of [25], denoted as **march**, considered here representative of the current state-of-the-art.

Our results (76.4 and 68.3) are largely below the ones of **march** (94.9 and 77.2). This can be explained by the fact that the "Scale and shift invariant time/frequency" representation proposed in [25] takes into account the interrelationships between the frequency bands of the rhythmic events while our HCQM does not.

To better understand these results, we indicate in Figure 3 [Top] the confusion matrices for the *Extended Ballroom*. The diagonal represents the Recall R_c of each class c. We see that our system is actually suitable to detect the majority of the classes: $R_c \geq 88\%$ for 9 classes over 13. ChaCha, Waltz and WcSwing make R completely drop. Waltz is actually estimated 97% of the

[6] The mean-Recall is not sensitive to the distribution of the classes. Its random value is always $1/C$ for a problem with C classes.

Datasets	sch1	sch2	böck	**new**	h=1	b=1
ACMMirum	38.3	**40.6**	29.4	38.2		
ISMIR04	**37.7**	34.1	27.2	29.4		
Ballroom	46.8	67.9	33.8	**73.8**		
Hainsworth	**43.7**	43.2	33.8	23.0		
GTzan	**38.8**	36.9	32.2	27.6		
SMC	14.3	12.4	**17.1**	7.8		
Giantsteps	53.5	**59.8**	37.2	25.5		
RWCpop	X	X	X	66.0		
Combined	40.9	**44.8**	31.2	36.8	29.8	32.4

Table 1. Class-Accuracy

sch1	sch2	böck	**new**	h=1	b=1
72.3	**79.5**	74.0	73.3		
63.4	60.6	55.0	61.2		
64.6	92.0	84.0	**92.6**		
65.8	77.0	**80.6**	73.4		
71.0	69.4	69.7	69.7		
31.8	33.6	**44.7**	30.9		
63.1	73.0	58.9	**83.6**		
X	X	60.0	**73.0**		
66.5	74.2	69.5	**74.4**	64.2	67.9

Table 2. Accuracy1

sch1	sch2	böck	**new**	h=1	b=1
97.3	97.4	**97.7**	96.5		
92.2	92.2	**95.0**	87.1		
97.0	98.4	**98.7**	**98.7**		
85.6	84.2	**89.2**	82.9		
93.3	92.6	**95.0**	89.1		
55.3	50.2	**67.3**	50.7		
88.7	89.3	86.4	**97.9**		
X	X	95.0	**98.0**		
92.2	92.1	**93.6**	92.0	82.0	88.4

Table 3. Accuracy2

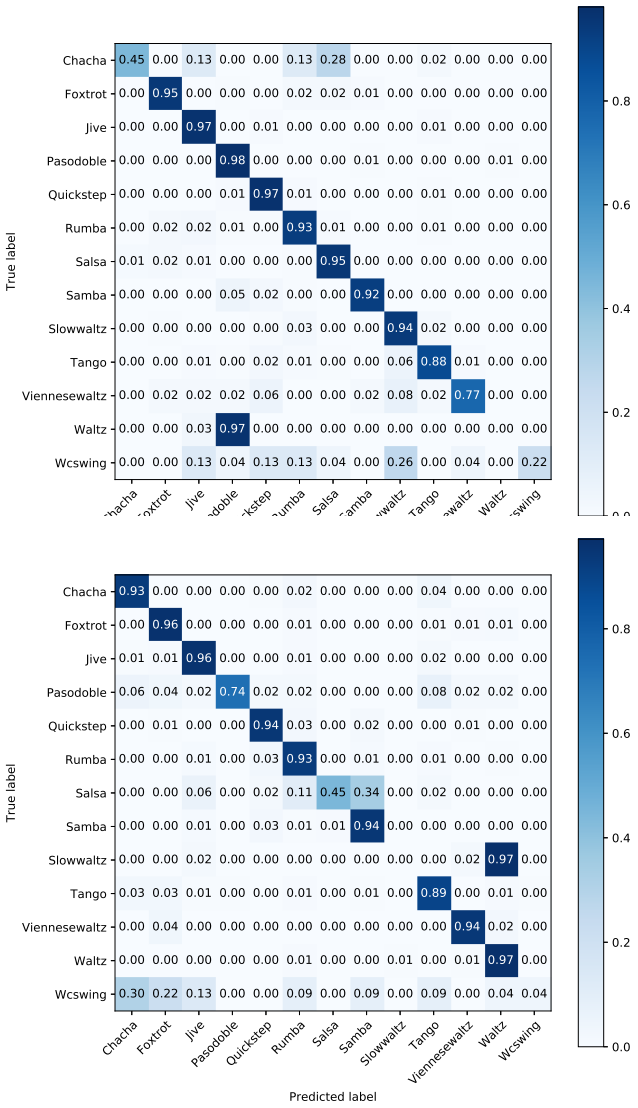


Figure 3. Confusion matrix for the *Extended Ballroom*. [Top] using $h \in \{0.5, 1, 2, 3, 4, 5\}$ [Bottom] using $h \in \{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, 1, 1.25, 1.33, \dots 8\}$

time as Pasadoble. This can be explained by the fact that our current harmonic series $h \in \{\frac{1}{2}, 1, 2, 3, 4, 5\}$ does not represent anything about the 3/4 meter specific to Waltz which would be represented by $h = \frac{1}{3}$. To verify our assumption, we redo the experiment using this time exactly the same harmonic series as proposed in [32]: $h \in \{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, 1, 1.25, 1.33, \dots 8\}$. The corresponding confusion matrix is indicated in 3 [Bottom] where Waltz is now perfectly recognized (97%), however SlowWaltz is now recognized as Waltz in 97% of the cases which makes (while the system is better) the average mean-Recall actually decreases to 74.6%. The low results of Wcswing can be explained by the (too) small number of training examples (23).

Datasets	march	**new**
Extended Ballroom	94.9	76.4 (33.1)
Greek-dances	77.2	68.3 (27.5)

Table 4. Average (std) Recall R for rhythm classification.

4. CONCLUSION

In this paper we have presented a new approach for global tempo and rhythm pattern classification. We have proposed the Harmonic-Constant-Q-Modulation (HCQM) representation, a 4D-tensor which represents the harmonic series at candidate tempo frequencies of a multi-band Onset-Strength-Function. This HCQM is then used as input to a deep convolutional network. The filters of the first layer of this network are supposed to learn the specific characteristic of the various rhythm patterns. We have evaluated our approach for two classification tasks: global tempo and rhythm pattern classification.

For tempo estimation, our method outperforms previous approaches (in terms of Accuracy1 and Accuracy2) for the *Ballroom* (ballroom music) and *Giant-steps tempo* (electronic music) test-sets. Both test-sets represent music genres with a strong focus on rhythm. It seems therefore that our approach works better when rhythm patterns are clearly defined. Our method also performs slightly better (in terms of Accuracy1) for the Combined test-set.

For rhythm classification, our method doesn't work as well as the state of the art [25]. However, the confusion matrices indicate that our recognition is above 90% for the majority of the classes of the *Extended Ballroom*. Moreover, we have shown that adapting the harmonic series h can help improving the performances.

Among future works, we would like to study the use of the HCQM 4D tensors directly, to study other harmonic series and to study the joint training of (or transfer learning between) tempo and rhythm pattern classification.

Acknowledgement: This work was partly supported by European Union's Horizon 2020 research and innovation program under grant agreement No 761634 (Future Pulse project).

5. REFERENCES

[1] Les Atlas and Shihab A Shamma. Joint acoustic and modulation frequency. *Advances in Signal Processing, EURASIP Journal on*, 2003:668–675, 2003.

[2] Rachel M Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan Pablo Bello. Deep salience representations for f0 estimation in polyphonic music. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Suzhou, China, 2017.

[3] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Accurate tempo estimation based on recurrent neural networks and resonating comb filters. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Malaga, Spain, 2015.

[4] Ching-Wei Chen, Markus Cremer, Kyogu Lee, Peter DiMaria, and Ho-Hsiang Wu. Improving perceived tempo estimation by statistical modeling of higher-level musical descriptors. In *Audio Engineering Society Convention 126*. Audio Engineering Society, 2009.

[5] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[6] Angel Faraldo, Sergi Jorda, and Perfecto Herrera. A multi-profile method for key estimation in edm. In *AES International Conference on Semantic Audio*, Ilmenau, Germany, 2017. Audio Engineering Society.

[7] Jonathan Foote, Matthew L Cooper, and Unjung Nam. Audio retrieval by rhythmic similarity. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Paris, France, 2002.

[8] Mikel Gainza and Eugene Coyle. Tempo detection using a hybrid multiband approach. *Audio, Speech and Language Processing, IEEE Transactions on*, 19(1):57–68, 2011.

[9] Aggelos Gkiokas, Vassilios Katsouros, and George Carayannis. Reducing tempo octave errors by periodicity vector coding and svm learning. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Porto, Portugal, 2012.

[10] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. Rwc music database: Popular, classical and jazz music databases. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Paris, France, 2002.

[11] Masataka Goto and Yoichi Muraoka. A beat tracking system for acoustic signals of music. In *Proceedings of the second ACM international conference on Multimedia*, San Francisco, California, USA, 1994.

[12] Fabien Gouyon, Anssi Klapuri, Simon Dixon, Miguel Alonso, George Tzanetakis, Christian Uhle, and Pedro Cano. An experimental comparison of audio tempo induction algorithms. *Audio, Speech and Language Processing, IEEE Transactions on*, 14(5):1832–1844, 2006.

[13] Stephen Webley Hainsworth. *Techniques for the Automated Analysis of Musical Audio*. PhD thesis, University of Cambridge, UK, September 2004.

[14] Andre Holzapfel, Matthew EP Davies, José R Zapata, João Lobato Oliveira, and Fabien Gouyon. Selective sampling for beat tracking evaluation. *Audio, Speech and Language Processing, IEEE Transactions on*, 20(9):2539–2548, 2012.

[15] André Holzapfel and Yannis Stylianou. Scale transform in rhythmic similarity of music. *Audio, Speech and Language Processing, IEEE Transactions on*, 19(1):176–185, 2011.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] Anssi P Klapuri, Antti J Eronen, and Jaakko T Astola. Analysis of the meter of acoustic musical signals. *Audio, Speech and Language Processing, IEEE Transactions on*, 14(1):342–355, 2006.

[19] Peter Knees, Angel Faraldo, Perfecto Herrera, Richard Vogl, Sebastian Böck, Florian Hörschläger, and Mickael Le Goff. Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Malaga, Spain, 2015.

[20] Mark Levy. Improving perceptual tempo estimation with crowd-sourced annotations. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Miami, Florida, USA, 2011.

[21] Robert C Maher and James W Beauchamp. Fundamental frequency estimation of musical signals using a two-way mismatch procedure. *JASA (Journal of the Acoustical Society of America)*, 95(4):2254–2263, 1994.

[22] Ugo Marchand, Quentin Fresnel, and Geoffroy Peeters. Gtzan-rhythm: Extending the gtzan test-set with

beat, downbeat and swing annotations. In *Late-Breaking/Demo Session of ISMIR (International Society for Music Information Retrieval)*, Malaga, Spain, October 2015. Late-Breaking Demo Session of the 16th International Society for Music Information Retrieval Conference, 2015.

[23] Ugo Marchand and Geoffroy Peeters. The modulation scale spectrum and its application to rhythm-content description. In *Proc. of DAFx (International Conference on Digital Audio Effects)*, Erlangen, Germany, 2014.

[24] Ugo Marchand and Geoffroy Peeters. The extended ballroom dataset. In *Late-Breaking/Demo Session of ISMIR (International Society for Music Information Retrieval)*, New York, USA, August 2016. Late-Breaking Demo Session of the 17th International Society for Music Information Retrieval Conf.. 2016.

[25] Ugo Marchand and Geoffroy Peeters. Scale and shift invariant time/frequency representation using auditory statistics: Application to rhythm description. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2016.

[26] R McAuley and T Quatieri. Speech analysis/synthesis based on a sinusoidal representation. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 34:744–754, 1986.

[27] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, Austin, Texas, 2015.

[28] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. of ICMC (International Computer Music Conference)*, Haifa, Israel, 2010.

[29] Geoffroy Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. Cuidado project report, Ircam, 2004.

[30] Geoffroy Peeters. Template-based estimation of time-varying tempo. *Advances in Signal Processing, EURASIP Journal on*, 2007(1):067215, 2006.

[31] Geoffroy Peeters. Template-based estimation of tempo: using unsupervised or supervised learning to create better spectral templates. In *Proc. of DAFx (International Conference on Digital Audio Effects)*, pages 209–212, Graz, Austria, September 2010.

[32] Geoffroy Peeters. Spectral and temporal periodicity representations of rhythm for the automatic classification of music audio signal. *Audio, Speech and Language Processing, IEEE Transactions on*, 19(5):1242–1252, 2011.

[33] Geoffroy Peeters and Joachim Flocon-Cholet. Perceptual tempo estimation using gmm-regression. In *Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies*, Nara, Japan, 2012.

[34] Graham Percival and George Tzanetakis. Streamlined tempo estimation based on autocorrelation and cross-correlation with pulses. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 22(12):1765–1776, 2014.

[35] Colin Raffel. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. PhD thesis, Columbia University, 2016.

[36] Eric D Scheirer. Tempo and beat analysis of acoustic musical signals. *JASA (Journal of the Acoustical Society of America)*, 103(1):588–601, 1998.

[37] Hendrik Schreiber and M Müller. A single-step approach to musical tempo estimation using a convolutional neural network. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Paris, France, 2018.

[38] Hendrik Schreiber and Meinard Müller. A post-processing procedure for improving music tempo estimates using supervised learning. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Suzhou, China, 2017.

[39] Xavier Serra and Julius Smith. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):12–24, 1990.

[40] Klaus Seyerlehner, Gerhard Widmer, and Dominik Schnitzer. From rhythm patterns to perceived tempo. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Vienna, Austria, 2007.

[41] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[42] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *Speech and Audio Processing, IEEE Transactions on*, 10(5):293–302, 2002.

[43] Linxing Xiao, Aibo Tian, Wen Li, and Jie Zhou. Using statistic model to capture the association between timbre and perceived tempo. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Philadelphia, PA, USA, 2008.

[44] Jose Zapata and Emilia Gómez. Comparative evaluation and combination of audio tempo estimation approaches. In *Aes 42Nd Conference On Semantic Audio*, Ilmenau, Germany, 2011.

LARGE-VOCABULARY CHORD TRANSCRIPTION VIA CHORD STRUCTURE DECOMPOSITION

Junyan Jiang[1,2,3] **Ke Chen**[1,2] **Wei Li**[1] **Gus Xia**[2]

[1] Computer Science Department, Fudan University [2] Music X Lab, NYU Shanghai
[3] Machine Learning Department, Carnegie Mellon University

[1]{jiangjy14, kchen15, weili-fudan}@fudan.edu.cn, [2]gxia@nyu.edu

ABSTRACT

While audio chord recognition systems have acquired considerable accuracy on small vocabularies (e.g., major/minor chords), the large-vocabulary chord recognition problem still remains unsolved. This problem hinders the practical usages of audio recognition systems. The difficulty mainly lies in the intrinsic long-tail distribution of chord qualities, and most chord qualities have too few samples for model training.

In this paper, we propose a new model for audio chord recognition under a huge chord vocabulary. The core concept is to decompose any chord label into a set of musically meaningful components (e.g., triad, bass, seventh), each with a much smaller vocabulary compared to the size of the overall chord vocabulary. A multitask classifier is then trained to recognize all the components given the audio feature, and then labels of individual components are reassembled to form the final chord label. Experiments show that the proposed system not only achieves state-of-the-art results on traditional evaluation metrics but also performs well on a large vocabulary.

1. INTRODUCTION

Large-vocabulary chord transcription is a difficult task, as the number of chord qualities is large, and the distribution of training chord classes is extremely biased. For example, the Billboard dataset [2], a human-annotated dataset, contains 230 different chord qualities, or equivalently, 2,749 distinct chord classes [1]. While the first 10% chord qualities cover 93.86% of the data, the last 50% chord qualities only cover 0.35% of the data altogether [2]. Such a long-tailed chord distribution makes it extremely hard to model rare chord qualities.

To bypass the problem, former systems typically adopt two kinds of strategies: chord quality simplification and chord structure representation. The first approach maps complex chord qualities into simpler ones (e.g., to map `C:maj7`, `C:maj9`, `C:maj11` etc. all to `C:maj`), and therefore suppress the number of different chord classes. Then, a classifier is trained based on the simplified chord vocabulary. The most common mapped chord vocabulary is the major/minor vocabulary, which consists of merely 12 major chords, 12 minor chords, and a non-chord label. However, this is often an over-simplification, especially for some pop and jazz chords with richer expressions than major/minor chords do.

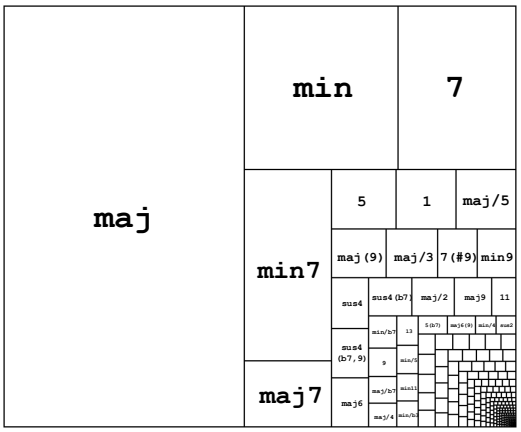


Figure 1. A visualization of chord quality distribution in our collected chord dataset. Each chord quality is denoted by a block whose size is proportional to its number of appearances. Labels for small blocks are omitted.

The second approach, instead, turns chord label classification into a structured estimation problem, and this study belongs to this category. To achieve this, a unified structure representation of chord symbols is required. Chromagram-like representations are often adopted for chord recognition as it is a direct reflection of acoustic features and often led to a good performance in practice [16, 20].

However, Chromagram-like representations miss some of the musical semantics that are important references for human transcribers. One example features two chords `E:min7/b3` and `G:maj6` which are very similar in the chromagram as they share the same chord notes {G, B, D, E} and bass G. However, they are quite different in musical

[1] We here assume that each chord quality can be combined with all possible 12 roots except for the `N` chord.

[2] In calculation, the chord quality counts are weighted by their durations.

semantics as one is a major chord and the other is minor. Moreover, in audio chord recognition, some chord degrees are more important than others. For example, a wrong triad is regarded as a more severe problem than a wrong ninth or other extensions. Such intuition is not directly reflected in a chromagram-like representation of a chord.

We believe that structured representation is the key to large-vocabulary chord recognition, yet a more musically meaningful representation is needed. In this paper, we propose a new representation method by chord structure decomposition. We also present a multitask classification model comprised of a Convolutional Recurrent Neural Network (CRNN) and a Conditional Random Field (CRF), that utilize such representation into the audio chord recognition system.

2. RELATED WORK

Audio chord recognition is an important task for content-based music information retrieval and has been actively studied for 20 years [22]. Audio chord recognition systems typically follow a two-stage process: feature extraction and chord sequence decoding. For traditional models, chromagram [19, 22, 23] is the most widely used feature for chord recognition as it reflects the acoustic properties of a chord. There are also other attempts for handcrafted features, such as the tonnetz feature [12]. For the sequence decoding model, template matching and Hidden Markov Models (HMMs) [24, 27] are adopted to decode the most likely chord sequence. Considering the relationship of chord labels and other musical concepts (e.g., beats, keys), complex HMMs and other Dynamic Bayesian Networks (DBNs) are also used for joint decoding [19, 23].

With the development of deep learning, recent studies begin to focus more on features extracted by neural networks, such as feed-forward Deep Neural Networks (DNNs) [16] and Convolutional Neural Networks (CNNs) [10, 17]. For the sequence decoding phase, beam search and the Viterbi algorithm are widely used [1,18,28]. While early papers in this area adopt a small and fixed vocabulary like the major/minor vocabulary [1,16,17], recent work focuses more and more on larger vocabularies [7, 8, 20].

Strategies for structured representation of chords has been widely discussed in the field of music generation and analysis [4, 9], but not all of them are helpful in the context of chord recognition problem. In audio chord recognition, the HPA system [23] uses two latent variables, the root-position form and the bass note, to model a chord in a Hidden Markov Model. McFee and Bello [20] adopt a 36-D vector to represent a chord, denoting a binary encoding of its root, bass and chord degrees respectively. Then, a plain 170-class classifier is adopted to decode chord symbols from the features.

3. PROPOSED METHOD

3.1 Chord Structure Representation

Most chord symbols can be regarded as a collection of musically meaningful components: root, bass (for possible inversions), a triad type, and a set of extra notes, which usually include certain 7^{th}, 9^{th}, 11^{th} and/or 13^{th} degrees above the root. Different combinations of these components form the large chord vocabulary that musicians use. However, the possible choices for each component are pretty limited. For example, the 7^{th} degree of a chord, if exists, is most likely to be one among `7`, `b7` and `bb7` (in `dim7` chords). This property is very helpful under the context of audio chord recognition, as the number of classes for chord label classification can be greatly reduced if we break down the chords into these components and perform classification on each component instead.

Chord	Root	Triad	Bass	7th	9th	11th	13th
G:maj	G	maj	G	N	N	N	N
G:maj7	G	maj	G	7	N	N	N
G:7(b9)	G	maj	G	b7	b9	N	N
G:min7/b3	G	min	Bb	b7	N	N	N
B:hdim7	B	dim	B	b7	N	N	N
A:sus4(b7)	A	sus4	A	b7	N	N	N
C:9(13)	C	maj	C	b7	9	N	13
N	N	N	N	N	N	N	N

Table 1. Some examples of chord structure decomposition.

We now formally introduce our chord representation method. We first define the chord components and their possible labels:

$$\begin{aligned}
\text{Root} &\leftarrow \{\text{N}, \text{C}, \text{C\#/Db}, \text{D}, ..., \text{B}\} \\
\text{Triad} &\leftarrow \{\text{N}, \text{maj}, \text{min}, \text{sus4}, \text{sus2}, \text{dim}, \text{aug}\} \\
\text{Bass} &\leftarrow \{\text{N}, \text{C}, \text{C\#/Db}, \text{D}, ..., \text{B}\} \\
\text{Seventh} &\leftarrow \{\text{N}, 7, \text{b7}, \text{bb7}\} \\
\text{Ninth} &\leftarrow \{\text{N}, 9, \#9, \text{b9}\} \\
\text{Eleventh} &\leftarrow \{\text{N}, 11, \#11\} \\
\text{Thirteenth} &\leftarrow \{\text{N}, 13, \text{b13}\}
\end{aligned}$$

Notice that we ignore incomplete triads like `C:(1,5)` and `C:(1)`, so these qualities will not appear in the triad category. To make other chord degrees (e.g., the major 6^{th} in `maj6` and `min6`) compatible with the representation, we here adopt octave equivalent versions of these degrees that match the form of the chord extensions listed above (e.g., 6^{th} to 13^{th}).

By enumerating distinct labels for these components, we can map the component labels into numeral values. The enumerated values are all indexed from 1. We now define the chord vector, the encoded form of chord used by our models. The chord vector $\mathbf{c} = (c_1, ..., c_6)$ of a chord C is defined as a 6-dimension vector. For non-chord class `N`, all $c_i = 1$. For any other chord label C, we have:

$$\begin{aligned}
c_1 &= (\text{TriadIndex}(C) - 1) \cdot 12 + \text{RootIndex}(C) + 1 \\
c_2 &= \text{BassIndex}(C) \\
c_3 &= \text{SeventhIndex}(C) \\
c_4 &= \text{NinthIndex}(C) \\
c_5 &= \text{EleventhIndex}(C) \\
c_6 &= \text{ThirteenthIndex}(C)
\end{aligned}$$

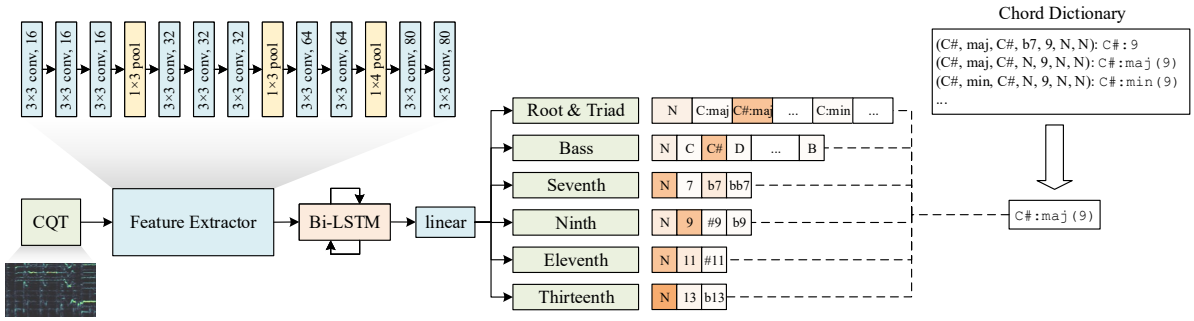


Figure 2. An overview of the system.

3.2 Model Architecture

The proposed chord recognition system consists of two parts, the feature processor and the decoding model. The feature processor transforms the low-level audio features to the frame-wise activations for each chord component value. The decoding model then decodes the final chord sequence given the activations. An overview of the system is shown in Figure 2.

3.2.1 Feature Processor

We first apply the Convolutional Recurrent Neural Network (CRNN), a powerful architecture for audio feature processing [3, 6, 20], to the input spectrogram. The audio feature is first fed into convolutional layers. After each convolutional layer, we perform batch normalization and then a rectified linear function to introduce non-linearity. Each convolutional layer adopts a 3×3 kernel size with 1×1 zero padding to the input, except for the last two layers where no padding is applied on the frequency axis, in order to reduce the feature size.

After that, we apply a Bi-directional Long Short-Term Memory (Bi-LSTM) layer to introduce temporal context for chord recognition. Each direction has a hidden size of 96. Then, for each frame t, a linear unit transforms the hidden states for both directions into six vectors $\mathbf{s}^{(t,1)}...\mathbf{s}^{(t,6)}$. The softmax function is performed on each vector to get the activation $\mathbf{a}^{(t,1)}...\mathbf{a}^{(t,6)}$ for each component of the chord vector:

$$a_k^{(t,i)} = \frac{\exp(s_k^{(t,i)})}{\sum_{k'=1}^{N_i} \exp(s_{k'}^{(t,i)})} \; \forall i = 1...6, \forall k = 1...N_i \quad (1)$$

Here, N_i denotes the vocabulary size of the i-th component category. The loss of the neural network given the ground-truth chord sequence $\mathbf{C} = \{C^{(1)}, ..., C^{(T)}\}$ is defined as weighted cross-entropy loss:

$$L = -\sum_{t=1}^{T}\sum_{i=1}^{6}\sum_{k=1}^{N_i} w_k^{(i)} \mathbb{I}[k = c_i^{(t)}] \log(a_k^{(t,i)}) \quad (2)$$

Here, $\mathbb{I}[\cdot]$ is the indicator function, and $c_i^{(t)}$ denotes the i-th component of the chord vector of $C^{(t)}$. $w_k^{(i)}$ is the class weight factor for index k of the i-th component. We will further explain it in section 3.3.

3.2.2 Chord Decoding Model

To decode the final chord sequence from the activation vectors $\mathbf{a}$, an intuitive way is to pick the class with the largest activation values for each component and each frame. However, this approach tends to produce excessive chord changes as there is no transition penalty between two frames. Also, it will be hard if we want to control the output chord vocabulary V. Therefore, we propose a decoding model by a linear Conditional Random Field (CRF).

The linear CRF takes the following form:

$$P(\mathbf{C} \mid \mathbf{F}) \propto \phi(C^{(1)}, \mathbf{F}) \prod_{t=2}^{T} \phi(C^{(t)}, \mathbf{F}) \psi(C^{(t-1)}, C^{(t)}) \quad (3)$$

Here, $\mathbf{F}$ is the audio feature of the whole song and $\mathbf{C} = \{C^{(1)}, ..., C^{(T)}\}$ is the target chord sequence with each $C^{(i)} \in V$. The observation potential function ϕ takes the form:

$$\phi(C^{(t)}, \mathbf{F}) = \exp \sum_{i=1}^{6}\sum_{k=1}^{N_i} \mathbb{I}[k = c_i^{(t)}] \log a_k^{(t,i)} \quad (4)$$

where $\mathbf{a}^{(t,i)}$ are the activation vectors given by the Convolutional Recurrent Neural Network (CRNN) and $\mathbf{c}^{(t)}$ is the encoded chord vector for $C^{(t)}$.

The transition potential function ψ takes the form:

$$\psi(C^{(t-1)}, C^{(t)}) = \exp\left(-d \cdot \mathbb{I}[C^{(t-1)} \neq C^{(t)}]\right) \quad (5)$$

where $d = 30$ is a hyper-parameter that controls the degree of transition penalty. A larger d penalize more on chord transition and vice versa.

3.3 Class Re-weighting

Although we do not directly perform classification on distinct chord labels, the problem of class imbalance still persists during model training. For example, some chord extensions are very infrequent in the training set, leading to

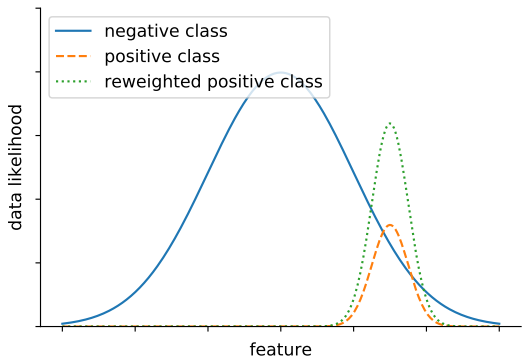


Figure 3. An illustration of why ambiguous class labeling causes problems. In this example, the class boundaries of the positive class and the negative class cross each other. Under the maximal likelihood principle, the accuracy of the positive class (with smaller prior than the negative class) is greatly harmed as its data likelihood is mostly covered by the negative class. Class re-weighting alleviates this problem to some extent.

excessive negative samples and insufficient positive samples for training. Generally, imbalanced training samples make the training process biased as the model tends to focus more on optimizing classes with more samples.

We here stress another issue that potentially aggravates this bias in learning-based automatic chord recognition systems. Audio chord annotation is an ambiguous problem even for human experts [11]. This means that the "ground truth" chord label of a fixed audio piece may contain uncertainty if annotated by different experts [15]. This may cause class boundaries for different labels to cross each other. In this case, we inevitably suffer from accuracy loss for the ambiguity between classes, and classes with small priors often suffer most (see Figure 3 for an example).

To make up for the bias, we introduce the class re-weighting strategy by introducing a class-wise weight factor $w_k^{(i)}$ for each possible chord component value in equation (2). By the re-weighting term, we want classes with fewer training samples to gain larger weights. We adopt the following weight term:

$$w_k^{(i)} = \min\left\{\left(\frac{n_k^{(i)}}{\max_{k'} n_{k'}^{(i)}}\right)^{-\gamma}, w_{max}\right\} \qquad (6)$$

Here, $n_k^{(i)}$ denotes the number of training samples for class k for the i-th component. γ and w_{max} are hyper-parameters, where $0 \leq \gamma \leq 1$ is the balance factor and $w_{max} \geq 1$ is the clamping value. A smaller γ will result in a more balanced weights and vice versa.

4. EXPERIMENTS

As previous chord recognition systems adopt smaller chord vocabularies compared to us, it is hard to make a direct comparison between different systems. Therefore, we divide the experiment results into two parts.

In the first part, we will compare the performance of our system on traditional chord evaluation metrics, most of which perform evaluations on simplified chord vocabularies. The evaluation metrics are calculated by the python package `mir_eval` [26]. In the second part, we will directly evaluate the system performance on larger chord vocabularies.

4.1 Datasets

We use 1217 songs from Isophonics, Billboard and MARL collections collected by Humphrey and Bello [11, 20] to form the dataset. To make a fair comparison, we adopt the 5-fold cross-validation with the same train/validation/test splits (60% for training, 20% for validation and 20% for testing) as in [20].

4.2 Pre-processing

We extract the Constant-Q Transform (CQT) spectrogram from the audio by the librosa [21] package with a sample rate of 22050 and a hop length of 512. We use the pitch range from midi note C1 (inclusive) to C7 (exclusive) with 36 filter banks per octave (252 CQT filter banks in total).

Data augmentation is performed by the pitch-shifting operation (from -5 semitones to +6 semitones) on the training set. Augmented features are directly calculated by shifting CQT spectrograms. The annotated chord labels are shifted accordingly.

4.3 Model Training

We use the Adam optimizer [14] to optimize the neural network parameters with a scheduled learning rate adjustment. To begin with, we use a learning rate 1e-3. If the validation loss does not improve in 5 epochs, we decrease the learning rate by 90%. We stop training after the learning rate drops below 1e-6.

In each epoch, a 1000-frame segment (approximately 23.2 seconds) is randomly selected from each song. 24 such segments form a mini-batch for gradient calculation.

4.4 Comparative Results

We first evaluate the performance of our system under traditional metrics proposed by [25]. For different metrics, the model outputs and ground truth chords are first mapped to some small vocabularies by removing extra notes and/or inversions. The scores are calculated by their matching percentages. The adopted metrics are: root only, root and thirds, major/minor chords, basic triads, sevenths, tetrads and MIREX (at least three correct notes between reference and estimated chords). All of these scoring methods ignore extended chord degrees (e.g., ninth, eleventh and thirteenth).

To evaluate the effect of our chord representation versus a plain classifier, we also perform evaluations on a modified version of the model, which contains a linear layer and a *flat* softmax output layer for all possible combinations of roots and chord types present in the dataset. The model is denoted by "Flat" in Figure 4. Other compared models are (1) Ours: the proposed model without re-weighting; (2) Ours+R: the proposed model trained with re-weighting factors ($\gamma = 0.5, w_{max} = 10$); (3) ACE18: an early design

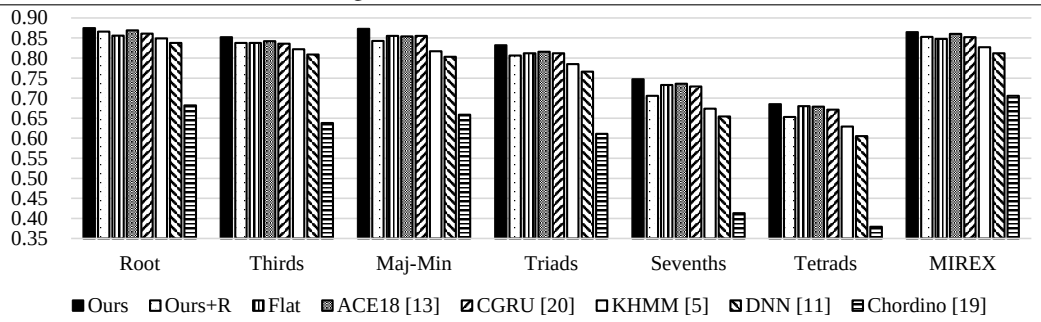


Figure 4. Comparison of median weighted recall scores.

of our model submitted to the MIREX 2018 contest [13] trained on the full vocabulary without beat alignment post-processing; (4) CGRU: A Convolutional Gated Recurrent Units (CGRU) model by [20]; (5) KHMM: A k-stream HMM by [5]; (6) DNN: A deep convolutional network by [11]; (7) Chordino: the classic baseline with template matching and HMM by [19].

In the results, Our system outperforms the baseline systems in all metrics, indicating that our system has a good performance on simplified chord vocabularies, even if the system is not trained for these vocabularies on purpose.

We can also see that the model with class re-weighting actually does not outperform the model without class re-weighting. The main reason is that class re-weighting assumes a more balanced class distribution, which is not the case of our test dataset. We will show in section 4.5.1 that class re-weighting actually has a better performance if we penalize the misclassification error of different classes equally, regardless of their frequency in the test dataset.

4.5 Performance on Large Chord Vocabulary

4.5.1 Evaluation on Common Chord Labels

To evaluate the chord recognition system on a large chord vocabulary, we first evaluate the system performance on common chord labels. We construct the target chord vocabulary V that includes the following chord qualities:

- Basic triads: `maj`, `min`, `aug`, `dim`
- Inverted triads: `maj/3`, `maj/5`, `min/b3`, `min/5`
- Seventh chords: `maj7`, `7`, `min7`, `dim7`, `hdim7`
- Extended chords: `maj9`, `9`, `min9`, `11`, `13`
- Suspended chords: `sus4`, `sus2`, `sus4(b7)`
- Slash chords: `maj/2`, `maj/b7`, `min/2`, `min/b7`
- Non-chord class: `N`

All chord qualities except `N` can be applied to 12 roots, resulting in 301 distinct chord classes. Under such a constraint, the model will not output other chord classes. To define the evaluation metrics, we use D to denote all pairs of estimated chords and reference chords (in vocabulary V) over the test dataset on the same frame. The per-class accuracy $\mathrm{acc}_{\mathrm{chord}}(C)$ for chord class $C \in V$ can be defined as:

Model	Frame-wise Acc.	Class-wise Acc.
No Re-weighting	**0.7719**	0.3475
(0.3,10.0)	0.7609	0.3745
(0.5,10.0)	0.7459	**0.4022**
(0.7,20.0)	0.7146	0.3738
(1.0,20.0)	0.6577	0.3832

Table 2. Mean frame-wise accuracy and mean class-wise accuracy for the proposed model with different class re-weighting factors (γ, w_{max}) for training. The re-weighting factors are shown in the model column.

$$\mathrm{acc}_{\mathrm{chord}}(C) = \frac{\sum_{(C_{\mathrm{est}},C_{\mathrm{ref}})\in D} \mathbb{I}[C_{\mathrm{est}} = C]\mathbb{I}[C_{\mathrm{ref}} = C]}{\sum_{(C_{\mathrm{est}},C_{\mathrm{ref}})\in D} \mathbb{I}[C_{\mathrm{ref}} = C]} \tag{7}$$

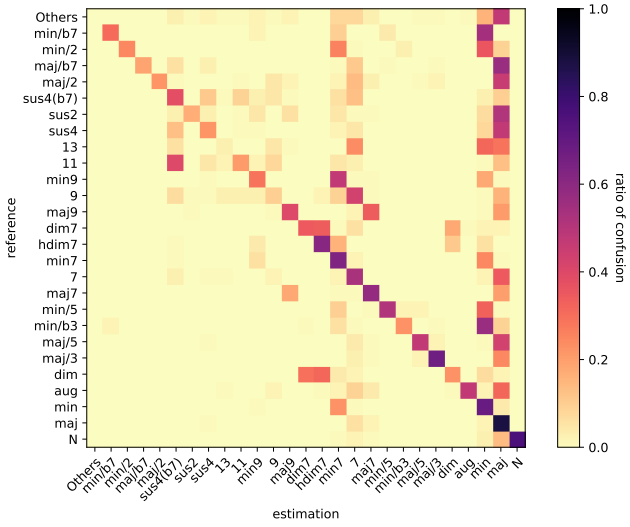


Figure 5. The quality confusion matrix of model trained with re-weighting factors $\gamma = 0.5, w_{max} = 10$.

We believe that a good chord recognition system should have high frame-wise accuracy, as well as a low bias among different chord classes. For example, a system with high bias may recognize all samples of one chord class C into another, resulting in a low score for $\mathrm{acc}_{\mathrm{chord}}(C)$. Therefore, we want the average of $\mathrm{acc}_{\mathrm{chord}}(C)$ over all chord classes as high as possible in the class-wise measurement. Formally, we define the mean frame- and class-wise

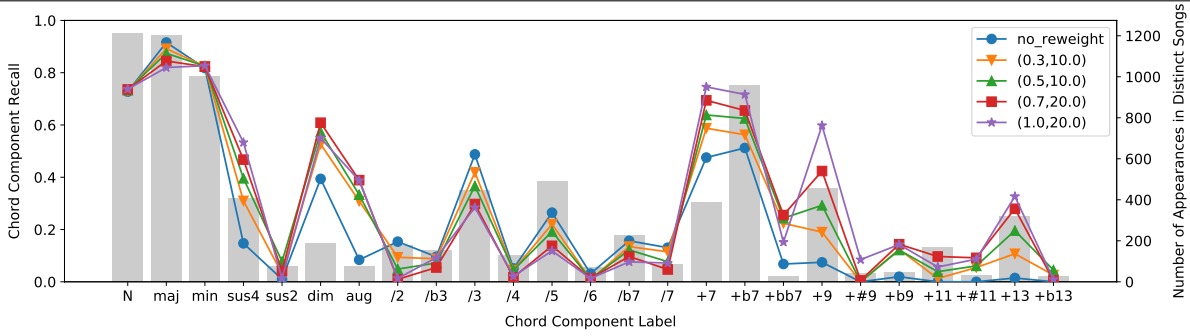


Figure 6. Recall values for different chord components on test sets (denoted by lines) and song-level component counts in the whole dataset (denoted by bars). Different lines denote the models trained with different re-weighting factors (γ, w_{max}).

accuracy as:

$$\text{acc}_{\text{frame}} = \frac{1}{|D|} \sum_{(C_{\text{est}}, C_{\text{ref}}) \in D} \mathbb{I}[C_{\text{est}} = C_{\text{ref}}] \quad (8)$$

$$\text{acc}_{\text{class}} = \frac{1}{|V|} \sum_{C \in V} \text{acc}_{\text{chord}}(C) \quad (9)$$

We evaluate our system with different class re-weighting parameters under these two metrics. The results are shown in Table 2. From the results, we can see that all models with class re-weighting have a lower mean frame-wise accuracy and a higher mean class-wise accuracy compared to the model without class re-weighting. This indicates that class re-weighting alleviates the class bias problem to some extent while leading to a trade-off that the frame-wise accuracy is harmed.

We also show the within-root quality confusion matrix (i.e., quality confusion matrix when the estimated root is correct) of the model trained with re-weighting factors ($\gamma = 0.5, w_{max} = 10$) in Figure 5 for error analysis. Two major trends can be observed. First, most quality errors are from mapping complex chord qualities to simpler ones. For example, `maj9`, `9` and `min9` are mapped to `maj7`, `7` and `min7`; tetrads are mapped to triads. Second, certain extended chord qualities like `13` and `11` are hard to be detected. Despite of the label bias, we also suspect that annotated extended qualities may omit some degrees in practice (e.g., a missing eleventh in a `13` chord) but the omission is not always annotated in the test set. The misclassification of `11` chords to `sus4(b7)` is also reasonable as a `11` chord without a third and a ninth has the same pitch classes as a `sus4(b7)` chord.

4.5.2 *Evaluation on Chord Components*

To further evaluate the system's performance on a large chord vocabulary, we adopt a full chord vocabulary (i.e., a chord vocabulary containing all chord qualities present in the dataset) for the decoding process and evaluate the class-wise accuracy for each chord component. For the root and bass category, we are interested in if their relationship (chord inversion) is correctly detected. The slash notation is defined as the interval between bass and root. For example, In the chord quality `maj/5`, the fifth of the chord is the bass note. Accuracy on slash notations reflects the model's ability to detect chord inversions.

Also, we want to evaluate how class re-weighting affects the accuracy of rare chord component labels. Therefore, we experiment with different hyper-parameters for class re-weighting and make a comparison as shown in Figure 6.

In Figure 6, it is clear to show that the model with no re-weighting declines to predict certain extensions including `11` and `13`. When class re-weighting is adopted, we can observe an accuracy boost for these components. However, the trade-off of class re-weighting is the fact that it harms the accuracy of some other classes as well as providing false positive predicts.

Also, it can be observed that the accuracy for a specific component class has a positive correlation with the number of its appearances in the dataset. Some label components surely have too few samples for the model to learn the correct acoustic and semantic properties, resulting in a low accuracy even if class re-weighting is performed.

5. CONCLUSION

In this paper, we propose a method to perform audio chord recognition on a large chord vocabulary. The core idea of the model is to recognize each chord component instead of the whole chord label itself, and each chord component has a much smaller vocabulary that is easier for the model to handle. We show by experiment that the model acquires state-of-the-art performance on traditional chord evaluation metrics. Also, we demonstrate its ability to detect rare chord component values.

However, the model still has several unsolved issues. First, the model still has strong class bias even if we adopt the re-weighting strategy, and the model performance on some rarest chord component values are still not satisfactory. Second, the proposed representation is not yet perfect. More theoretical analysis and experiments are required to design the best chord representation method for the audio chord recognition task.

We also stress the class ambiguity issue with regard to the datasets. Although a class-balanced dataset is nearly unfeasible in the realm of audio chord recognition, we would like to call for datasets with more precise annotations as well as a more thorough analysis of currently available datasets.

6. REFERENCES

[1] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio chord recognition with recurrent neural networks. In *ISMIR*, pages 335–340. Citeseer, 2013.

[2] John Ashley Burgoyne, Jonathan Wild, and Ichiro Fujinaga. An expert ground truth set for audio chord recognition and music analysis. In *ISMIR*, volume 11, pages 633–638, 2011.

[3] Emre Cakır, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1291–1303, 2017.

[4] Emilios Cambouropoulos, Maximos A Kaliakatsos-Papakostas, and Costas Tsougras. An idiom-independent representation of chords for computational music analysis and generation. In *ICMC*, 2014.

[5] Taemin Cho. *Improved techniques for automatic chord recognition from music audio signals*. PhD thesis, New York University, 2014.

[6] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2392–2396. IEEE, 2017.

[7] Jun-qi Deng and Yu-Kwong Kwok. A hybrid gaussian-hmm-deep learning approach for automatic chord estimation with very large vocabulary. In *ISMIR*, pages 812–818, 2016.

[8] Jun-qi Deng and Yu-Kwong Kwok. Large vocabulary automatic chord estimation with an even chance training scheme. In *ISMIR*, pages 531–536, 2017.

[9] Christopher Harte, Mark B Sandler, Samer A Abdallah, and Emilia Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *ISMIR*, volume 5, pages 66–71, 2005.

[10] Eric J Humphrey and Juan P Bello. Rethinking automatic chord recognition with convolutional neural networks. In *2012 11th International Conference on Machine Learning and Applications*, volume 2, pages 357–362. IEEE, 2012.

[11] Eric J Humphrey and Juan Pablo Bello. Four timely insights on automatic chord estimation. In *ISMIR*, pages 673–679, 2015.

[12] Eric J Humphrey, Taemin Cho, and Juan P Bello. Learning a robust tonnetz-space transform for automatic chord recognition. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 453–456. IEEE, 2012.

[13] Junyan Jiang, Ke Chen, Wei Li, and Guangyu Xia. Mirex 2018 submission: A structural chord representation for automatic large-vocabulary chord transcription. *MIREX evaluation results*, 2018.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] HV Koops, WB de Haas, J Bransen, and A Volk. Chord label personalization through deep learning of integrated harmonic interval-based representations. In *Proc. of the Int. Workshop on Deep Learning and Music. Anchorage, US.*, 2017.

[16] Filip Korzeniowski and Gerhard Widmer. Feature learning for chord recognition: The deep chroma extractor. *arXiv preprint arXiv:1612.05065*, 2016.

[17] Filip Korzeniowski and Gerhard Widmer. A fully convolutional deep auditory model for musical chord recognition. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2016.

[18] Filip Korzeniowski and Gerhard Widmer. Improved chord recognition by combining duration and harmonic language models. In *ISMIR*, pages 10–17, 2018.

[19] Matthias Mauch and Simon Dixon. Approximate note transcription for the improved identification of difficult chords. In *ISMIR*, pages 135–140, 2010.

[20] Brian McFee and Juan Pablo Bello. Structured training for large-vocabulary chord recognition. In *ISMIR*, pages 188–194, 2017.

[21] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25, 2015.

[22] Matt McVicar, Raúl Santos-Rodríguez, Yizhao Ni, and Tijl De Bie. Automatic chord estimation from audio: A review of the state of the art. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(2):556–575, 2014.

[23] Yizhao Ni, Matt McVicar, Raul Santos-Rodriguez, and Tijl De Bie. An end-to-end machine learning system for harmonic analysis of music. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1771–1783, 2012.

[24] Hélene Papadopoulos and Geoffroy Peeters. Large-scale study of chord estimation algorithms based on chroma representation and hmm. In *2007 International Workshop on Content-Based Multimedia Indexing*, pages 53–60. IEEE, 2007.

[25] Johan Pauwels and Geoffroy Peeters. Evaluating automatically estimated chord sequences. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 749–753. IEEE, 2013.

[26] Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel PW Ellis, and C Colin Raffel. mir_eval: A transparent implementation of common mir metrics. In *ISMIR*. Citeseer, 2014.

[27] Yushi Ueda, Yuki Uchiyama, Takuya Nishimoto, Nobutaka Ono, and Shigeki Sagayama. Hmm-based approach for automatic chord detection using refined acoustic features. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5518–5521. IEEE, 2010.

[28] Yiming Wu and Wei Li. Music chord recognition based on midi-trained deep feature and blstm-crf hybird decoding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 376–380. IEEE, 2018.

Session F

BANDNET: A NEURAL NETWORK-BASED, MULTI-INSTRUMENT BEATLES-STYLE MIDI MUSIC COMPOSITION MACHINE

Yichao Zhou[1,2,*] **Wei Chu**[1] **Sam Young** [1,3] **Xin Chen**[1]

[1] Snap Inc. 63 Market St, Venice, CA 90291,
[2] Department of EECS, University of California, Berkeley,
[3] Herb Alpert School of Music, University of California, Los Angeles,
zyc@berkeley.edu, wei.chu@liulishuo.com, samyoungmusic@gmail.com, xin.chen@snap.com

ABSTRACT

In this paper, we propose a recurrent neural network (RNN)-based MIDI music composition machine that is able to learn musical knowledge from existing Beatles' music and generate full songs in the style of the Beatles with little human intervention. In the learning stage, a sequence of stylistically uniform, multiple-channel music samples was modeled by an RNN. In the composition stage, a short clip of randomly-generated music was used as a seed for the RNN to start music score prediction. To form structured music, segments of generated music from different seeds were concatenated together. To improve the quality and structure of the generated music, we integrated music theory knowledge into the model, such as controlling the spacing of gaps in the vocal melody, normalizing the timing of chord changes, and requiring notes to be related to the song's key (C major, for example). This integration improved the quality of the generated music as verified by a professional composer. We also conducted a subjective listening test that showed our generated music was close to original music by the Beatles in terms of style similarity, professional quality, and interestingness. The generated music samples can be downloaded at `https://goo.gl/uaLXoB`.

1. INTRODUCTION

Automatic music composition has been an active research area for the last several decades, and researchers have proposed various methods to model many different kinds of music. [7,8,12,23,26] used rules and criteria developed by professional musicians to generate songs. These methods usually relied heavily on the input of music experts, handcrafted rules, consistent intervention during the process of composition, and fine-tuning the generated music in the post-processing stage. Although the quality of the composed music may be quite satisfactory, the composition process can be time-consuming and the composed music can be biased toward a particular style. Recently, agnostic approaches that do not depend on expert knowledge have been emerging [9]. Instead of relying on music experts, these methods employ a data-driven approach to learn generalizable theory and patterns from existing pieces of music, and this approach has proven to be effective. For example, [2,15] trained a hidden Markov model from music corpora and [10] modeled polyphonic music from the perspective of the graphic model. With the recent progress made in deep learning, there have been many research efforts that have tried to compose music using neural networks: [27] used a deep convolutional network to generate a melody conditioned on 24 basic chord triads found in each measure; [19] generated the drum pattern for songs using an RNN [13]; [9, 14, 17, 18] described RNN approaches to modeling and harmonizing Bach-style polyphonic music; and [5] proposed a multi-layer RNN to model pop music by encoding drum and chord patterns as one-hot vectors.

While most of the aforementioned machine-learning methods were able to generate music in some categories such as Bach chorale and folk music, we found that it is challenging to use their methods to model songs by the Beatles. Formed in 1960, The Beatles are arguably one of the most influential bands of all time. Their primary songwriters, John Lennon and Paul McCartney, were considered masters and many of their songs are still well known today. The Beatles music drew on elements of 50s rock and roll, and their musical style can be characterized by catchy vocal melodies, unique chord progressions, and an upbeat, energetic sound. The standard instrumentation of the Beatles contains vocals, two electric guitars, bass, drums, and occasional piano.

One difficulty of replicating the Beatles' music is that all the component parts depend on each other but have different characteristics. For example, the bass line is often monophonic while the guitar chords are polyphonic, and the guitar chords are likely to contain certain notes found in the bass part. The model needs to be able to generate different instrumental parts within a uniform musical structure. In addition, the style of the musical features of-

*This work was done when Yichao Zhou was an intern at Snap Inc.

ten changes between songs. For example, many Beatles' songs use monophonic vocal melodies while others use polyphonic, two-part vocal melodies. The chords in the Beatles' music often contain a lot of non-standard combinations of notes that are different from the common chord triads, with the added complexity that certain chords may be incomplete and missing one or more of their component parts. They can be played by either a piano or a guitar, each of which uses different chord spacing. All of these variations are challenging to model for a machine learning algorithm. Moreover, the Beatles are known for using complex harmonies that can be difficult to classify, with the added complication that certain chords may be incomplete or missing one or more of their component parts. Thus it may not be appropriate to encode the chord progression aspect of the music as one-hot vectors [27], as they treat two similar harmonies differently.

To overcome these difficulties, we introduce BandNet, an RNN-based, Beatles-style multi-instrument music composition machine. We exploit the song structures that can be commonly found in pop music to generate complete songs without relying too much on labeled data. Our system requires little expert knowledge to use and it can be successfully trained on a relatively small corpus. We explain the proposed approach in Section 2 and evaluate the performance of our algorithm in Section 3.

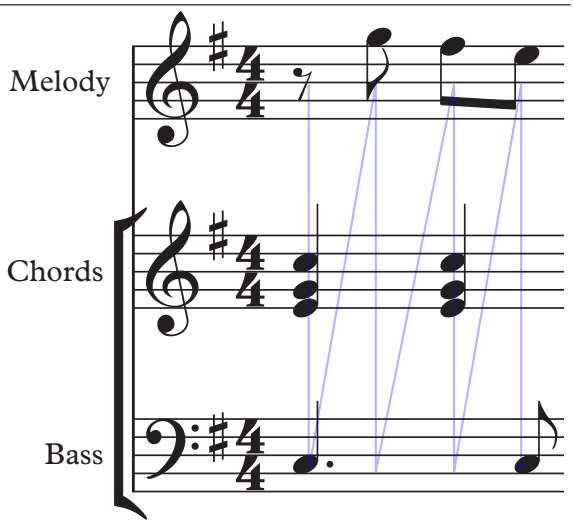
2. METHODS

2.1 Data Representation

Our BandNet uses MIDI files as input and output and utilizes the same data processing pipeline from Magenta [4]. For each Beatles' song, we consider the three most important channels: the vocal melody, guitar chords, and bass part. All the channels are allowed to be polyphonic, to maximize the flexibility of the model.

In our dataset, we include all the songs that use a 4/4 time signature, which means that a quarter note is felt as the beat, and each measure (a.k.a one bar, a short segment of music whose boundaries are shown by vertical bar lines in the score) has four beats. It is reasonable to discretize note lengths into sixteenth notes. We call the duration of a sixteenth note a *step*. Therefore, each measure is discretized into 16 steps and each beat is discretized into 4 steps.

Because a song may be played by different instruments with different pitch ranges, we first transpose the pitch by octave so that the average pitch of each channel in each song is as close as possible to the global pitch average of that channel. Next, we transpose each song by -5 to 6 semitones to augment the training data 11 times so that it is able to generate music in all possible keys. Other approaches, such as transposing each song to the same key, C major for example, do not work well for the Beatles' music because we have yet to find a reliable way to detect the key of each song.




```
01. NXT_CHNL          16. NEW_NOTE(F5)
02. NEW_NOTE(C5)      17. NXT_CHNL
03. NEW_NOTE(G4)      18. NEW_NOTE(C5)
04. NEW_NOTE(E4)      19. NEW_NOTE(G4)
05. NXT_CHNL          20. NEW_NOTE(E4)
06. NEW_NOTE(C3)      21. NXT_CHNL
07. NXT_STEP          22. CNT_NOTE(C3)
08. NEW_NOTE(G5)      23. NXT_STEP
09. NXT_CHNL          24. NEW_NOTE(E5)
10. CNT_NOTE(C5)      25. NXT_CHNL
11. CNT_NOTE(G4)      26. CNT_NOTE(C5)
12. CNT_NOTE(E4)      27. CNT_NOTE(G4)
13. NXT_CHNL          28. CNT_NOTE(E4)
14. CNT_NOTE(C3)      29. NXT_CHNL
15. NXT_STEP          30. NEW_NOTE(C3)
```

Figure 1: An example showing how we encode an excerpt from *I Want to Hold Your Hand (1964)*. Notes are quantized to eighth notes rather than sixteenth notes for demonstration purposes. The sheet music example is shown at the top where the scan lines are marked in blue. The encoded sequence of the sheet music is shown at the bottom.

2.2 Score Encoding

BachBot [17] and Magenta [4] convert polyphonic MIDI music into a sequence of symbols so that RNN can be used to model the probabilistic distribution of such a sequence. We expand their schemes to music with multiple channels.

Figure 1 gives an example showing how we encode the music score. We create a new type of symbol `NXT_CHNL`, along with the three existing categories: `NEW_NOTE`, `CNT_NOTE`, and `NXT_STEP`. The strategy is to scan the score in a left to right (time dimension), top to bottom (channel dimension), zig-zag fashion. Each time we meet a note during the scan, we will first check whether it is a new note or a continuation of a previous note (e.g., the second sixteenth interval of an eighth note). We will then either emit a `NEW_NOTE` or a `CNT_NOTE` symbol depending on the case, followed by the pitch of that note. When a channel is polyphonic, the note with higher pitch will al-

ways be in front of the notes with lower pitch according to this strategy. When the scan line comes across the boundary of a channel, we will emit a `NXT_CHNL` symbol, and when the scan line comes across a time step, we will emit a `NXT_STEP`. Unlike other common methods where each symbol will represent all the notes inside a time step, we decompose them into multiple symbols and the advancement of the time step is explicitly expressed using the symbol `NXT_STEP`.

2.2.1 Note Features

With the previous encoding mechanism, we can encode any of the Beatles' songs into a sequence $S = \{S_i\}_{i=0}^{N}$. Here $S_i \in \mathcal{S}$ in which $\mathcal{S}$ is the set of all the possible symbols. We have $|\mathcal{S}| = |T| * 2 + 2$, where T is the set of possible pitches.

Because the training data is limited, it is helpful to incorporate additional features for each symbol to help the neural network learn the theory and patterns of the music. We pair each symbol S_i with its feature F_i when we feed the encoded sequence into the RNN. We designed two features for BandNet, i.e., $F_i = (B_i, G_i)$. The feature $B_i \in \{0, 1\}^5$ contains the beat information. $B_i = 1$ if and only if the global time step of ith symbol is a multiple of 2^i. We find that this feature is helpful for the RNN to keep the style of the chord channel consistent inside a measure. The second feature $G_i \in \{0, 1\}$ represents whether the melody will be generated at the current time step. Without this feature, we find that sometimes BandNet will not generate a vocal melody due to silences in the melody channel of the training data (usually because of an instrumental or guitar solo section). By setting this variable to one or zero, we can easily control whether we want to generate the vocal part in a given section of music.

2.3 Network Structure

Figure 2 shows how a classical multi-layer LSTM-RNN [13] models the probabilistic distribution of the symbol sequence. At the bottom layer, each LSTM cell takes the symbol S_i in its one-hot vector form together with the corresponding binary feature vector F_i as its input I_i. These LSTM cells are chained so that they will apply nonlinear transformations to the previous cell state C_{i-1}^1 and input I_i and produce the current hidden state h_i^1 and cell state C_i^1, respectively. In order to increase the nonlinearity of the model, we make the network deep by stacking multiple layers of LSTM cells. Starting from the second layer, each cell will take the hidden state from the previous layer as input. Finally, we apply a linear transformation to the hidden states in the last layer with softmax to compute the conditional probability $P_\Theta(S_{i+1} \mid I_{\{1\cdots i\}})$, where Θ contains the parameters of the network. We use BPTT [20] to find the parameters that locally maximizes the likelihood of the training data.

2.4 Keeping Notes in the Key

By using the LSTM-RNN and the encoding schemes from previous sections, our generative model is able to compose multi-instrument music. During the test, we find that the melody channel generated by the LSTM occasionally contained some unexpected notes. We found that many of these notes are dissonant because they are not in the key of the music. We speculate that this is because the Beatles often used notes in their music that deviated from conventional practices of other popular music. These notes may work well under some conditions, but the amount of data does not allow our neural network to learn how to use these notes in the right context. Therefore, in order to improve the quality of our music, it is reasonable to filter them out in BandNet, i.e., restricting the notes that are not in the song's key during the generating stage. This can be achieved by applying a mask to the probability distributions returned by the neural network and re-normalizing them so that they all sum to unity.

2.5 Generating a Complete Song

Most of the Beatles' music has a repetitive and sectional song structure. Figure 3 shows an example of the structure in the song *Yesterday (1965)*. This song uses an *AABABA* structure, where the A section is called the *verse* and the B section is called the *chorus*. The verse section is repeated four times, with each repetition being exactly the same or having only minor differences. It is hard for the RNN to learn this phenomenon because the distance between two sections is as long as eight measures, i.e., 128 time steps. RNN normally cannot carry hundreds of symbols in its memory across a span of that long. Folk-RNN [25] used a data format called ABC notation that has an annotation for repeating sections so that they do not need to deal with this problem. We do not have such fine-level annotation in our dataset. Instead, we use a template-based method to generate structured music. Users of BandNet will first select a predefined song structure template, e.g., AABA or ABABCBB, and then BandNet can generate a clip for each section whose length can vary from 4 to 16 measures. After that, we assemble the generated clips to form a complete song. Because we do not model the drum pattern in this work, we assign a precomposed drum pattern for each section of music, which is beneficial as we can select different styles of drum patterns for different sections of the song.

The well-known DeepBach [9] and BachBot [17] can generate a new harmony or re-harmonize an existing melody from a single instrument, i.e. piano. BandNet can generate a song with multiple instruments, e.g. guitar, keyboard, bass, and drum. Because we do not have a melody to condition on, BandNet needs a short sequence of notes, also known as a *seed*, to begin a section. Although in theory it is possible not to condition on any seeds, we found that the resulting music was often unsatisfactory. In order to avoid depending on a professional musician to compose note sequences as seeds, we adopt the following strategy: First, we let BandNet generate long sequences of music without conditioning on any seeds. Second, we can listen to these randomly generated segments and mark the clips that sound most compelling to us. Third, we use these clips

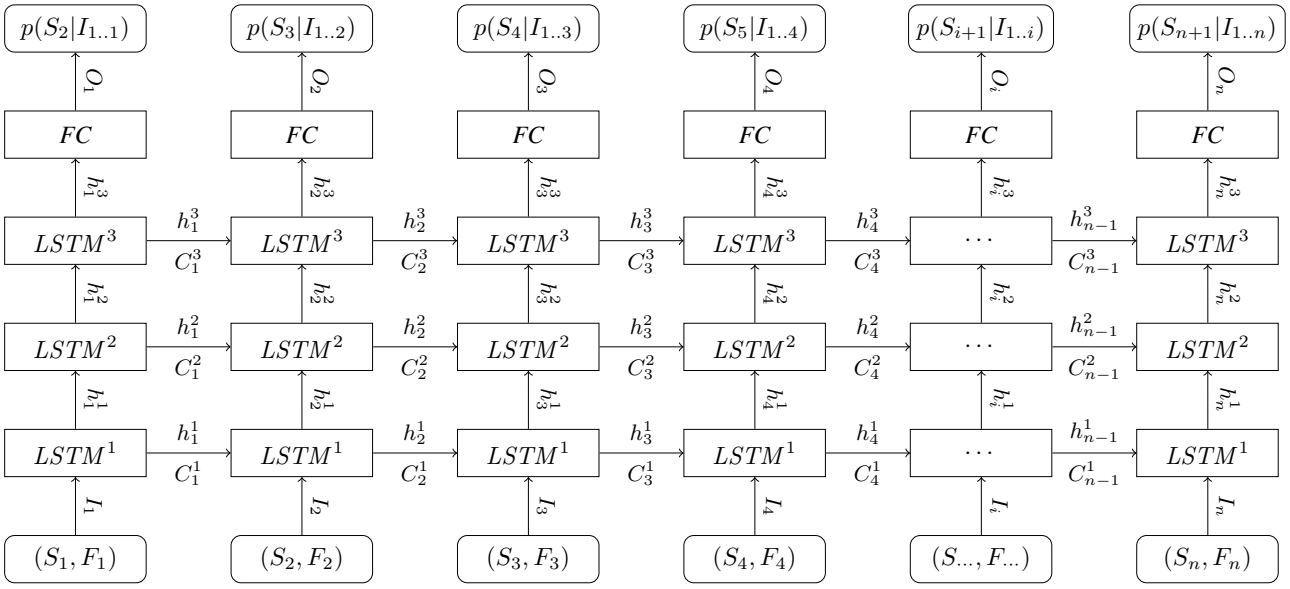


Figure 2: A diagram showing how an unrolled 3-layer LSTM-RNN works for music composition. Here, symbol S_i and feature F_i are encoded to the vector I_i. $LSTM^j$ represents an LSTM cell in the jth layer. Cells in the same layer share the same parameter. C_i^j and h_i^j are the cell state and hidden state of the ith cell in the jth layer. FC represents a fully-connected layer and its output O_i is fed into a softmax function to produce a distribution over all the possible symbols.

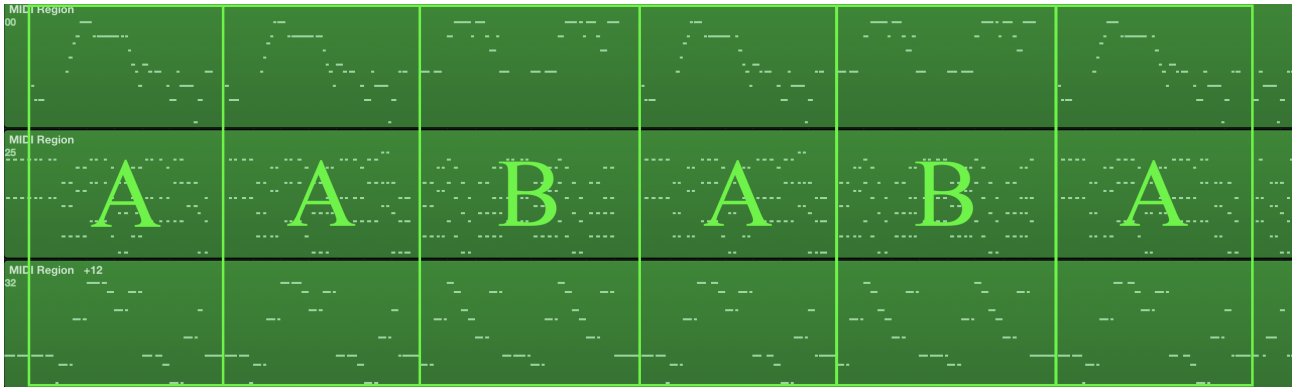


Figure 3: The piano roll of the song *Yesterday (1965)*. It has a song structure AABABA, whose sections are labeled in green in the Figure. The channels from top to bottom are melody, chords, and bass line.

	Melody		Chords		Bass		Overall	
	CQ	SQ	CQ	SQ	CQ	SQ	ACSQ	GSQ
MGT-M	2.60 ± 1.14	2.70 ± 0.84	-	-	-	-	-	2.65 ± 0.65
MGT-P	-	-	3.20 ± 0.57	2.50 ± 0.35	-	-	-	2.85 ± 0.22
BN	2.90 ± 0.55	1.50 ± 0.50	2.70 ± 0.76	2.40 ± 0.82	3.30 ± 0.67	2.40 ± 0.82	2.53 ± 0.25	2.60 ± 0.65
BN-S	2.90 ± 0.42	2.50 ± 0.87	3.05 ± 0.76	2.90 ± 0.65	3.20 ± 0.76	3.20 ± 0.45	2.96 ± 0.26	2.95 ± 0.62
BN-SB	2.90 ± 0.52	3.40 ± 0.22	2.85 ± 0.42	3.25 ± 0.40	3.30 ± 0.27	3.25 ± 0.40	3.16 ± 0.30	3.10 ± 0.42
BN-SBK	3.85 ± 0.49	3.75 ± 0.25	3.45 ± 0.51	3.45 ± 0.57	3.75 ± 0.25	3.65 ± 0.22	3.65 ± 0.13	3.90 ± 0.38
BEATLES	$\mathbf{4.45 \pm 0.37}$	$\mathbf{4.80 \pm 0.11}$	$\mathbf{4.20 \pm 0.27}$	$\mathbf{4.75 \pm 0.43}$	$\mathbf{4.40 \pm 0.22}$	$\mathbf{4.95 \pm 0.11}$	$\mathbf{4.59 \pm 0.13}$	$\mathbf{4.65 \pm 0.22}$

Table 1: Results of a professional composer evaluating the quality of music generated by different models. **MGT-M**: Magenta's MelodyRNN, **MGT-P**: Magenta's PolyphonyRNN, **BN**: BandNet without note features, **BN-S**: BN with **s**ilence features, **BN-SB**: BN-S with **b**eat features, **BN-SBK**: BN-SB while keeping notes in the **k**ey, **BEATLES**: original Beatles' songs. The definitions of CQ, SQ, ACSQ, and GCQ can be found in Section 3.2.

as seeds for BandNet to generate all the sections of the song.

3. EXPERIMENTS

3.1 Settings and Datasets

We collected 183 Beatles MIDI songs from the Internet as our training dataset. We removed 60 songs from the dataset because they were either divergent in musical style when compared with other Beatles' songs, or were missing important components such as a clear vocal melody or bass line. We found that MIDI files in the wild can be messy. For example, the chords may be divided across three channels in some MIDI files, while there can be up to eight channels used for instrumental decoration in others, which is not necessary for our purposes. We cleaned this dataset by deleting the unnecessary channels and merging the fragmented channels.

Due to the number of songs that the Beatles composed, the size of our dataset is smaller compared to those used in the literature [5, 17, 27], but we found that it is sufficient to train a reasonably good model. Aside from its influence in popular music history, there are two reasons why we choose to use the Beatles' catalog as our training dataset: First, the style of the Beatles' music is relatively consistent when compared to other categories of pop music, and therefore it is easier for the RNN to learn its underlying structures. Second, most of the Beatles' music contains the elements required by our music generation pipeline, such as distinct melody, chord, and bass parts, as well as repeating song structures, which can be missing in genres such as classical and folk music.

The two most important parameters of the recurrent neural network were the dimension of LSTM cells and the number of layers. We found that a 3-layer RNN in which each LSTM cell had 256 hidden units worked well in practice.

Our implementation was based on Magenta [4] and Tensorflow [1] for processing the MIDI files and training the RNN. Because the number of parameters in our network was large, we applied dropout [24] to alleviate overfitting. We trained our model using the Adam optimizer [16], which is a variant of stochastic gradient descent that is not sensitive to the global learning rate. We used 10% songs in our dataset for cross validation and we stopped the training process when the error on the validation dataset no longer decreased. During the training, we clipped the gradients so that their L2-norms were less than or equal to 1. This technique was proposed in [24] to alleviate the gradient explosion problem.

3.2 Quality Scoring by a Professional Composer

In this section, a professional music composer evaluated the music generated by each subsequent version of BandNet. The composer gave two scores for each individual channel (melody, chords, and bass) based on their musical content and structure. The Content Quality (CQ) was defined as how well the notes and rhythms in the generated music function according to music theory principles consistent with the music of the Beatles, and the Structure Quality (SQ) was defined as to what extent the music sample exhibits an organizational structure. All scores were given on a scale of 1 to 5. In addition, we designed two overall scores to evaluate the overall quality of each multiple-channel song. The Averaged Content and Structure Quality (ACSQ) were calculated through averaging the CQs and SQs of all the channels, and the Group Synergy Quality (GSQ) score evaluated how well the individual channels work together to make a unified whole.

The results are shown in Table 1. The score was an average across five songs under each setting. We found that model BN was on par with Magenta's melody and polyphony generators [4] in terms of content and structure scores, which is reasonable because models from Magenta were designed to model melody and chords (as in polyphonic music) separately, and modeling them jointly in the case of BandNet would not improve the score of each individual channel. After introducing the silence feature, the GSQ of BandNet increased from 2.6 to 2.95 because we were able to exclude unusual silences in the melody. By adding the beat feature, BandNet continued to receive rewards in SQs for the melody and chord channels; a possible explanation for this is that the beat feature gave the RNN measure and section information, which helped it learn the structure of the music more efficiently. Both of these features also improved GSQs, as the normalization of each individual channel also improved the alignment between individual parts. Finally, the greatest improvement in both metrics was from the key restriction feature. This significantly improved the CQs of individual channels by removing "wrong" notes, and also improved SQs and GSQs by reducing the amount of notes that were dissonant with one another across individual channels.

3.3 Subjective Listening

We also conducted a subjective listening experiment to evaluate the quality of our generated songs from the perspective of amateurs. We received 17 responses in this user study: 16 said that they had never received formal musical training. In this test, we asked users to listen to 15 songs. All of the songs were in AABA structure and each section had a length of 8 measures. The first 5 songs, labeled as group A, were composed by BandNet using randomly generated seeds; the next 5 songs, labeled as group B, were composed by BandNet using professionally composed seeds. Each seed was 2 measures in length, with BandNet generating the remaining 6-measure clip for each section. Songs in group A and B were generated randomly without human selection. The last 5 songs, labeled as group C (the control group), included relatively unknown Beatles' songs, with the intention that listeners had likely never heard them before. We shuffled the order of the songs so that listeners could not guess whether a song was composed by BandNet prior to listening. We also modified the drum patterns for the group C Beatles' songs,

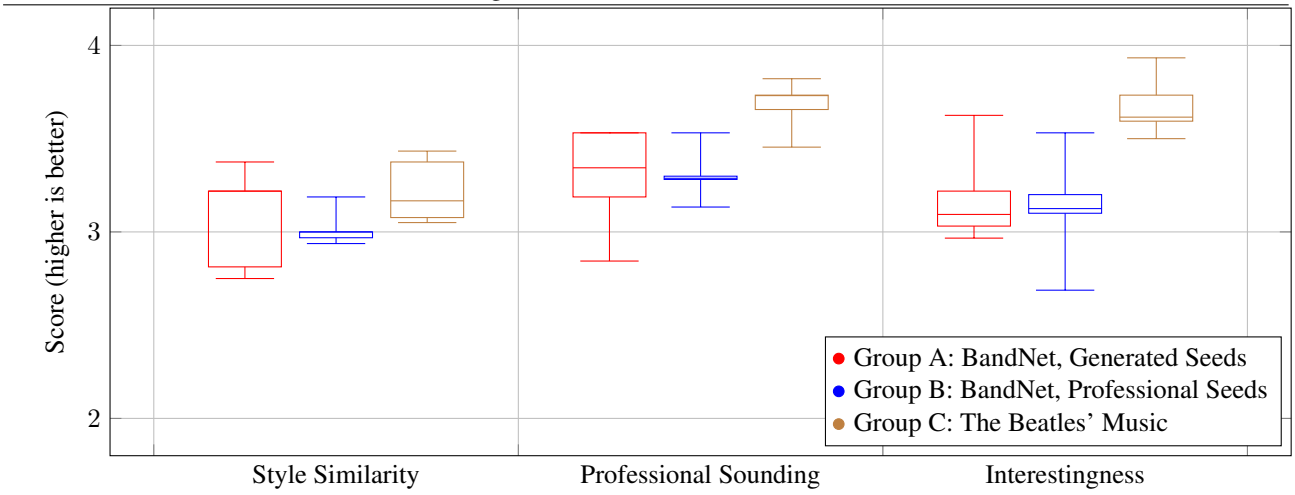


Figure 4: Result of a user study that evaluates the performance of different ways to generate music. The x-axis represents the sources of the music and the y-axis represents the score. The box plot shows the distribution of the average score of each song rated by the listener. From bottom to top, the horizontal lines of each box show the minimum, the first quartile, the median, the third quartile, and the maximum of the average score, respectively.

so that listeners could not distinguish them from BandNet-composed songs based on differences in the drum pattern.

At the beginning, we asked subjects to listen to 5 well-known songs by the Beatles, such as *I Want to Hold Your Hand (1964)*, in order to familiarize them with the Beatles musical style. Next, we asked them to listen to the 15 songs mentioned above and to answer the following 4 questions for each song:

Q1: Have you heard this song before?
Q2: Does it sound similar to the music of the Beatles?
Q3: How likely is it that this music was professionally composed?
Q4: How interesting is this music?

We asked listeners to only choose between "Yes, definitely!" and "No/Not sure" in Q1; if they answered "Yes", we removed their scoring of that song from our results. This is because a subject may be biased to give a song a higher score if he had heard it song before. For Q2, Q3, and Q4, we let users grade each song using a scale from 1 to 5 with an increment of 0.5. Figure 4 shows the distribution of those scores from 17 responses. The labels in the horizontal axis, Style Similarity, Professional Sounding, and Interestingness correspond to Q2, Q3, and Q4, respectively. Each sample in the box plot represents the average score over 17 responses to a question for a particular song.

For Q1, about 13.3% of responses indicated that they had heard those little-known Beatles' songs, while the percentages were only 0% and 1.3% for BandNet-generated songs using automatically-generated seeds and professional seeds, respectively. This could be an indicator showing that we did not overfit the training data and just replicated some clips from the original Beatles' music. For the rest of the questions, we found that the authentic Beatles' songs constantly outperformed the BandNet-generated songs, but only by a small margin. In particular, the average Style Similarity scores for songs in group A, B, and C are 3.08, 3.02, and 3.22, respectively. The score difference of Q2 between the authentic and generated songs was less than 0.202, which showed that BandNet was able to imitate the style of the Beatles relatively well. The average Professional Sounding scores were 3.29, 3.16, and 3.68, and the average Interestingness scores were 3.19, 3.13, and 3.68 for songs in group A, B, and C, respectively. The score gaps of Q3 and Q4 between authentic and generated songs were approximately 0.5. The musical knowledge that BandNet learned came primarily from The Beatles, and in theory may be difficult for an RNN-based machine learning algorithm to generate more professional and interesting music than The Beatles. Concerning the seeds used in generation, our experiments have shown that using professionally-composed seeds did not have a significant advantage over selecting from randomly-generated seeds in terms of subjective listening evaluation. This means that we may no longer need a composer in the loop for generating a complete song and an amateur would be able to "compose" a Beatles-style song without the guide of a professional by using BandNet.

4. CONCLUSIONS AND FUTURE WORKS

In this paper, we propose an RNN-based, multi-instrument MIDI music composition machine, which learns musical knowledge from existing Beatles' music and automatically generates music in the style of the Beatles with little human intervention. We also integrate expert knowledge into the data-driven based learning process. We prove that our method is effective in both professional evaluation and subjective listening tests. Our future work includes explicitly modeling the drum parts, designing a better neural network structure, employing Gibbs sampling, improving the evaluation metrics, and testing BandNet for other genres of music on a larger dataset.

5. REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Moray Allan and Christopher Williams. Harmonising chorales by probabilistic inference. In *Advances in neural information processing systems*, pages 25–32, 2005.

[3] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

[4] Google Brain. Magenta. `https://magenta.tensorflow.org/`, 2000–2004.

[5] Hang Chu, Raquel Urtasun, and Sanja Fidler. Song from PI: A musically plausible network for pop music generation. *arXiv preprint arXiv:1611.03477*, 2016.

[6] Michael Scott Cuthbert and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. 2010.

[7] Kemal Ebcioğlu. An expert system for harmonizing four-part chorales. *Computer Music Journal*, 12(3):43–51, 1988.

[8] Manfred Eppe, Roberto Confalonieri, Ewen Maclean, Maximos Kaliakatsos, Emilios Cambouropoulos, Marco Schorlemmer, Mihai Codescu, and K Kühnberger. Computational invention of cadences and chord progressions by conceptual chord-blending. IJCAI'15 Proceedings of the 24th International Conference on Artificial Intelligence, 2015.

[9] Gaëtan Hadjeres and François Pachet. DeepBach: a steerable model for Bach chorales generation. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[10] Gaëtan Hadjeres, Jason Sakellariou, and François Pachet. Style imitation and chord invention in polyphonic music with exponential families. *arXiv preprint arXiv:1609.05152*, 2016.

[11] Hermann Hild, Johannes Feulner, and Wolfram Menzel. Harmonet: A neural net for harmonizing chorales in the style of js bach. In *Advances in neural information processing systems*, pages 267–274, 1992.

[12] Lejaren Arthur Hiller and Leonard M Isaacson. *Experimental Music; Composition with an electronic computer*. Greenwood Publishing Group Inc., 1979.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. *arXiv preprint arXiv:1903.07227*, 2019.

[15] Maximos Kaliakatsos-Papakostas and Emilios Cambouropoulos. Probabilistic harmonization with fixed intermediate chord constraints. In *ICMC*, 2014.

[16] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Feynman Liang, Mark Gotham, Matthew Johnson, and Jamie Shotton. BachBot: Automatic composition in the style of bach chorales. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, 2017.

[18] Feynman T Liang, Mark Gotham, Matthew Johnson, and Jamie Shotton. Automatic stylistic composition of bach chorales with deep lstm. In *ISMIR*, pages 449–456, 2017.

[19] Dimos Makris, Maximos Kaliakatsos-Papakostas, Ioannis Karydis, and Katia Lida Kermanidis. Combining LSTM and feed forward neural networks for conditional rhythm composition. In *International Conference on Engineering Applications of Neural Networks*, pages 570–582. Springer, 2017.

[20] Michael C Mozer. A focused back-propagation algorithm for temporal pattern recognition. *Complex systems*, 3(4):349–381, 1989.

[21] Alexandre Papadopoulos, Pierre Roy, and François Pachet. Assisted lead sheet composition using FlowComposer. In *International Conference on Principles and Practice of Constraint Programming*, pages 769–785. Springer, 2016.

[22] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[23] Donya Quick. *Kulitta: A framework for automated music composition*. Yale University, 2014.

[24] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[25] Bob Sturm, Joao Felipe Santos, and Iryna Korshunova. Folk music style modelling by recurrent neural networks with long short term memory units. In *16th International Society for Music Information Retrieval Conference*, 2015.

[26] Raymond P Whorley, Geraint A Wiggins, Christophe Rhodes, and Marcus T Pearce. Multiple viewpoint systems: Time complexity and the construction of domains for complex musical viewpoints in the harmonization problem. *Journal of New Music Research*, 42(3):237–266, 2013.

[27] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, 2017.

CAN WE LISTEN TO IT TOGETHER?: FACTORS INFLUENCING RECEPTION OF MUSIC RECOMMENDATIONS AND POST-RECOMMENDATION BEHAVIOR

Jin Ha Lee
University of Washington
jinhalee@uw.edu

Liz Pritchard
University of Washington
thelizpritchard@gmail.com

Chris Hubbles
University of Washington
chubbles@uw.edu

ABSTRACT

Few prior studies on music recommendations investigate the context in which users receive the recommendations, and what impact the recommendation has on the user. In this paper, we aim to better understand the factors that affect people's decisions as to whether they choose to listen to music recommendations and how the recommendations impact their music-listening behaviors. We conducted an online survey asking about people's past experiences on giving and receiving music recommendations. We found that in addition to the aesthetic qualities of music and the respondent's taste, expectations regarding the delivery (e.g., timing, persistence) of the recommendations, familiarity, trust in the recommender's abilities, and the rationale for suggestions were important factors. We discuss the implications for the design of music recommenders based on the findings, including better rationale for and accessibility of recommended music, improved saving options, and more targeted delivery at specific times. The data also suggests disparities in how people wish to receive music recommendations and what will influence them to listen to recommendations, versus how they would like to offer recommendations to others. In addition, the findings highlight the importance of music recommendations in people's existing social relationships and their role in building/improving new relationships.

1. INTRODUCTION

Music recommendation has been a well explored topic in the field of music information retrieval over the past few decades. Much of the recent research related to music recommendation focuses on improving recommendations for individual users or user groups by using various data or methods; for instance, user characteristics [22], tags or metadata [20, 21], or collaborative filtering [24]. In addition to more traditional content-based approaches, user behavior [6] and social/contextual features [20, 22] have also been explored to improve recommendation results.

However, few studies explore the broader process of users receiving music recommendations and what happens after the recommendations have been made. What kinds of contextual factors affect people to choose whether they listen or not listen to the music recommendations? Are there any changes that could be made in the way that people or music recommendation systems make the suggestions to improve the likelihood of someone listening to them? What kind of impact do music recommendations have on the user and the social relationships of recommenders and recommendees?

This paper aims to gain a deeper understanding of the user context where music recommendations happen, and the interaction between music recommendations and underlying social relations. We address the following research questions in this study:

RQ1: When people do not listen to recommendations, what are the reasons they do not do so?
RQ2: What can be done to improve the chances that people will listen to recommendations?
RQ3: What happens after the music recommendations? What are the perceived impacts of music recommendations on people's music listening behavior or social life?

We conclude the study by presenting a set of implications for designing music recommendation systems based on what we learned about people's post-recommendation behavior.

2. LITERATURE REVIEW

2.1 Music Recommendations

There is little literature about what happens after a user receives a recommendation, and how the chances of a user listening to the recommendation might be increased. Prior research seeks to understand the motivations behind sharing [12], but it has not necessarily examined *what comes next*, with regards to the recommendation's impact on a user's music listening behavior, social practices, and other aspects of their lives.

As commercial music streaming services become the primary way that many people access music, machine recommendation systems have become an important way to help music listeners find what they want to hear [19]. Jun et al. [11] proposed that there are two primary issues with providing "efficient music recommendation" (p. 1934). These issues are how accurately a recommender system can predict user preference and how accurately a system can assist with searching for new music [11]. The researchers identified that the flow, or sequence, of songs provided

by recommender systems could be improved and suggested blending related recommendations into one seamless clip that takes a user's temporal-spatial information into account [12].

Studies like Lee et al.'s 2011 paper [12], one of the first studies on music sharing behavior via social networks, discusses the motivations behind why users share music. They examined social music practices on Korean social networks like Cyworld and Tisory, finding that "self-expression, ingratiation, altruism, and interactivity" are the main "social motivation factors" driving sharing behaviors on social media platforms (p. 716). Previously, most studies about online sharing music focused on piracy and the motivations behind those behaviors (p. 717). Understanding users' motivations for sharing music and the connections they make to their music may help in improving the likelihood that a recommendation is heard.

Su et al. [19] set out to examine the reliability of collaborative filtering recommender systems. They proposed a system called Recommendation by Tag-driven Item Similarity (RTIS), which takes both "play counts as implicit ratings and item tags as semantic preferences" into account (p. 304). Wang et al. [23] examined the effectiveness of a sequence-based recommender system that takes contextual information into account when providing recommendations, as "people usually have different [music] preferences and requirements under different contexts" (p. 231).

Zhang et al. [25] posit that music recommendation systems should, to the extent possible, simulate the kind of music recommendation that a friend might provide. Listeners are more likely to listen to recommendations from sources they trust, like friends. They built a recommendation framework called Auralist that used four identified factors relating to successful music recommendations: accuracy, diversity, novelty and serendipity. It was found that increasing serendipity in machine-based music recommendation improves user reception of recommendations.

In order for a music recommendation framework to be successful, Schedl [16] believes that recommendation systems must work at three levels: music content, music context, and user context (p. 1). Schedl's study focuses primarily on user-centric models in MIR and uses of geospatial location data for music recommendation. Based on their findings, they present an adaptable mobile player that automatically adjusts the playlist given user context.

2.2 Impact of Music Recommendations on Users

Music can have a memorable, sometimes lasting, impact on one's everyday life. Leong and Wright [14] examined social music practices in the home and the impact that various music technologies have on "people's sociality and in turn how various social practices affect people's interactions with technology" (p. 951). They found that participants that "explore and discover music together… provided opportunities for bonding, with new discoveries and insights into their shared interests in music" (p. 955).

Boer et al. [3] found that one's music preferences can help facilitate social bonding between strangers. Selfhout et al. [17] found that social music rituals and shared music preferences can contribute to adolescents' development of friendships. Boer and Abubakar [2] published a study to expand the evidence of the positive effects social music activities can have on "social cohesion and emotional well-being" (p. 1). They examined music listening behaviors in families and peer groups in four countries: Philippines, Kenya, New Zealand, and Germany. They found that "across four cultures music listening in families and peer groups contributes to family and peer cohesion, respectively" (p. 10).

North, Hargreaves, and Hargreaves' 2004 study [15] of everyday listening among 346 people provided "initial normative data on who people listen with, what they listen to (and what their emotional responses to this music are), when they listen, where they listen, and why they listen" (p. 41). Notably, this study indicated that users' music preferences are situationally dependent, and shifted based on where users are and who they are with. It also highlighted that music was most often accessed during activities independent of deliberate music listening, which we define in our study as passive listening.

Our study seeks to build upon the research of Lee and Price [13] on user personalities and personality characteristics and their relation to music information systems. Lee and Price identified seven personas that exemplified specific user music-listening attitudes, behaviors, and traits. User personas were indicative of how a user might access or curate music, as well as how that user might react to a music recommendation. In this study, we will specifically explore user behavior and how different personas may manifest in the context of music recommendation.

3. STUDY DESIGN AND METHOD

In order to understand user attitudes and behavior post-music recommendation, we designed a web-based survey. Links to the survey were shared via flyer in St. Petersburg, FL and Seattle, WA, posted online across various social media platforms, and disseminated in-person at local band events in St. Petersburg, FL. We received 219 total responses, with 92% of respondents from the United States, 2% from Canada, and 6% from other countries. 42% of respondents were male and 53% were female, with the remaining 6% identifying as other genders or preferring not to answer. The average respondent was 28 years old, with a youngest age of 15 and an oldest of 70 years old (Median: 27; Std dev: 9.71). We asked participants when and for how long they listened to music in their daily life. Respondents answered that they actively listen to music for 1-2 hours (Mean: 1.44 hours; Median: 1 hour; Std dev: 2.01) and passively listen to music for 3-4.5 (Mean: 4.43 hours; Median: 3 hours, Std dev: 3.69) a day. 95% of respondents typically listened to music through a streaming service like YouTube (72%), Spotify (64%), or Pandora (27%). Participants listened to music, actively or

passively, during a wide range of activities. 89% of respondents listened to music while driving or commuting, 77% while working or studying, 74% while exercising, 73% while cleaning, and 63% while cooking.

We also asked respondents open-ended questions about what happens when they receive or give recommendations. Respondent answers were subject to multiple-coder grounded theory analysis [5]. Researchers looked for patterns in the answers given by respondents, and proposed qualitative codes to describe those patterns. The researchers generated a codebook for each question, though some questions were similar enough that certain codes could be used across multiple questions. The definition of each code and the rules for its usage were refined through an iterative process. Once codebooks were finalized, the codes were applied to responses using a consensus code model [9, 10]. Researchers coded answers independently, then met and discussed the applicability of codes until there was agreement between the three researchers for each code usage. The number of codes assigned to a response was not limited; some answers could be captured by a single code, while others were complex enough to require up to half a dozen codes. A copy of the codebook can be accessed at: https://tinyurl.com/ISMIR2019LeePritchardHubbles.

4. RESULTS AND DISCUSSION

4.1 Reasons for Not Listening to Recommendations

We asked respondents what influenced their decisions when they decided not to listen to a recommendation. Responses varied significantly depending on whether the recommendation came from an automated service or from another human being. Three general assessment categories were noted: recommendations tended to be rejected because the recommendation was aesthetically displeasing, because the recommender strategy was suboptimal, or because of external factors out of the control of the recommender.

For automated recommender services, respondents most often tended to judge whether or not to take recommendations based on aesthetic factors such as personal taste (71 respondents, 32.42%) or their level of familiarity with the artist, song, or genre (69, 31.51%). Respondents often rejected recommendations or suggested playlists that include artists/songs they know they hate. Other aesthetic factors included deciding against the recommendation based on descriptive information such as song title, lyrics, or style keywords (22, 10.05%). A few respondents also judged based on the artist's general popularity or based on visual cues like album art or band photography. Some commented on also relying on reviews or their perception of the artist (e.g., "*If I have heard bad reviews from peers or online, or think the band is against my values or promotes things I'm particularly against.*" (P20)). Several respondents stated that they will make a quick judgment as to whether they will continue to listen to the song or not after listening to a short snippet of a song, for about 10-20 seconds (20, 9.13%).

"*The beginning of the piece. If it doesn't sound catchy or doesn't have a decent layout of tone and rhythm, I'll skip it. I try to give all music a fair chance but sometimes I only have a certain window of listening time and I want to use it wisely*." (P192)

Factors external to the recommender system also played a substantial role in rejecting suggestions from automated services. Some respondents simply prefer a listening experience that does not involve recommendations (47, 21.46%). Other respondents mentioned not being in the right mood for the recommendation (34, 15.53%), alluding to the situationally-dependent music preferences of users [14]. A few also mentioned not having enough time to investigate recommendations or having inertia or a lack of interest.

"*This is my default state. I have to want to listen to a suggestion which really means that I'm in an exploratory mood*." (P163)

"*My mood mostly. I'm generally resistant to trying new things, but I always want to. Conditions must be perfect*." (P104)

Relatively few respondents criticized the recommendation strategy. Some chose not to listen based on whether the recommender service had given poor suggestions in the past (22, 10.05%), and a few mentioned annoying or inconvenient means of delivering suggestions, problems accessing the recommendations, getting too many recommendations, or simply needing an easy way to remember the suggestions.

"*The frequency of the suggestions making them easy to ignore and pin as spam.*" (P42)

"*I'd love it if I could choose to add recommended music to a, 'listen later,' or, 'recommend to me again later,' list, just with the touch of a button*." (P196)

In some cases, the reasons people do not listen to music had nothing to do with the content of the music, but more with the context of the song or artist. Several users suggested that content-based music recommendations provided by recommender services will inherently be limited in their ability to predict the likelihood of someone listening to the music recommendation.

"*Honestly, a lot of reasons I won't listen to something is outside the sphere of music. How would a music streaming service know I don't want those recs because the fanbase is full of white supremacists or because the singer is a sexual predator?*" (P7)

For recommendations that came from other human beings, the recommendation strategy was the most important consideration in deciding whether or not to listen. The most common human factor was whether the recommender was reliable, or had given good or bad

recommendations in the past (62, 28.31%); this consideration was much more prominent for human recommenders than for automated systems.

"*Whether or not I think they understand the very specific type of music I have asked them to suggest to me. Or if I have not solicited them if I will listen if I generally like the music they listen to and consider them to have 'good tastes'."* (P49)

The underlying social relationship that a respondent had with a human recommender often played an important role in determining what the recomendee did with the music. In some cases, the feelings toward the recommender overrode other factors like the recommendee's taste in assessing the value of the recommendation.

"*It really depends on the person that gives me the recommendation. If it is someone I know I have a similar preference as, then I am definitely going to listen to it. Similarly, if it is someone I am friends with or just in general like, I will listen to it even if I don't know their music preferences that well. I won't listen to a song if I don't like the person or I know they like a kind of music I don't.*" (P38)

"*If I don't like the person I don't listen. If they're a jerk but I know they have good taste I check it out but I don't get back to them. I tend to associate my favorite songs with people I care about who introduced me to them originally*." (P208)

Many respondents also mentioned that they easily forgot recommendations given by other people, and needed a means by which to remember them (34, 15.53%). A few mentioned annoying or inconvenient recommendation delivery tactics, or difficulty accessing the songs.

"*When people recommend music to me, it's also often not convenient. When I'm already using a streaming music, I'm already relying on them to suggest me new music. That's what they are for. However, when a friend recommends me music, it may come at a time when I'm not in the mood to explore but want to listen to some familiar favorites*. (P178)

Aesthetic considerations were also important in evaluating human recommendations. Familiarity (47, 21.46%) and taste (29, 13.24%) factors also played significant roles in deciding to skip recommendations from people, but were less prominent than with machine suggestions. External factors were also less prominent; quite a few respondents said they often did not have time to explore recommendations (24, 10.96%), and some also mentioned not being in the mood, having a lack of interest, or not wanting to take recommendations in general.

4.2 Things That Can Improve the Chance of People Listening to Recommendations

Additionally, we asked respondents what, if anything, could be done differently to make them more likely to listen to recommendations from streaming services or from other people, and responses were not dramatically different between the two. Two important considerations arose. One was the design or delivery strategy of the recommender or recommendation service. The recommender's design (broadly conceived for both services and people - when, where, and how the recommender or service chose to deliver suggestions) was important to respondents (services 36, 16.44%; people 43, 19.63%), as were components like whether information about the artist or song was provided with the recommendation, why the recommender made the suggestion, whether a clip was available for listening, and whether incentives for listening were provided. For recommender systems, the ability to manually change parameters was important, as respondents felt that they could receive more personalized recommendations.

"*I would be more interested if I would understand why those certain songs are being recommended to me (are the recommended songs based on similar tunes or because people who listen to my type of music like those recommended songs?). I would also probably listen to the recommended songs more if the recommendations were personal (such as seeing what types of people are listening to it, where it's being listened, what kinds of playlists it often appears in).*" (P13)

"*Maybe a better attempt at explaining why it was recommended (e.g., same scene, era, lyrical themes, mood, instrumentation, etc. of what I was already listening to)*." (P66)

For human recommenders, how often or how enthusiastically recommenders persisted in pushing recommendations, and whether the respondent's friends or acquaintances also liked the music being recommended, were also important factors that influenced people's decision to listen or not listen to music.

"*They could mention specific aspects of what I'd like. For example, 'I know you love Neko Case. This singer has a similar voice'."* (P4)

"*A good description: the story behind the track creates an emotional connection to it and makes you listen more attentively. A direct link to the track (preview) available from everywhere. Even better if it can be previewed right in the messenger.*" (P120)

The social connection between the recommender and recommendee was mentioned repeatedly as a factor that might encourage to listen to the music: "*Make me like them? Or at least be charismatic enough and not a horrible garbage person.*" (P208); "*Make me better friends with those other people.*" (P26).

Second, respondents mentioned the importance of the content of the recommendations. Whether the recommendations were similar to personal taste - what the respondent likes or listens to - was most important (services 47, 21.46%; people 40, 18.26%). Respondents also mentioned basing recommendations on artist or genre/style similarity - i.e., musical similarity, rather than closeness to what the

recommender likes. Other content factors included whether the person or system had a deep, intimate knowledge of the respondent; problems with older and newer listening desires conflicting (e.g., recommenders making recommendations based on old preferences); a desire for new songs unfamiliar to the recommendee; recommendations based on general popularity (or deliberately avoiding popular songs); and the ability to compartmentalize - to separate out genres or styles and get siloed recommendations for each. Additionally, some users desired contextual information about music and artists that may influence whether they would listen.

"*If they told me WHY they recommended a particular artist, or gave me some kind of cool "family tree" of the piece they're recommending (e.g., it featured a musician I liked).*" (P79)

"*If some of the algorithmic rationale was a bit more transparent in the messaging to the user (e.g. 'You might like this artist because they feature their bassist and you like other bass-forward bands' or 'Here is a collaboration between an artist you like and a different artist from a genre/label you like').*" (P161)

"*Essentially I'd like to feel like I'm geeking out about the music and somehow digging deeper into things (like the feeling of researching things on Wikipedia) rather than automatically thrown into a new radio station or a recommendation with no context.*" (P174)

Some respondents also talked about social features that aggregate people with similar music tastes or leverage the existing social connections among the listeners: "*Maybe tag artists that many of my friends listen to, sort of indirect friend suggestion.*" (P68); "*Aggregate 'listeners like you' - functionality where I can see what others with tastes similar to me like or are listening to.*" (P179).

We additionally asked what could be done to make it more likely that others would listen to music suggestions the respondent gave. A broad spectrum of responses materialized from this question. Common responses included providing a means for the person to listen (42; 19.18%), having similar music preferences (38; 17.35%), having a deep knowledge of the recommendee's personality or music preferences (34; 15.53%), pushing the recommendation hard or ginning up excitement about it (30; 13.70%), talking in person about the recommendation as opposed to via distance communication (29; 13.24%); and giving context or rationale for why the recommendation was made (24; 10.96%). In general, we noticed an asymmetric relationship between how participants felt about music recommendations from other people versus the music recommendations that they were giving to others. Most participants were able to articulate with specificity the different criteria they use to decide *not* to listen to music recommendations given to them, yet they generally exhibited high confidence that their recommendations to others were in fact listened to (further discussed in 4.3).

4.3 Post-Recommendation

We asked respondents about moments where giving or receiving recommendations led them to have more music-related interactions with another person. 102 (46.58%) said conversations followed about the song, artist, lyrics or genre, which often led to the discovery of mutual musical interests (50; 22.83%) and additional sharing of music (49, 22.37%). 41 (18.72%) described the opening up of a bond or deepening of a friendship with the recommendee, and an equal number talked about having shared experiences with the recipient, such as going to a concert, checking out a record store, or listening to music together at home or on trips. Several respondents talked about the depth and lasting impact of this kind of experience.

"*Once I shared a particular song with an acquaintance and we sat in rapt silence as we listened together, and the song ended up sparking one of the best conversations I had in my teens. Even now, whenever I hear even one song from that album, I remember what it was like when it was 'in the air' so to speak.*" (P27)

"*A friend I had spoken to about music prior had talked to me about an album and asked me to come over so that we could both listen to it on their record player. I was so moved listening to the entire album I started crying and talked to my friend about it, thus leading to a really deep meaningful conversation that deepened our relationship and understanding of each other. We eventually became best friends and this person is now really important to me*." (P82)

We also asked respondents whether they had shared a song, artist, or album with someone they knew in the past three months; whether they knew if the person to whom they had provided the recommendation actually listened to it; and how they knew the recommendation had been listened to. 148 (67.58%) of respondents responded both that they had made a recommendation and that the recommendee had listened to it. By far, the most common means of verifying this was through discussion. Of those who said the recommendee had listened, 103 (69.59%) said they had talked with the recommendee about the recommendation in person or via messaging or social media. 43 (29.05%) mentioned playing the song for the recipient or listening to the songs together, and a few mentioned making follow-up inquiries, singing the song for the recipient, or checking in on the recipient's listening or streaming activity within an application.

5. CONCLUSION AND FUTURE WORK

In this study, we investigated the contexts in which music recommendations occur, in order to improve understanding of the impact of music recommendations on people's lives and social relationships (and vice versa). The main design implications for recommendation systems based on our data analysis are as follows:

Needs for providing and receiving recommendations are asymmetric: In general, it seemed as if respondents were more comfortable broadcasting recommendations than receiving them. The systems and strategies that users would like designed for themselves to receive music differed from those they would build to recommend to others. They seemed more willing to be forward and persistent about pushing the recommendations out than they would prefer for recommendations aimed at them.

Music is an important tool for building social relationships: "Companionship (willingness to engage in social aspects of music listening)" [13] continued to be an important aspect related to music recommendations. Servicing people recommendations that come from their friends may help introduce a desired human element into the systems. Facilitating exchange of individual songs between people, and presenting these exchanges explicitly as recommendations, may improve the user experience beyond algorithmic or expert-curated recommendations. Automated suggestions drawn from friends' listening patterns or notifications of friends' activity ('Your friend just listened to: Track X') may not sufficiently substitute for intentional sharing of recommendations. This intention seems to be important, as the importance of a social relationship often overrode factors like musical tastes and preferences. People paid extra attention to music recommendations that came from people they cared about. Sometimes they were willing to listen to music that they personally had no interest in because they perceived it as an opportunity to spark an interesting conversation or have a shared experience and potentially improve their relationship with the recommender.

The co-listening experience was also important to many of our respondents. Co-listening was a factor that Spinelli et al. [18] identified as a significant social music behavior. Hagen & Lüders [7] noted that users on commercial streaming services might choose to follow each other to deepen interpersonal relationships, not necessarily because of "shared music preferences alone" (p. 10). Brown and Sellen [4] also discussed the social aspects of consuming music and how users can form or deepen relationships through listening to music together. Our findings enrich this literature by showing that our respondents viewed co-listening not simply as a way to ensure or verify that the recommendee listens to the recommendation, but also because that is how shared bonding experiences are created [14]; the recommendation could become the foundation or catalyst of the social relationship between recommender and recommendee. In music recommendation systems, perhaps a feature to support co-listening remotely (e.g., 'Your friend X is listening to Y. Would you also like to listen to it together?' and the system informing friend X that another user chose to co-listen to the song), rather than just providing the recommendation, might encourage people to be more willing to listen to the recommended song.

People desire for more personalized and contextualized recommendations: While the underlying social interaction is important, there is also a persistent desire for better algorithmic curation, over and above simple suggestions from friends. While recommendation systems have gotten more sophisticated and individualized over time, a variety of different recommendation requests surfaced - better matched to users' tastes, better matched to specific musical styles, better attuned to popularity and extramusical cultural associations, or compartmentalized based on different listening sessions. This kind of personalization could help meet the needs of the users in the long tail, who have stronger needs and wants regarding music recommendations (those labeled with the "music epicurean" persona by Lee & Price [13]). The context of the recommendation was also important to many respondents; they wanted to understand why the song was recommended to them, see the musical and social connections between the songs and artists, and know which friends were also listening to or interested in the music recommended to them. This desire for more contextual knowledge was a common theme for both machine and human based recommendations. While there is a fair amount of research available to support context-based music recommendation systems [1, 8, 11, 16], few have examined what might happen if users are provided insight into why an algorithm recommended something.

Some people are simply less interested in recommendations: In designing services, it may be valuable to note a significant recalcitrant population that is unreachable for recommendations, either because they do not want new music at all, or because they do not want new music from the service specifically (persona labeled as "Non-believer" in [13]). Part of this population still seems to respond to recommendations provided by people they know, especially if they feel like they can trust them. This trust was based on two factors: positive past recommendation experience, and how well the recommender understood the recommendee's musical taste. Designing a system to incorporate this social aspect of recommendation may help reach out to this reluctant population.

Many contextual factors need to be further investigated to gain a comprehensive understanding of the impact of music recommendations. In our future work, we plan to dive deeper into people's social music behaviors and explore perceptions of the value of specific social features such as collaborative playlists, co-listening, and music recommendations via videos and other audiovisual media.

6. REFERENCES

[1] L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K-H. Lüke, and R. Schwaiger: "InCarMusic: context-aware music recommendations in a car." *Lecture Notes in Business Information Processing*, Vol. 85, pp. 89-100, 2011.

[2] D. Boer and A. Abubakar: "Music Listening in Families and Peer Groups: Benefits for Young People's Social Cohesion and Emotional Well-being across Four Cultures." *Frontiers in Psychology*, Vol. 5, Article 392, 2014.

[3] D. Boer, R. Fischer, M. Strack, M. H. Bond, E. Lo, and J. Lam: "How Shared Preferences in Music Create Bonds between People: Values as the Missing Link," *Personality and Social Psychology Bulletin*, Vol. 37, No. 9, pp. 1159-1171, 2011.

[4] B. Brown and A. Sellen: "Sharing and Listening to Music," In K. O'Hara and B. Brown, ed. *Consuming Music Together*, Springer, Berlin, pp. 37-56, 2006.

[5] J. M. Corbin and A. Strauss: *Basics of qualitative research: Techniques and procedures for developing grounded theory*, Sage, Los Angeles, 2008.

[6] K. Farrahi, M. Schedl, A. Vall, D. Hauger, M. Tkalcic: "Impact of Listening Behavior on Music Recommendation," *Proc. ISMIR*, pp. 483-388, 2014.

[7] A. N. Hagen and M. Lüders: "Social Streaming? Navigating Music as Personal and Social," *Convergence*, Vol. 23, No. 6, pp. 643-659, 2017.

[8] B. Han, S. Rho, S. Jun, et al.: "Music Emotion Classification and Context-based Music Recommendation," *Multimedia Tools and Applications*, Vol. 47, No. 3, pp. 430-460, 2010.

[9] C. Hill, B. Thompson, and E. Williams: "A Guide to Conducting Consensual Qualitative Research," *The Counseling Psychologist*, Vol. 25, No. 4, pp. 517–572, 1997.

[10] C. Hubbles, D. W. McDonald, and J. H. Lee: "F#%@ That Noise: SoundCloud as (A)Social Media?". *Proceedings of the Association for Information Science and Technology*, Vol. 54, pp. 179-188, 2017.

[11] S. Jun, D. Kim, M. Jeon, S. Rho, and E. Hwang: "Social Mix: Automatic Music Recommendation and Mixing Scheme Based on Social Network Analysis," *The Journal of Supercomputing*, Vol. 71, No. 6, pp. 1933–1954, 2015.

[12] D. Lee, J. Y. Park, J. Kim, J. Kim, and J. Moon: "Understanding Music Sharing Behaviour on Social Network Services," *Online Information Review*, Vol. 35, No. 5, pp. 716-733, 2011.

[13] J. H. Lee and R. Price: "Understanding users of commercial music services through personas: Design implications," *Proc. ISMIR*, pp. 476-482, 2015.

[14] T. W. Leong and P. C. Wright: "Revisiting Social Practices Surrounding Music," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 951–960, 2013.

[15] A. C. North, D. J. Hargreaves, and J. J. Hargreaves: "Uses of Music in Everyday Life," *Music Perception: An Interdisciplinary Journal*, Vol. 22, No. 1, pp. 41-77, 2004.

[16] M. Schedl: "Ameliorating Music Recommendation: Integrating Music Content, Music Context, and User Context For Improved Music Retrieval and Recommendation," *Proceedings of the International Conference on Advances in Mobile Computing & Multimedia*, pp. 3-9, 2013.

[17] M. H. W. Selfhout, S. J. T. Branje, T. F. M. Ter Bogt, and W. H. J. Meeus: "The Role of Music Preferences in Early Adolescents' Friendship Formation and Stability," *Journal of Adolescence*, Vol. 32, No. 1, pp. 95-107, 2009.

[18] L. Spinelli, J. Lau, L. Pritchard, and J. H. Lee: "Influences on the Social Practices Surrounding Commercial Music Services: A Model for Rich Interactions," *Proc. ISMIR*, pp. 671-677, 2018.

[19] J-H. Su, W-Y. Chang, and V. S. Tseng: "Personalized Music Recommendation by Mining Social Media Tags," *Procedia Computer Science*, Vol. 22, No. C, pp. 303-312, 2013.

[20] P. Symeonidis, M. M. Ruxanda, A. Nanopoulos, and Y. Manolopoulos: "Ternary Semantic Analysis of Social Tags for Personalized Music Recommendation," *Proc. ISMIR*, pp. 219-224, 2008.

[21] A. Vall, M. Skowron, P. Knees, and M. Schedl: "Improving Music Recommendations with a Weighted Factorization of the Tagging Activity," *Proc. ISMIR*, pp. 65-71, 2015.

[22] G. Vigliensoni and I. Fujinaga: "Automatic Music Recommendation Systems: Do Demographic, Profiling, and Contextual Features Improve Their Performance?" *Proc. ISMIR*, pp. 94-100, 2016.

[23] D. Wang, S. Deng, G. Xu: "Sequence-based Context-aware Music Recommendation," *Information Retrieval Journal*, Vol. 21, No. 2-3, pp. 230-252, 2018.

[24] Z. Xing, X. Wang, and Y. Wang: "Enhancing Collaborative Filtering Music Recommendation by Balancing Exploration and Exploitation," *Proc. ISMIR*, pp. 445-450, 2014.

[25] Y. C. Zhang, D. Ó Séaghdha, D. Quercia, and T. Jambor: "Auralist: Introducing Serendipity into Music Recommendation," *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM '12)*, pp.13-22, 2012.

ADVERSARIAL LEARNING FOR IMPROVED ONSETS AND FRAMES MUSIC TRANSCRIPTION

Jong Wook Kim, Juan Pablo Bello
Music and Audio Research Lab, New York University
{jongwook,jpbello}@nyu.edu

ABSTRACT

Automatic music transcription is considered to be one of the hardest problems in music information retrieval, yet recent deep learning approaches have achieved substantial improvements on transcription performance. These approaches commonly employ supervised learning models that predict various time-frequency representations, by minimizing element-wise losses such as the cross entropy function. However, applying the loss in this manner assumes conditional independence of each label given the input, and thus cannot accurately express inter-label dependencies. To address this issue, we introduce an adversarial training scheme that operates directly on the time-frequency representations and makes the output distribution closer to the ground-truth. Through adversarial learning, we achieve a consistent improvement in both frame-level and note-level metrics over Onsets and Frames, a state-of-the-art music transcription model. Our results show that adversarial learning can significantly reduce the error rate while increasing the confidence of the model estimations. Our approach is generic and applicable to any transcription model based on multi-label predictions, which are very common in music signal analysis.

1. INTRODUCTION

Automatic music transcription (AMT) concerns automated methods for converting acoustic music signals into some form of musical notation [4]. AMT is a multifaceted problem and comprises a number of subtasks, including multi-pitch estimation (MPE), note tracking, instrument recognition, rhythm analysis, score typesetting, etc. MPE predicts a set of concurrent pitches that are present at each instant, and it is closely related to the task of note tracking, which predicts the onset and offset timings of every note in audio. In this paper, we address an issue in the recent approaches for MPE and note tracking, where the probabilistic dependencies between the labels are often overlooked.

A common approach for MPE and note tracking is through the prediction of a two-dimensional representation that is defined along the time and frequency axes and contains the pitch tracks of notes over time. Piano rolls are the most common example of such representations, and deep salience [5] is another example that can contain more granular information on pitch contours. Once such representation is obtained, pitches and notes can be decoded by thresholding [23] or other heuristic methods [17, 25].

To train a model that predicts a two-dimensional target representation $\hat{\mathbf{Y}} \in \mathbb{R}^{P \times T}$ from an input audio representation $\mathbf{X}$, where P is the number of pitch labels and T is the number of time frames, a common approach is to minimize the element-wise sum of a loss function $\mathcal{L}$:

$$\text{minimize } \mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y}) = \sum_{p=1}^{P} \sum_{t=1}^{T} \mathcal{L}(\hat{\mathbf{Y}}_{pt}, \mathbf{Y}_{pt}), \tag{1}$$

where $\mathbf{Y} \in \mathbb{R}^{P \times T}$ is the ground truth. In a probabilistic perspective, we can interpret $\mathcal{L}$ as the negative log-likelihood of the model parameters ϑ of a discriminative model $p_\vartheta(\mathbf{Y}|\mathbf{X})$:

$$p_\vartheta(\mathbf{Y}|\mathbf{X}) = e^{-\mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y})} = \prod_{p=1}^{P} \prod_{t=1}^{T} e^{-\mathcal{L}(\hat{\mathbf{Y}}_{pt}, \mathbf{Y}_{pt})} = \prod_{p=1}^{P} \prod_{t=1}^{T} p_\vartheta(\mathbf{Y}_{pt}|\mathbf{X}) \tag{2}$$

which indicates that each element of the label $\mathbf{Y}$ is conditionally independent with each other given the input $\mathbf{X}$. This encourages the model to predict the average of the posterior, making blurry predictions when the posterior distribution is multimodal, e.g. natural images [10].

Music data is highly contextual and multimodal, and the conditional independence assumption does not hold in general. This is why many computational music analysis models employ a separate post-processing stage after sequence prediction. One approach is to factorize the joint probability using the chain rule and assume the Markov property:

$$p_\vartheta(\mathbf{Y}|\mathbf{X}) \approx \prod_{p=1}^{P} \prod_{t=1}^{T} p_\vartheta(\mathbf{Y}_{pt}|\mathbf{Y}_{\cdot(t-1)}, \mathbf{X}). \tag{3}$$

This corresponds to appending hidden Markov models (HMMs) [36] or recurrent neural networks (RNNs) [17,39] to the transcription model. The Markov assumption is effective for one-dimensional sequence prediction tasks, such as chord estimation [35] and monophonic pitch tracking [32], but when predicting a two-dimensional representation, it still does not address the inter-label dependencies along the frequency axis.

There exist a number of models in the computer vision literature that can express inter-label dependencies in two-dimensional predictions, such as the neural autoregressive distribution estimator (NADE) [27], PixelRNN [43], and PixelCNN [42]. However, apart from a notable exception using a hybrid RNN-NADE approach [7], the effect of learning the joint posterior distribution for polyphonic music transcription has not been well studied.

To this end, we propose a new approach for effectively leveraging inter-label dependencies in polyphonic music transcription. We pose the problem as an image translation task and apply an adversarial loss incurred by a discriminator network attached to the baseline model. We show that our approach can consistently and significantly reduce the transcription errors in *Onsets and Frames* [17], a state-of-the-art music transcription model.

2. BACKGROUND

2.1 Automatic Transcription of Polyphonic Music

Automatic transcription models for polyphonic music can be classified into frame- or note-level approaches. Frame-level transcription is synonymous with multi-pitch estimation (MPE) and operates on tiny temporal slices of audio, or frames, to predict all pitch values present in each frame. Note-level transcription, or note tracking, operates at a higher level, predicting a sequence of note events that contains the pitch, the onset time, and optionally the offset time of each note. Note tracking is typically implemented as a post-processing stage on the output of MPE [3], by connecting and grouping the pitch estimates over time to produce discrete note events. In this sense, we can say that MPE is at the core of polyphonic music transcription.

Two categories of approaches for MPE have been most successful in recent years: matrix factorization and deep learning. Factorization-based models for music transcription [40] use non-negative matrix factorization (NMF) [29] to factorize a time-frequency representation $\mathbf{X} \in \mathbb{R}^{F \times T}$ as a product of a dictionary matrix $\mathbf{D} \in \mathbb{R}^{F \times K}$ and an activation matrix $\mathbf{A} \in \mathbb{R}^{K \times T}$, where K is the number of pitch labels to be transcribed, e.g. 88 keys for piano transcription. This allows for an intuitive interpretation of each matrix, where each column of $\mathbf{D}$ contains a spectral template for a pitch label, and each row of $\mathbf{A}$ contains the activation of the corresponding pitch over time. Various extensions of factorization-based methods have been proposed to leverage sparsity [1], adaptive estimation of harmonic spectra [14, 44], and modeling of attack and decay sounds [2, 12]. In all of these approaches, an iterative gradient-descent algorithm is used to minimize an element-wise divergence function between the matrix factorization $\mathbf{DA}$ and the target matrix $\mathbf{X}$ [13].

Deep learning [28] methods for music transcription are increasingly popular [3], as larger labeled datasets and more powerful hardware become accessible. These approaches use neural networks (NN) to produce music transcriptions from the input audio. An early work [34] used deep belief networks [20] to extract audio features which are subsequently fed to pitch-wise SVM-HMM pairs to predict the target piano rolls. More recent approaches are based on convolutional [5, 23] and/or recurrent neural networks [6, 17, 39], which are also optimized with gradient descent to minimize an element-wise loss of predicting the target time-frequency representations.

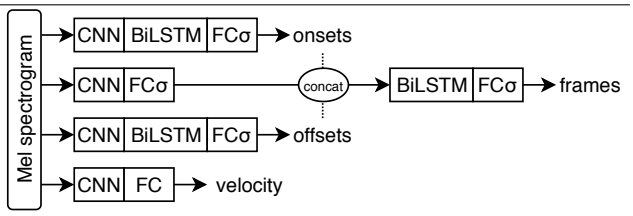


Figure 1. The Onset and Frames model. CNN denotes the convolutional acoustic model taken from [23], FC denotes a fully connected layer, and σ denotes sigmoid activation. Dotted lines mean stop-gradient, i.e. no backpropagation.

Onsets and Frames [17] is a state-of-the-art piano transcription model that we use as our baseline. It uses multiple columns of convolutional and recurrent layers to predict onsets, offsets, velocities, and frame labels from the Mel spectrogram input, as shown in Figure 1. Predicted onset and frame posteriors are then used for decoding the note sequences, where a threshold value is used to create binary onset and frame activations, and frame activations without the corresponding onsets are disregarded.

As discussed above, most NMF- and NN-based methods, including Onsets and Frames, use an element-wise optimization objective which does not consider the inter-label dependencies. This motivates the adversarial training scheme that is outlined in the following subsection.

2.2 Generative Adversarial Networks and `pix2pix`

Generative adversarial networks (GANs) [16] refer to a family of deep generative models which consist of two components, namely the generator G and the discriminator D. Given a data distribution $\mathbf{x} \sim p(\mathbf{x})$ and latent codes $\mathbf{z} \sim p(\mathbf{z})$, GAN performs the following minimax game:

$$\min_G \max_D \underbrace{\left[\mathbb{E}_{\mathbf{x}} \log D(\mathbf{x}) + \mathbb{E}_{\mathbf{z}} \log(1 - D(G(\mathbf{z})))\right]}_{\mathcal{L}_{\text{GAN}}(G,D)}. \quad (4)$$

G and D are implemented as neural networks trained in an adversarial manner, where the discriminator learns to distinguish the generated samples from the real data, while the generator learns to produce realistic samples to fool the discriminator. GANs are most renowned for their ability to produce photorealistic images [22] and have shown promising results on music generation as well [9, 11, 45]. We refer the readers to [8, 15] for a comprehensive review of the techniques, variants, and applications of GANs.

The second term in Equation 4 has near-zero gradients when $D(G(\mathbf{z})) \approx 0$, which is usually the case in early training. To avoid this, a non-saturating variant of GAN is suggested in [16] where the generator is trained with the following optimization objective instead:

$$\max_G \mathbb{E}_{\mathbf{z}} \log D(G(\mathbf{z})). \quad (5)$$

The non-saturating GAN loss is used more often than the minimax loss in Equation 4 and is implemented by flipping the labels of fake data while using the same loss function. Least-squares GAN [31] is an alternative method to address the vanishing gradient problem, which replaces the cross entropy loss in Equations 4-5 with squared errors:

$$\begin{aligned} &\min_D \ \Big[\mathbb{E}_{\mathbf{x}}(D(\mathbf{x})-1)^2+\mathbb{E}_{\mathbf{z}}D(G(\mathbf{x}))^2\Big], \\ &\min_G \ \mathbb{E}_{\mathbf{z}}(D(G(\mathbf{z}))-1)^2. \end{aligned} \tag{6}$$

While the default formulation of GAN concerns unconditional generation of samples from $p(\mathbf{x})$, conditional GANs (cGAN) [33] produce samples from a conditional distribution $p(\mathbf{y}|\mathbf{x})$. To do this, the generator and the discriminator are defined in terms of the condition variable $\mathbf{x}$ as well:

$$\min_G \max_D \underbrace{\Big[\mathbb{E}_{\mathbf{x},\mathbf{y}}\log D(\mathbf{x},\mathbf{y})+\mathbb{E}_{\mathbf{x},\mathbf{z}}\log(1-D(\mathbf{x},G(\mathbf{x},\mathbf{z}))\Big]}_{\mathcal{L}_{\text{cGAN}}(G,D)}. \tag{7}$$

`pix2pix` [21] is an image translation model that learns a mapping between two distinct domains of images, such as aerial photos and maps. A `pix2pix` model takes paired images $(\mathbf{x},\mathbf{y})$ as its training data and minimizes the conditional GAN loss along with an additional L1 loss:

$$\mathbb{E}_{\mathbf{x},\mathbf{y},\mathbf{z}} \left\| \mathbf{y}-G(\mathbf{x},\mathbf{z}) \right\|_1, \tag{8}$$

which encourages the conditional generator to learn the forward mapping from $\mathbf{x}$ to $\mathbf{y}$. It can be thought that the GAN loss in Equation 7 is fine-tuning the mapping learned by the L1 loss in Equation 8, resulting in a predictive mapping that better respects the probabilistic dependencies within the labels $\mathbf{y}$.

In this paper, we adapt this approach to music transcription tasks and show that we can indeed improve the performance by introducing an adversarial loss to an existing music transcription model.

3. METHOD

We describe a general method for improving an NN-based transcription model G that performs prediction of a two-dimensional target $\mathbf{Y}$ from an input audio representation $\mathbf{X}$. Say the original model G is trained by minimizing the loss $\mathcal{L}_{\text{task}}(G(\mathbf{X}),\mathbf{Y})$ between the predicted target $\hat{\mathbf{Y}}=G(\mathbf{X})$ and the ground-truth $\mathbf{Y}$. The main idea of our method is to adapt `pix2pix` [21] to this setup, by introducing an adversarial discriminator D during the training process. The adversarial training objective includes the conditional GAN loss $\mathcal{L}_{\text{cGAN}}$ (Equation 7):

$$\min_G \max_D \mathbb{E}_{\mathbf{X},\mathbf{Y}}\Big[\nu\mathcal{L}_{\text{task}}(G(\mathbf{X}),\mathbf{Y})+\mathcal{L}_{\text{cGAN}}(G,D)\Big], \tag{9}$$

where ν is a hyperparameter that controls how much the conditional GAN loss contributes to the gradient steps relative to the discriminative loss $\mathcal{L}_{\text{task}}$. Figure 2 illustrates how the two components are connected in the computation graph and how the loss terms are calculated.

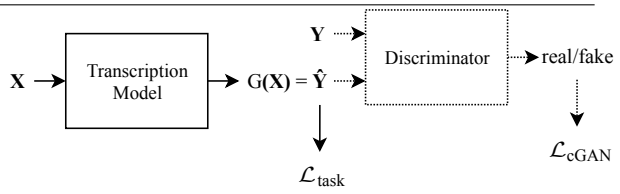


Figure 2. A computation graph showing how a discriminator is appended to the original model. The appended parts are shown as dotted components.

Adversarial training with $\mathcal{L}_{\text{cGAN}}$ allows the model to learn the inter-label dependencies as desired, even when $\mathcal{L}_{\text{task}}$ is defined only in terms of element-wise operations between $\hat{\mathbf{Y}}$ and $\mathbf{Y}$, as in Equation 1. In the next subsection, we describe a neural network architecture for the cGAN discriminator that leverages prior knowledge on music.

3.1 Musically Inspired Adversarial Discriminator

Following `pix2pix`, we use a fully convolutional architecture [30] for the discriminator. By being fully convolutional, the discriminator has translation invariance not only along the time axis (as in HMMs and RNNs) but also along the frequency axis. Since the discriminator determines how realistic a polyphonic note sequence is, the translation invariance enforces that the decision does not depend on the musical key, but only on the relative pitch and time intervals between the notes. This effectively implements a music language model (MLM) [7, 39] and biases the transcription toward more realistic note sequences.

Unlike the image-to-image translation problem, the input representations (e.g. Mel spectrograms) and the output representations (e.g. piano rolls) of a music transcription model can have different dimensions. This makes combining $\mathbf{X}$ and $\mathbf{Y}$ in a fully convolutional manner difficult. For this reason, we make the discriminator a function of $\mathbf{Y}$ only, simplifying the objective in Equation 7 to:

$$\mathcal{L}_{\text{cGAN}}(G,D)=\mathbb{E}_{\mathbf{Y}}\log D(\mathbf{Y})+\mathbb{E}_{\mathbf{X}}\log(1-D(G(\mathbf{X}))). \tag{10}$$

Note that $\mathbf{z}$ is also omitted in Equation 10, as we follow [21] and implement the stochasticity of $\mathbf{z}$ only in terms of dropout layers [41], without explicitly feeding random noises into the generator. This causes a mode collapse problem where the learned $p(\mathbf{Y}|\mathbf{X})$ is not diverse enough, but it does not harm our purpose of producing more realistic target representations.

3.2 TTUR and *mixup* to Stabilize GAN Training

Although an ideal GAN generator can fully reconstruct the data distribution at the global optimum [16], training of GANs in practice is notoriously difficult, especially for high-dimensional data [15]. This led to the inventions of a plethora of techniques for stabilizing GAN training, among which we employ the two-timescale update rule (TTUR) [19] and *mixup* [47]. TTUR means simply setting the generator's learning rate a few times larger than that of the discriminator, which has been empirically shown to stabilize GAN training significantly.

The other technique, *mixup*, is an extension to empirical risk minimization where training data samples are drawn from convex interpolations between pairs of empirical data samples. For a pair of feature-target tuples $(\mathbf{X}_i, \mathbf{Y}_i)$ and $(\mathbf{X}_j, \mathbf{Y}_j)$ sampled randomly from the empirical distribution, their convex interpolation is given by:

$$\begin{aligned}\tilde{\mathbf{X}} &= \lambda \mathbf{X}_i + (1-\lambda)\mathbf{X}_j \\ \tilde{\mathbf{Y}} &= \lambda \mathbf{Y}_i + (1-\lambda)\mathbf{Y}_j\end{aligned} \tag{11}$$

where $\lambda \sim \text{Beta}(\alpha, \alpha)$, and α is the *mixup* hyperparameter which controls the strength of interpolation. When $\alpha = 0$, the Beta distribution becomes Bernoulli(0.5) which recovers the usual GAN training without *mixup*.

mixup is readily applicable to the binary classification task of GAN discriminators. In our conditional GAN setup, we have an additional advantage of having paired samples of a real label $\mathbf{Y}$ and a fake label $\hat{\mathbf{Y}} = G(\mathbf{X})$, which allow us to replace Equation 10 with:

$$\min_G \max_D \mathbb{E}_{\mathbf{X},\mathbf{Y},\lambda}\Big[-\ell(D(\lambda \mathbf{Y} + (1-\lambda)G(\mathbf{X})), \lambda)\Big]. \tag{12}$$

where $\ell(p, y) = -y \log p - (1-y)\log(1-p)$ is the binary cross entropy (BCE) function. With this *mixup* setup, the discriminator now has to operate on the convex interpolation between the predicted representation and the corresponding ground truth. This makes the discriminator's task even more difficult when the prediction gets close to the ground truth, which is desirable because the discriminantor should be inconclusive (i.e. $D = \frac{1}{2}$ everywhere) at the global optimum [16].

Algorithm 1 details the procedure of training the conditional GAN using *mixup*, based on Equations 10 and 12. Note that for training the generator network, we perform label flipping in $\mathcal{L}^G_{\text{cGAN}}$ similarly as in Equation 5. Also, to train a least-squares GAN (Equation 6) instead, we can simply replace ℓ with a mean squared error (MSE) loss.

Input: Generator $G_\vartheta(\mathbf{X})$ with initial parameters ϑ, learning rate η, and loss function $\mathcal{L}_{\text{task}}(\hat{\mathbf{Y}}, \mathbf{Y})$, discriminator $D_\varphi(\mathbf{Y})$ with initial parameters φ, learning rate β, and loss function $\ell \in \{\text{BCE}, \text{MSE}\}$, batch size m, training data distribution $p(\mathbf{X}, \mathbf{Y})$, `pix2pix` weight ν, *mixup* strength α.

Output: Trained conditional generator $G_\vartheta(\mathbf{X})$.

while *φ and ϑ have not converged* **do**
- $\{(\mathbf{X}_i, \mathbf{Y}_i)\}_{i=1,\cdots,m} \leftarrow m$ samples from $p(\mathbf{X}, \mathbf{Y})$
- **for** $i = 1, \cdots, m$ **do**
  - $\hat{\mathbf{Y}}_i \leftarrow G_\vartheta(\mathbf{X}_i)$
  - $\lambda_i \leftarrow$ sample from $\text{Beta}(\alpha, \alpha)$
  - $\tilde{\mathbf{Y}}_i \leftarrow \lambda_i \mathbf{Y}_i + (1-\lambda_i)\hat{\mathbf{Y}}_i$
- **end**
- $\mathcal{L}^D_{\text{cGAN}} \leftarrow \sum_{i=1}^M \ell(D_\varphi(\tilde{\mathbf{Y}}_i), \lambda_i)$
- $\varphi \leftarrow \varphi - \beta \cdot \nabla_\varphi \mathcal{L}^D_{\text{cGAN}}$
- $\mathcal{L}^G_{\text{cGAN}} \leftarrow \sum_{i=1}^M \ell(D_\varphi(\tilde{\mathbf{Y}}_i), 1-\lambda_i)$
- $\vartheta \leftarrow \vartheta - \eta \cdot \nabla_\vartheta \Big[\sum_{i=1}^m \nu \mathcal{L}_{\text{task}}(\hat{\mathbf{Y}}_i, Y_i) - \mathcal{L}^G_{\text{cGAN}}\Big]$

end

Algorithm 1: Training of a *mixup* Conditional GAN.

4. EXPERIMENTAL SETUP

To verify the effectiveness of our approach, we compare Onsets and Frames [17], a state-of-the-art piano transcription model, with variants of the same model that are trained with the adversarial loss. We also aim to evaluate the choices of the GAN loss and the *mixup* strength α.

4.1 Model Architecture

We use the extended Onsets and Frames model [18] which increased the CNN channels to 48/48/96, the LSTM units to 256, and the FC units to 768. The extended model has total 26.5 million parameters. We do not use the frame loss weights described in [17] in favor of the offset stack introduced in the extended version (see Figure 1). During inference, we first calculate the posteriors corresponding to overlapping chunks of audio, with the same length as the training sequences, and perform overlap-add using Hamming windows to obtain the full-length posterior. This is because the effects of adversarial learning do not continue further than the training sequence length when we let the recurrent networks continue to predict longer sequences.

The input to the discriminator has two channels for the onset and frame predictions. The discriminator has 5 convolutional layers: `c32k3s2p1`, `c64k3s2p1`, `c128k3s2p1`, `c256k3s2p1`, `c1k5s1p2`, where the numbers indicate the number of output channels, the kernel size, the stride amount, and the padding size. At each non-final layer, dropout of probability 0.5 and leaky ReLU activation with negative slope 0.2 are used. The mean of the final layer output along the time and frequency axes is taken as the discriminator output.

4.2 Hyperparameters

Table 1 summarizes the hyperparameters used during the experiments, which are mostly taken directly from [17] and [21]. Also following [17], we use Adam [26] and apply learning rate decay of factor 0.98 in every 10,000 iterations, for both the generator and the discriminator. We examine two types of GAN losses, the non-saturating GAN (ℓ = BCE) and the least-squares GAN (ℓ = MSE). For each GAN loss, multiple values of *mixup* strengths are compared with $\alpha = 0$, i.e. no *mixup*. Training runs for one million iterations, and the iteration that best performs on the validation set are used for evaluation on the test set.

Hyperparameter	Values
Generator learning rate η	0.0006
Discriminator learning rate β	0.0001
Discriminator loss function ℓ	{BCE, MSE}
Batch size m	8
`pix2pix` weight ν	100
mixup strength α	{0, 0.2, 0.3, 0.4}
Activation threshold τ	0.5
Training sequence length	327,680

Table 1. Hyperparameters used during the experiments.

4.3 Dataset

We use the MAESTRO dataset [18], which contains Disklavier recordings of 1,184 classical piano performances. The dataset consists of 172.3 hours of audio, which are provided with 140.1, 15.3, and 16.9 hours of train/validation/test splits such that recordings of one composition only appear in the same split. We resample the audio to 16 kHz and down-mix into a single channel. Following [17], an STFT window of 2,048 samples is used for producing 229-bin Mel spectrograms, and a hop length of 32 ms is used. Training sequences sliced at random positions are used, unlike the official implementation which slices training sequences at silence or zero crossings.

4.4 Evaluation Metrics

The Onsets and Frames model perform both frame-level and note-level predictions, and their performance can be evaluated with the standard precision, recall, and F1 metrics. For multi-pitch estimation, we also report the error rate metrics defined in [36], which include total error, substitution error, miss error, and false alarm error. We use the `mir_eval` [37] library for all metric calculations. For the note-level metrics, we use the default settings of the library, which use 50 ms for the onset tolerance, 50 ms or 20% of the note length (whichever is longer) for the offset tolerance, and 0.1 for the velocity tolerance.

5. RESULTS

5.1 Comparison with the Baseline Metrics

Table 2 and 3 summarize the transcription performance, clearly showing a consistent improvement in the conditional GAN models over the Onsets and Frames baseline. Table 2 shows that both non-saturating GAN and least-squares GAN achieve the highest frame and note F1 scores when the *mixup* strength $\alpha = 0.3$ is used, and they both outperform the baseline. The binary piano rolls are easy to distinguish from the non-binary predictions, which may cause imbalanced adversarial training. *mixup* allows non-binary piano rolls to be fed to the discriminator, making its task more challenging and leading to higher performance.

Table 3 shows an important trend of the cGAN results compared to the baseline that cGAN trades off a bit of precision for a significant improvement in recall; this is a side effect of the cGAN producing more confident predictions, as will be discussed in the following subsections.

	Baseline	GAN type	*mixup* strength α 0	0.2	0.3	0.4
Frame F1	0.899	Non-Saturating	0.664	0.912	**0.914**	0.907
		Least-Squares	0.904	0.903	0.906	0.898
Note F1	0.942	Non-Saturating	0.717	0.953	**0.956**	0.951
		Least-Squares	0.944	0.947	0.950	0.943

Table 2. Frame and note F1 scores are the highest when the non-saturating GAN loss and $\alpha = 0.3$ are used.

While the percentage differences are moderate, our method achieves statistically significant improvements in F1 metrics on the MAESTRO test dataset ($p < 10^{-14}$ for all 4 metrics, two-tailed paired t-test). The distribution of per-track improvement in each F1 metric is shown in Figure 4, which indicates that the improvements are evenly distributed across the majority of the tracks. These improvements are especially promising, considering that Onsets and Frames is already a very strong baseline.

5.2 Visualization of Frame Activations

To better understand the inner workings of the conditional GAN framework, we visualize the frame posteriorgrams created by the baseline and the best performing conditional GAN model in Figure 3. In contrast to the baseline posteriorgrams which have many blurry segments, the posteriorgrams generated by our method mostly contain segments with solid colors, meaning that the model is more confident in its prediction. Figure 5 shows that the proportion of frame activation values in $(0.1, 0.9)$ is noticeably higher in the baseline, thus making the output less sensitive to the threshold choice. This is because indecisive predictions are penalized by the discriminator, since they are easy to distinguish from the ground-truth which contains only binary labels. The generator is therefore encouraged to output the most probable note sequences even when it is unsure, rather than producing blurry posteriorgrams that might hamper the decoding process. This allows for an interpretation in which the GAN loss provides a prior for valid onset and frame activations, and the model learns to perform MAP estimation based on this prior.

5.3 Training Dynamics and The Generalization Gap

Figure 6 shows the learning curves for the frame F1 and note F1 scores, where the scores on the training dataset are plotted in dotted lines. It is noticeable in the figure

	mixup strength α	Frame Metrics							Note Metrics			Note Metrics with Offsets			Note Metrics with Offsets & Velocity		
		F1	P	R	E_{total}	E_{subs}	E_{miss}	E_{fa}	F1	P	R	F1	P	R	F1	P	R
Baseline		.899	.946	.857	.179	.013	.130	.036	.942	.990	.899	.802	.842	.765	.790	.830	.755
Non-Saturating GAN	0.3	**.914**	.931	.898	**.156**	.012	.089	.054	**.956**	.981	.932	**.813**	.835	.793	**.802**	.823	.782
Least-Squares GAN	0.3	.906	.942	.875	.167	.013	.113	.042	.950	.988	.916	.810	.841	.781	.799	.830	.771

Table 3. Summary of transcription performance using *mixup* strength $\alpha = 0.3$. The non-saturating GAN loss has the highest performance across all F1 metrics. The average metrics across the tracks in the MAESTRO test dataset are reported, and the model checkpoint where the average of frame F1 and note F1 is the highest on the validation dataset is used.

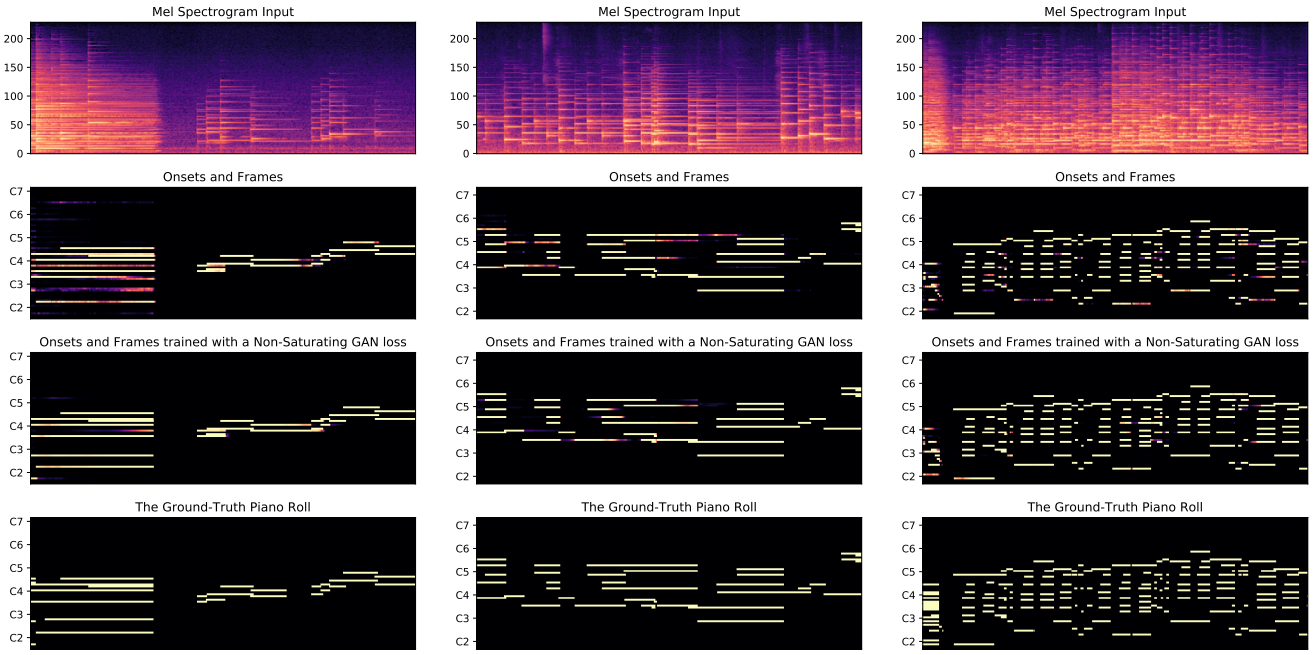


Figure 3. Comparisons of the frame activation posterior predicted by the baseline and our model (ℓ = BCE, $\alpha = 0.3$), on three example segments. The input Mel spectrograms and the target piano rolls are shown together. The GAN version produces more confident predictions compared to the noisy baselines, leading to more accurate predictions.

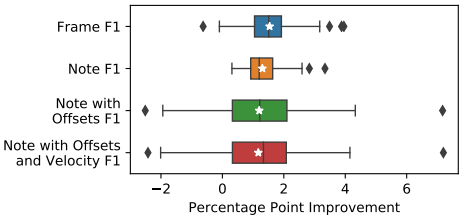


Figure 4. F1 score improvements over the baseline, tested on the MAESTRO test tracks.

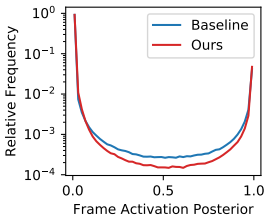


Figure 5. Distribution of frame activation values.

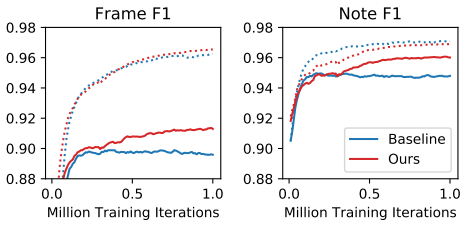


Figure 6. Learning curves showing the generalization gaps; training curves are dotted.

that the validation F1 scores for the baseline stagnate after 300k iterations, while the F1 scores of our model steadily grow until the end of 1 million iterations. Thanks to this, the generalization gap — the difference between the training and validation F1 scores — is significantly smaller for the conditional GAN model. This means that the GAN loss works as an effective regularizer that encourages the trained model to generalize better to unseen data, rather than memorizing the note sequences in the training dataset as LSTMs are known to be capable of [46].

6. CONCLUSIONS

We have presented an adversarial training method that can consistently outperform the baseline Onsets and Frames model, using the standard frame-level and note-level transcription metrics and visualizations that show how the improved model predicts more confident output. To achieve this, a discriminator network is trained competitively with the transcription model, i.e. a conditional generator, so that the discriminator serves as a learned regularizer that provides a prior for realistic note sequences.

Our results show that modeling the inter-label dependencies in the target distribution is important and brings measurable performance improvements. Our method is generic, and any model that involves predicting two-dimensional representation should be able to benefit from including an adversarial loss. These approaches are common not only in transcription models but also in speech or music synthesis models that predict spectrograms as an intermediate representation [24, 38].

Our results do not include the effects of using data augmentation [18], which is orthogonal to our approach and should bring additional performance improvements when applied. As discussed, the discriminator imposes the prior on the target domain whereas data augmentation enriches the input audio distribution. This implies that our method would be less effective when the majority of errors are due to the discrepancy in the audio distribution between the training and test datasets. How to apply adversarial learning for better generalization on the input distribution is a potential future research direction.

7. REFERENCES

[1] Samer A Abdallah and Mark D Plumbley. Unsupervised analysis of polyphonic music by sparse coding. *IEEE Transactions on Neural Networks*, 17(1):179–196, 2006.

[2] Emmanouil Benetos and Simon Dixon. Multiple-instrument polyphonic music transcription using a temporally constrained shift-invariant model. *The Journal of the Acoustical Society of America*, 133(3):1727–1741, 2013.

[3] Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36(1):20–30, 2019.

[4] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[5] Rachel M Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan Pablo Bello. Deep salience representations for f0 estimation in polyphonic music. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 63–70, 2017.

[6] Sebastian Böck and Markus Schedl. Polyphonic piano note transcription with recurrent neural networks. In *Proceedings of the IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 121–124, 2012.

[7] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012.

[8] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[9] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[10] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016.

[11] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. GANSynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710*, 2019.

[12] Sebastian Ewert and Mark Sandler. Piano transcription in the studio using an extensible alternating directions framework. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):1983–1997, 2016.

[13] Cédric Févotte and Jérôme Idier. Algorithms for non-negative matrix factorization with the β-divergence. *Neural computation*, 23(9):2421–2456, 2011.

[14] Benoit Fuentes, Roland Badeau, and Gaël Richard. Harmonic adaptive latent component analysis of audio and application to music transcription. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(9):1854–1866, 2013.

[15] Ian Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[17] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 50–57, 2018.

[18] Curtis Hawthorne, Andrew Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[19] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[20] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[21] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1125–1134, 2017.

[22] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410.

[23] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 475–481, 2016.

[24] Jong Wook Kim, Rachel Bittner, Aparna Kumar, and Juan Pablo Bello. Neural music synthesis for flexible timbre control. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[25] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello. CREPE: A convolutional representation for pitch estimation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 161–165, 2018.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations, (ICLR)*, 2015.

[27] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 29–37, 2011.

[28] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[29] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, pages 556–562, 2001.

[30] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[31] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

[32] Matthias Mauch and Simon Dixon. pYIN: A fundamental frequency estimator using probabilistic threshold distributions. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 659–663. IEEE, 2014.

[33] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[34] Juhan Nam, Jiquan Ngiam, Honglak Lee, and Malcolm Slaney. A classification-based polyphonic piano transcription approach using learned feature representations. In *Proceedings of the 12th International Society for Music Information Retrieval (ISMIR) Conference*, pages 175–180, 2011.

[35] Yizhao Ni, Matt McVicar, Raul Santos-Rodriguez, and Tijl De Bie. An end-to-end machine learning system for harmonic analysis of music. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1771–1783, 2012.

[36] Graham E Poliner and Daniel PW Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, 2007(1):048317, 2006.

[37] Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel PW Ellis, and C Colin Raffel. mir_eval: A transparent implementation of common MIR metrics. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, 2014.

[38] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. Natural TTS synthesis by conditioning WaveNet on Mel spectrogram predictions. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4779–4783. IEEE, 2018.

[39] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):927–939, 2016.

[40] Paris Smaragdis and Judith C Brown. Non-negative matrix factorization for polyphonic music transcription. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 177–180, 2003.

[41] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[42] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with PixelCNN decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

[43] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1747–1756, 2016.

[44] Emmanuel Vincent, Nancy Bertin, and Roland Badeau. Adaptive harmonic spectral decomposition for multiple pitch estimation. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):528–537, 2010.

[45] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 324–331, 2017.

[46] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[47] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. *mixup*: Beyond empirical risk minimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

AUTOMATIC MUSIC TRANSCRIPTION AND ETHNOMUSICOLOGY: A USER STUDY

Andre Holzapfel
KTH Royal Institute of Technology
holzap@kth.se

Emmanouil Benetos
Queen Mary University of London
emmanouil.benetos@qmul.ac.uk

ABSTRACT

Converting an acoustic music signal into music notation using a computer program has been at the forefront of music information research for several decades, as a task referred to as automatic music transcription (AMT). However, current AMT research is still constrained to system development followed by quantitative evaluations; it is still unclear whether the performance of AMT methods is considered sufficient to be used in the everyday practice of music scholars. In this paper, we propose and carry out a user study on evaluating the usefulness of automatic music transcription in the context of ethnomusicology. As part of the study, we recruited 16 participants who were asked to transcribe short musical excerpts either from scratch or using the output of an AMT system as a basis. We collect and analyze quantitative measures such as transcription time and effort, and a range of qualitative feedback from study participants, which includes user needs, criticisms of AMT technologies, and links between perceptual and quantitative evaluations on AMT outputs. The results show no quantitative advantage of using AMT, but important indications regarding appropriate user groups and evaluation measures are provided.

1. INTRODUCTION

Automatic music transcription (AMT) is the process of transferring an music audio signal to a symbolic representation using computational methods [14, p.30]. Engineering research has been developing AMT methods for a number of decades now (see *e.g.* [17] for an early example), and it represents a recurrent theme in the discourse in the field of music information retrieval (MIR). In the field of comparative musicology – the historical predecessor of ethnomusicology – the idea of using automatic methods to obtain a graphical representation (not necessarily symbolic) from a music recording attracted interest from the early days of audio recording technology [7]. This long history of interest in AMT technology in two rather remote fields motivates us to investigate what the current state of the art in AMT may have to offer for (ethno)musicologists transcribing a piece of music.

In the field of MIR, recent AMT research has mostly focused on automatic transcription of piano recordings in the context of Western/Eurogenetic music (see [3] for a recent overview). The vast majority of proposed methods aim to create systems which can output a MIDI or MIDI-like representation in terms of detected notes with their corresponding onsets/offsets in seconds. Such methods are typically evaluated quantitatively using multi-pitch detection and note tracking metrics also used in the respective MIREX public evaluation tasks [2]. Methods that can automatically convert audio into staff notation include the beat-informed multi-pitch detection system of [8] in the context of folk music (the dataset of this work is also used in the present user study) and the multi-pitch detection and rhythm quantization system of [15], which was applied to Western piano music. Recently, methods inspired by deep learning theory have also attempted to automatically convert audio directly into staff notation [5, 18], although these methods are mostly constrained to synthesized monophonic excerpts using piano soundfonts.

Within ethnomusicology, transcription may take a large variety of forms, depending on analytic goals and the analyzed musical context [20]. An early study of the commonalities and discrepancies between transcriptions of the same piece by several experts was conducted by List in 1974 [12]. The study investigated transcriptions of three pieces by up to eight transcribers, and documented higher consistency in the notation of pitch than in duration. An estimated pitch curve was provided as well, and the study demonstrated that only small corrections were conducted by the transcribers. The value of user studies has been recognized in MIR [9, 10, 19, 21], and MIR user studies have been conducted, for instance, in the context of applying music to achieve certain emotional states [6] and therapeutic applications [11]. However, to the best of our knowledge, since [12] no user studies have been conducted that study a larger group of transcribers in their interaction with the output of an AMT system.

In the context of AMT, the research questions that can be approached by a user study are manifold. In this study, we investigate the relations between the output of a state-of-the-art AMT system and manual transcriptions, the validity of quantitative evaluation metrics when comparing manual transcriptions, and the question of whether using an AMT as a starting point for transcription provides ad-

vantages of any kind. To this end we conducted a study with 16 experienced transcribers, and asked them to transcribe eight excerpts of a particular musical style with a specific analytic goal. Transcriptions were either to be performed completely manually, or using an AMT output as a starting point. Our results document a range of insights into the qualities and problems of AMT and evaluation metrics. Even after decades of development of AMT systems, our study cannot reveal a clear advantage of using AMT to inform manual transcription, but our results indicate promising avenues for future development.

The outline of this paper is as follows. Section 2 presents the method, and Section 3 the results of the study. Sections 4 and 5 provide discussion and conclusions, respectively.

2. PROPOSED STUDY

2.1 Subjects

Participants for the proposed study were recruited from the Institute of Musicology in Vienna, Austria, from SOAS University of London, UK, and from City, University of London, UK. In total, 16 subjects participated in our study, nine male and seven female. The criteria for the participation in the study were being an advanced student or recent graduate in a musicology or ethnomusicology program, having attended training on music transcription / musical dictation and being recommended by a member of faculty as being good transcribers. Apart from these students, two musicology lecturers also participated as subjects.

The participants had 16 years of music training on average, with a standard deviation of 9 years. In terms of their interests, 6 participants closely identified with Western classical music, and 10 participants identified with world/folk/traditional music. In terms of their professional practice, 9 participants engaged with Western classical music, and 8 with world/folk/traditional music. In terms of software for music notation and transcription, 7 participants were familiar with MuseScore, 6 with Transcribe!, 5 with Sibelius, and 2 with Sonic Visualiser.

2.2 Material

For this study, we use audio recordings and corresponding transcriptions collected as part of the Crinnos project [1], which were also used as part of the *Sousta Corpus* for AMT research in [8]. All recordings used in this study were recorded in 2004 in Crete, Greece, and all regard a specific dance called *sousta*. Recordings selected for this study were transcribed by ethnomusicologists in Western staff notation as part of the Crinnos project.

These recordings were chosen for the present study for several reasons. They provide a dataset that is highly consistent in terms of musical style, thus appropriate for an AMT user study consisting of multiple excerpts. The sousta dance is usually notated in 2/4 meter and has a relatively stable tempo, again providing consistency for human transcribers. The instrumental timbres are likewise highly consistent, with one Cretan lyra (a pear-shaped fiddle) playing the main melody, and usually two Cretan lutes playing the accompaniment.

Eight audio excerpts from the *Sousta Corpus* were selected for the present study. The length of each excerpt was set to 4 bars, which results in a duration of 7-8 seconds per excerpt. The number of excerpts and their duration were determined through pilot studies, with the goal to constrain the duration of the proposed study for each participant to 2 hours. The position of the 4 bars within each piece was chosen as such to provide study participants with a complete musical phrase, in order to aid transcription. The corpus of [1] also contains corresponding reference transcriptions in musicXML format, which were used for quantitative evaluations. [1]

We did not assume that participants are familiar with the music culture used in this study. Therefore, one complete recording from the corpus of [8] was also selected in order to familiarize participants with the music culture prior to the start of the study.

2.3 AMT methods

For the purposes of this study, an AMT method is needed that can convert an audio recording into machine-readable Western staff notation, suitable for audio recordings from the particular music culture employed for this study. In terms of academic research, the pool of candidate AMT methods for audio-to-staff music transcription is limited to the beat-informed matrix factorization-based system used for the same corpus in [8], the two-stage piano-specific polyphonic transcription system from [15], plus preliminary works for end-to-end piano-only transcription using synthesized audio [5, 18]. In terms of commercial AMT software, a partial list is included in [3], out of which only a small subset (ScoreCloud, Sibelius' AudioScore plugin) produces transcriptions in staff notation.

We selected the beat-informed matrix factorization-based AMT method [8] from the above list of candidate AMT methods, because of its suitability for the present corpus. In terms of commercial tools, we selected ScoreCloud [2], given its competitive performance in monophonic transcription of violin recordings, an instrument with timbre characteristics similar to the Cretan lyra. Based on quantitative AMT evaluations of both systems (shown in Section 3.1), it was decided to use the ScoreCloud AMT system for the present user study.

Since the objective of this study is for participants to transcribe the main melody and not to focus on accompaniment and ornamentations, the automatic transcriptions produced by the systems of [8] and ScoreCloud were modified as to remove the bass staff (if it exists) along with all transcribed notes in that staff; all ornamentations (e.g. trills, grace notes) and note groupings were deleted. We also changed sharp or flat symbols in the automatic transcriptions as to have consistent accidentals for each excerpt.

[1] The excerpts and reference transcriptions can be obtained at `https://bit.ly/2ZKhmnY`

[2] `https://scorecloud.com/`

2.4 Procedure

Experiments took place in quiet rooms; participants were provided with a laptop (if they did not have their own), headphones, printed or digital automatic transcriptions (as desired by participant), manuscript paper, and the study questionnaire. Participants were video recorded in order to assist with the subsequent annotation process.

Participants were asked to transcribe the main melody for each excerpt and to not transcribe the accompaniment or ornamentations. The purpose of this specification was to clarify the analytic goal of the transcription. Participants were free to use the music notation software of their preference or to transcribe on manuscript paper. The study consisted of 8 excerpts per participant, with 4 excerpts to be manually transcribed, and 4 excerpts to be accompanied with AMT outputs in printed and machine-readable format, to be used as a starting point for transcriptions. The order of manual and edited transcriptions was interleaved, and participants were either asked to start transcribing their first segment manually or to edit an automatic transcription. The order of the 8 excerpts exposed to participants was randomized. Fig. 1 shows an example automatic transcription produced using ScoreCloud, compared with a reference and a study participant transcription.

Following the study, a short conversation with participants took place, in order to obtain qualitative feedback as well as information on their experience with automated tools for the task. All participant transcriptions that were produced on manuscript paper were re-transcribed by the authors in machine-readable music notation using MuseScore, in order to carry out quantitative evaluations.

2.5 Evaluation Metrics

2.5.1 Participant Questionnaire

Participants were asked to quantify their effort for every excerpt towards producing the transcription on a scale 1-10 (1: no effort, 10: very high effort). In addition, for every excerpt to be edited from an automatic transcription, participants were asked to rate the quality of the AMT (on a scale 1-10, with 10 being excellent). After completing the experiment, participants were asked to specify the most crucial mistakes present in the automatic transcriptions, and to comment on the possible value of AMT as a starting point towards producing manual transcriptions.

2.5.2 Quantitative metrics

In order to evaluate the performance of the automatic transcription methods, as well as to compare the participants' transcriptions with the reference transcriptions, we use the quantitative metrics for complete AMT proposed in [15] [3] . We chose to compare with these particular reference transcriptions, because their transcribers had extended experience with both transcription and the musical style.

These quantitative metrics are based on an automatic alignment of the estimated score to the reference score using the method of [16]. Following alignment, we are able to identify correctly detected notes, notes with pitch errors (also called *pitch substitution errors*), extra notes, and missing notes. Based on the above definitions, the following error rates are used, as per [15]: pitch error rate E_{p}, extra note rate E_{e}, missing note rate E_{m}, and onset time error rate E_{on}. We also define an average error metric E_{mean} as the arithmetic mean of all 4 aforementioned metrics. As additional quantitative metric, we also measured the time taken by each participant to transcribe each excerpt.

[3] Noting that typical metrics used in multi-pitch detection and note tracking [2] are not suitable for evaluating transcriptions in staff notation.

3. RESULTS

3.1 AMT system evaluation

Excerpt#	E_{p}	E_{e}	E_{m}	E_{on}	E_{mean}
1	18.75	18.18	15.62	48.15	25.18
2	10.71	3.85	10.71	36	15.32
3	20.69	28.57	31.03	45	31.32
4	10	29.03	26.67	18.18	20.97
5	2.5	38	22.5	25.81	22.20
6	7.69	7.14	0	23.08	9.48
7	13.64	3.12	29.54	61.29	26.90
8	40.91	41.18	9.09	70	40.29
Average	15.61	21.13	18.15	40.93	23.96

Table 1. AMT quantitative evaluation scores using ScoreCloud.

Excerpt#	E_{p}	E_{e}	E_{m}	E_{on}	E_{mean}
1	12.5	66.67	66.67	75	55.21
2	21.05	22.22	26.31	35.71	26.33
3	0	100	100	0	50.00
4	3.70	77.78	77.78	50	52.31
5	17.39	39.13	39.13	21.43	29.27
6	12.5	16.67	16.66	35	20.21
7	33.33	0	25.64	31.03	22.50
8	38.46	53.57	0	53.85	36.47
Average	17.37	47.00	44.02	37.75	36.54

Table 2. AMT quantitative evaluation scores using the method of [8].

Tables 1 and 2 depict the error rates for all eight examples used in the experiments, using ScoreCloud and the AMT system of [8], respectively. The ScoreCloud system performs significantly better than the system of [8] (based on a paired-sample t-test, $p < 0.05$) for the extra- and missing note rates, and for the mean error rate E_{mean}.

Ranking of the examples looks quite inconsistent between ScoreCloud and the method of [8]. Segment 6 has lowest error rates for both, but the highest error rates are for Segment 8 for ScoreCloud, and Segment 1 for [8]. However, when calculating the average E_{mean} of both algorithms, one could identify Segment 3 as the most challenging (2nd highest error rate for both algorithms).

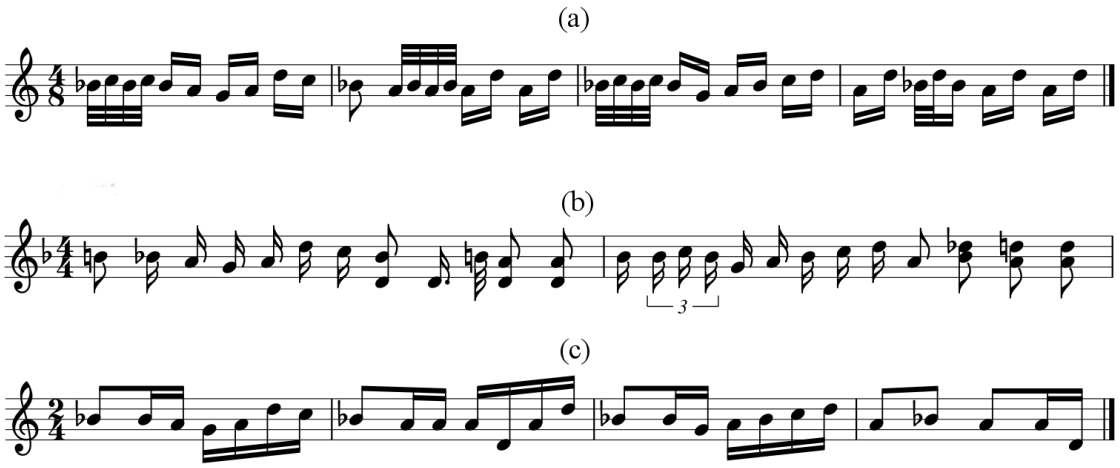


Figure 1. Transcriptions for bars 85-88 of segment 114 from the corpus of [8]. (a) Reference transcription. (b) Automatic transcription using ScoreCloud. (c) Manual transcription created by one of the expert participants.

3.2 Participant transcription evaluation

Table 3 depicts the transcription error rates obtained for each piece, averaged over all participants. The lowest mean error rate E_{mean} is obtained for Excerpt 6, and the highest for Excerpt 3, which is consistent with the ranking obtained from the two automatic transcription algorithms.

In Table 3, error rates with statistically significant differences (based on one-sample t-tests) to the error rates obtained from the ScoreCloud AMT (Table 1) are underlined. In addition, significantly lower error rates are emphasized using bold numbers. For the overall average (last row of Table 3), the only significant differences are increases in error rates (E_e, E_m) for the participant transcriptions, which indicates that participants' transcriptions include a larger number of extra and missing notes compared to the ScoreCloud transcriptions. Regarding significant changes of error rates for the individual pieces, four out of five for E_p, three out of five for E_{on}, and one out of two for E_{mean} are decreases in error rate. This indicates that at least some participant transcriptions were more consistent with the reference regarding meter (E_{on}) and pitch (E_p), compared to the ScoreCloud automatic transcriptions.

The inconsistent tendencies observed for the various metrics depicted in Table 3 motivate to investigate further which of the metrics most accurately reflect the notion of the quality of a transcription. Based on the quality ratings that the participants provided for each AMT, a conclusion was obtained which of the five error rates depicted in Tables 1 to 3 most correlated with the rated quality of a transcription. The highest correlation with the participants' stated AMT quality ratings was obtained for E_{mean} ($r = -0.91, p = 0.0019$). Correlations for E_p and E_{on} were still significant ($p < 0.05$), but correlations with both E_e and E_m were not. This motivates to focus on E_{mean} as the main metric for the rating of transcription quality in this paper, and to rather consider E_e and E_m as indicators for the chosen level of detail in the manual transcription.

Excerpt#	E_{p}	E_{e}	E_{m}	E_{on}	E_{mean}
1	14.58	42.15	24.42	40.85	30.50
2	**4.76**	20.07	15.37	**21.49**	15.42
3	**7.97**	54.05	48.45	45.83	39.07
4	18.16	38.50	35.61	44.13	34.10
5	19.52	33.75	16.52	37.16	26.74
6	**1.75**	18.84	11.50	16.90	12.25
7	17.53	15.33	30.68	**41.20**	26.18
8	**20.25**	37.81	13.97	**47.22**	**26.81**
Average	12.96	32.75	24.47	36.70	26.72

Table 3. Average error rates over all participant transcriptions. Underlined values emphasize statistical significant difference to the value in Table 1 (one-sample t-test, $p < 0.05$); bold values emphasize significant decrease in error rates over AMT.

3.3 Differences depending on the subject

Group	T(s)	Eff.	E_p	E_e	E_m	E_{on}	E_{mean}
Experts	430.33	4.1	**9.71**	31.58	28.60	**28.49**	24.60
Other	803.96	5.4	13.99	**33.12**	23.16	39.29	**27.39**

Table 4. Comparison of mean transcription times, T(s), rated efforts (Eff.), and error rates between experts and other participants. Statistically significant differences between experts and others are underlined (Welch's t-test, $p < 0.05$), and significant differences to average error rates in Table 1 are emphasized using bold numbers.

Based on the participant information, participants were grouped into experts and non-experts. The group of experts comprises three participants, who were either instructors of transcription courses, or had several decades' experience in transcribing folk music. Table 4 depicts the mean transcription times, rated transcription efforts, and error rates obtained from the transcriptions of these two groups. Whereas the expert group's transcription times

and effort ratings were significantly lower, the results regarding the quantitative error rate metrics remain inconclusive. Only regarding the onset error rate (E_{on} - which assesses the metrical correctness of the transcriptions when compared with the reference), the expert group had significantly better values that the non-expert group. In comparison with the average error rates obtained using the ScoreCloud algorithm (Table 1), the other transcribers have significantly higher error rates for E_e and E_{mean}, whereas the experts have significantly lower error rates regarding E_p and E_{on}. This implies that the tendency towards decreased error rates in the latter two metrics observed in Table 3 is more emphasized among the expert group.

3.4 Differences between editing and manual transcription

Case	T(s)	Eff.	E_p	E_e	E_m	E_{on}	E_{mean}
AMT	733.14	4.98	11.91	29.51	21.58	36.02	24.75
Man.	695.44	5.20	14.02	35.99	27.35	37.37	28.68

Table 5. Comparison of mean transcription times, T(s), rated efforts (Eff.), and error rates between AMT editing and manual transcription. None of the differences between the cases were found statistically significant (Student's t-test, all p-values > 0.18).

Table 5 shows the error rates over all participant transcriptions when editing automatic transcriptions as a starting point and when carrying out manual transcriptions, respectively. In order to address the question if any difference in the quality of the obtained participant transcriptions exists between the manual transcriptions and the editing of the automatic ones, the distributions of the error rates from the two cases were compared using two-sample t-tests. However, no significant differences were observed, indicating that the transcription times, efforts, and quality neither improved, nor deteriorated by using automatic transcriptions as a starting point. Significantly decreased variances were observed for two error rates (E_p, E_m) when using the AMT as a starting point (two-sample F-test, $p < 0.05$). This indicates that the usage of AMT as a starting point – at least in our experiments – led to transcriptions that are more similar, which may be interpreted as a bias imposed on the transcribers by using the AMT.

Since we observed in Table 4 that experts transcribe generally faster, we investigated if some gain in using AMT in terms of transcription time and effort can be observed at least for particular participants. To this end, we computed the relative changes in transcription time comparing manual transcription with editing per participant, and the absolute differences in the effort ratings per participant. For each participant, negative values for transcription time difference indicate that editing AMT was faster, whereas negative values for rating difference imply less effort when editing AMT. Figure 2 shows a correlation between the differences in effort and transcription times, which indicates that participants who had a tendency to rate a decreased effort in editing tend also to be those spending less time when editing. The approximately equal number of points in the lower-left and the upper-right quadrant reflects the absence of an overall effect of using AMT on transcription times and effort. The fact that all three expert transcribers (emphasized by circles) spend longer when editing AMT may indicate that providing AMT is not of practical use for experienced transcribers, a point further discussed in the following Section.

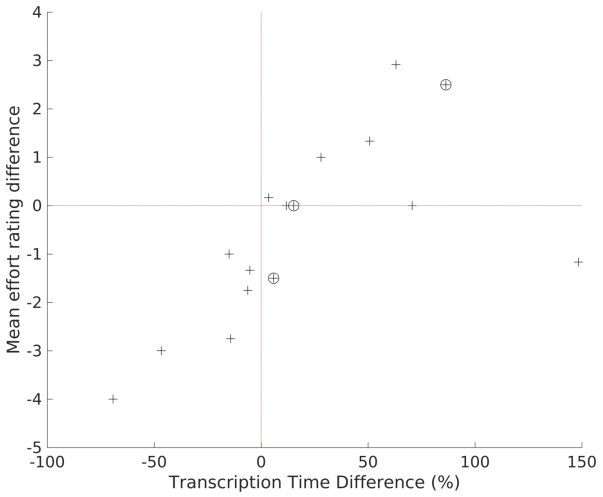


Figure 2. Scatter plot of absolute differences in the effort ratings and change in transcription time. Expert participants are emphasized by a circle.

3.5 Qualitative Results

The experiment was designed to be flexible in terms of providing the participants with exactly those tools for transcription that they would normally use outside of the context of this experiment. Therefore, the choices regarding these tools provide valuable insights into the transcription practice in the context of musicology. Out of the 16 participants, eight decided to transcribe the segments on paper, using mainly the software *Transcribe!* as a tool to loop certain phrases, and to decrease the speed of the playback. The other eight transcribers used notation software, four of them *MuseScore*, and four of them *Sibelius*; the latter exclusively applied by transcribers based in the UK, which indicates differences between the transcription practices based on the local musicology education.

Even though many participants transcribed on paper, all but one participant provided a positive response to the question if AMT tools are able to provide a valuable starting point for a manual transcription. Four of the participants expressed their opinion that use of AMT would be helpful mainly for inexperienced transcribers, and another four explicitly mentioned the potential to save time when using AMT.

A thematic analysis [4] was applied to the questionnaire responses in order to obtain the main reasons for criticism and appraisal of AMT. Four main themes emerged as depicted in Figure 3, three expressing criticism, and one expressing the value of AMT. Most frequently, participants

criticized rhythmic aspects of the AMT, referring most of the time to note durations contained in the AMT. The participants' second most frequent criticism concerns the omission of notes sounding in the recordings, and the addition of notes not heard by the transcriber, with omissions and additions being similarly frequent in the comments. Finally, participants criticized the simultaneous notation of notes, resulting from the notation either of accompaniment notes played on the lute or of overtones of the main instrument. Despite the fact that our questions focused on criticism of the AMT, one positive theme emerged as well, as participants emphasized the value of the AMT to obtain an understanding of the overall shape of the melody.

Summing up the qualitative observations, the most important finding is that current AMT technologies in the context of musicology may have a generally positive value for inexperienced transcribers in terms of pitch information. This value is, however, diminished by the frequent problems related to rhythm, addition/omission of notes, and poor separation of main melody and accompaniment.

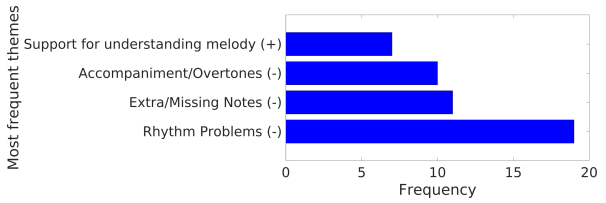


Figure 3. Most frequent themes in the discussions of qualities (+) and problems (-) of the AMT.

4. DISCUSSION

There are several aspects of the proposed study that need to be taken into account before making any claims on the usefulness of AMT in the context of (ethno)musicology. Firstly, the sample size in terms of participants is relatively small, which indicates the difficulty in locating subjects who are trained in transcription and are willing to work with automated methods. It is also difficult to rate the participants' transcription skills: future work could enlist the assistance of transcription instructors and to ask them to rate the participants' transcriptions. Another aspect to take into account is the bias introduced by the AMT system when asking participants to edit transcriptions.

The quantitative evaluation metrics proposed in [15], which were used as part of this study are not error-free: they rely on automatic score-to-score alignment that has been designed to align performance MIDI with reference scores. In particular, the symbolic alignment step could fail in the case of "abstract" transcriptions which could only focus on transcribing notes that are on strong beats, e.g. ignoring any passing notes. Therefore, additional work can be done towards improving the automatic symbolic alignment approach of [16] towards supporting the alignment automatic and manual transcriptions with reference scores.

Additionally, it should be stressed that the present study is focused on the usefulness of AMT in the context of (ethno)musicology, thus not taking into account potential uses of AMT in other application domains. The focus of this study was also on monophonic transcription (despite the presence of polyphony in certain segments); therefore, the usefulness of polyphonic AMT, and also of multiple-instrument AMT technologies, remains to be explored.

Finally, the question posed in the study on the value of AMT could be viewed as suggestive and could have biased participants towards providing a positive answer. There might also be a bias on participants who agreed to take part in the study, since their participation could indicate their general interest into the subject of automatic music transcription, and more generally on the use of technology in the transcription process.

5. CONCLUSIONS

This paper presented a user study on AMT in the context of ethnomusicology. Participants were asked to manually transcribe four segments of folk dance tunes, and to transcribe four different segments of the same style using the output of an AMT system as a starting point. Quantitative analysis shows: a comparative quality between automatic and manual transcriptions; differences between expert and non-expert transcribers, in terms of the time required to carry out transcriptions and also on the metrical quality of the resulting transcriptions; a correlation between the differences in stated effort between manual and edited transcriptions and transcription times; and a correlation between AMT quality ratings and some of the employed quantitative metrics. Finally, qualitative results show support for AMT to obtain an understanding of the overall melodic shape, although combined with criticism of the AMT related to harmonic errors, missing/extra notes, and rhythmic problems. Importantly, however, using an AMT output as a starting point for a transcription did not result in any quantifiable differences regarding quality, transcription time, or effort.

Future work will investigate similarity/dissimilarity of participants' transcriptions, in particular between those by the expert participants and the reference transcriptions, and will liaise with transcription instructors towards grading the resulting transcriptions. We will investigate ways to improve the quantitative metrics of [15] towards a more robust symbolic alignment of automatic and manual transcriptions with reference scores, and conduct evaluations using the newly-proposed metrics of [13]. Exploring whether the conclusions of this paper hold more broadly across other AMT systems and musical repertoires will have important impact on AMT research and on the applicability of AMT in ethnomusicology. We believe that this paper provides a viable and effective framework for user-based evaluation of AMT methods.

6. ACKNOWLEDGMENTS

EB is supported by UK RAEng Research Fellowship RF/128 and a Turing Fellowship. AH is supported by NordForsk's Nordic University Hub "Nordic Sound and Music Computing Network - NordicSMC" (proj. nr.

86892). The authors would like to thank Sven Ahlbäck, Stephen Cottrell, Emir Demirel, Michael Hagleitner, Eita Nakamura, August Schmidhofer, Richard Widdess, and Adrien Ycart for support and feedback.

7. REFERENCES

[1] Website of the Crinnos project. `http://crinnos.ims.forth.gr`. Accessed: 2019-03-26.

[2] M. Bay, A. F. Ehmann, and J. S. Downie. Evaluation of multiple-F0 estimation and tracking systems. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 315–320, 2009.

[3] E. Benetos, S. Dixon, Z. Duan, and S. Ewert. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36(1):20–30, 2019.

[4] V. Braun, V. Clarke, N. Hayfield, and G. Terry. Thematic analysis. In Pranee Liamputtong, editor, *Handbook of Research Methods in Health Social Sciences*, chapter 48, pages 843–860. Springer, 2019.

[5] R. G. C. Carvalho and P. Smaragdis. Towards end-to-end polyphonic music transcription: Transforming music audio directly to a score. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 151–155, 2017.

[6] A. Demetriou, M. A. Larson, and C. Liem. Go with the flow: When listeners use music as technology. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 292–298, 2016.

[7] P. E. Goddard. A graphic method of recording songs. In *Anthropological papers written in honor of Franz Boas*, page 137. New York, 1906.

[8] A. Holzapfel and E. Benetos. The Sousta Corpus: beat-informed automatic transcription of traditional dance tunes. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 531–537, 2016.

[9] J. H. Lee and S. J. Cunningham. The impact (or non-impact) of user studies in music information retrieval. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 391–396, 2012.

[10] J. H. Lee and S. J. Cunningham. Toward an understanding of the history and impact of user studies in music information retrieval. *Journal of Intelligent Information Systems*, 41(3):499–521, 2013.

[11] Z. Li, Q. Xiang, J. Hockman, J. Yang, Y. Yi, I. Fujinaga, and Y. Wang. A music search engine for therapeutic gait training. In *ACM International Conference on Multimedia*, pages 627–630, 2010.

[12] G. List. The reliability of transcription. *Ethnomusicology*, 18(3):353–377, 1974.

[13] A. McLeod and M. Steedman. Evaluating automatic polyphonic music transcription. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 42–49, 2018.

[14] M. Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer, 2015.

[15] E. Nakamura, E. Benetos, K. Yoshii, and S. Dixon. Towards complete polyphonic music transcription: Integrating multi-pitch detection and rhythm quantization. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 101–105, 2018.

[16] E. Nakamura, K. Yoshii, and H. Katayose. Performance error detection and post-processing for fast and accurate symbolic music alignment. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 347–353, 2017.

[17] M. Piszczalski and B. A. Galler. Automatic music transcription. *Computer Music Journal*, pages 24–31, 1977.

[18] M. A. Román, A. Pertusa, and J. Calvo-Zaragoza. An end-to-end framework for audio-to-score music transcription on monophonic excerpts. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 34–41, 2018.

[19] M. Schedl, A. Flexer, and J. Urbano. The neglected user in music information retrieval research. *Journal of Intelligent Information Systems*, 41(3):523–539, 2013.

[20] J. Stanyek. Forum on transcription. *Twentieth-Century Music*, 11(1):101–161, 2014.

[21] D. Weigl and C. Guastavino. User studies in the music information retrieval literature. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 335–340, 2011.

LakhNES: IMPROVING MULTI-INSTRUMENTAL MUSIC GENERATION WITH CROSS-DOMAIN PRE-TRAINING

Chris Donahue[1] **Huanru Henry Mao**[2] **Yiting Ethan Li**[2]
Garrison W. Cottrell[2] **Julian McAuley**[2]
[1] Department of Music, UC San Diego [2] Department of Computer Science, UC San Diego

ABSTRACT

We are interested in the task of generating multi-instrumental music scores. The *Transformer* architecture has recently shown great promise for the task of piano score generation—here we adapt it to the multi-instrumental setting. Transformers are complex, high-dimensional language models which are capable of capturing long-term structure in sequence data, but require large amounts of data to fit. Their success on piano score generation is partially explained by the large volumes of symbolic data readily available for that domain. We leverage the recently-introduced *NES-MDB* dataset of four-instrument scores from an early video game sound synthesis chip (the NES), which we find to be well-suited to training with the Transformer architecture. To further improve the performance of our model, we propose a pre-training technique to leverage the information in a large collection of heterogeneous music, namely the Lakh MIDI dataset. Despite differences between the two corpora, we find that this transfer learning procedure improves both quantitative and qualitative performance for our primary task.

1. INTRODUCTION

In this paper, we extend recent results for symbolic piano music generation [1] to the multi-instrumental setting. Both piano and multi-instrumental music are *polyphonic*, where multiple notes may be sounding at any given point in time. However, the generation of multi-instrumental music presents an additional challenge not present in the piano domain: handling the intricate interdependencies between multiple instruments. Another obstacle for the multi-instrumental setting is that there is less data available than for piano, making it more difficult to train the types of powerful generative models used in [1].

Until recently, music generation methods struggled to capture two rudimentary elements of musical form: long-term structure and repetition. Huang et al. [1] demonstrated that powerful neural network *language models*, i.e., models which assign likelihoods to sequences of discrete tokens, could be used to generate classical piano music containing these elusive elements. In order to adapt this method to the multi-instrumental setting we incorporate instrument specification directly into our language-like music representation. However, this strategy alone may be insufficient to generate high-quality multi-instrumental music, as the results of [1] also depend on access to large quantities of piano music.

To begin to address the data availability problem, we focus on an unusually large dataset of multi-instrumental music. The *Nintendo Entertainment System Music Database* (NES-MDB) [2] contains 46 hours of *chiptunes*, music written for the four-instrument ensemble of the NES (video game system) sound chip. This dataset is appealing for music generation research not only for its size but also for its structural homogeneity—all of the music is written for a fixed ensemble. It is, however, smaller than the 172 hours of piano music in the *MAESTRO Dataset* [3] used to train Music Transformer.

The largest available source of symbolic music data is the *Lakh MIDI Dataset* [4] which contains over 9000 hours of music. This dataset is structurally heterogeneous (different instruments per piece) making it challenging to model directly. However, intuition suggests that we might be able to benefit from the musical knowledge ingrained in this dataset to improve our performance on chiptune generation. Accordingly, we propose a procedure to heuristically map the arbitrary ensembles of music in Lakh MIDI into the four-voice ensemble of the NES. We then pre-train our generative model on this dataset, and fine-tune it on NES-MDB. We find that this strategy improves the quantitative performance of our generative model by 10%. Such transfer learning approaches are common practice in state-of-the-art natural language processing [5, 6], and here we develop new methodology to employ these techniques in the music *generation* setting (as opposed to analysis [7]).

We refer to the generative model pre-trained on Lakh MIDI and fine-tuned on NES-MDB as *LakhNES*. In addition to strong quantitative performance, we also conduct multiple user studies indicating that LakhNES produces strong qualitative results. LakhNES is capable of generating chiptunes from scratch, continuing human-composed material, and producing melodic material corresponding to human-specified rhythms. [1]

[1] Sound examples: `https://chrisdonahue.com/LakhNES`
Code/data: `https://github.com/chrisdonahue/LakhNES`

2. RELATED WORK

Music generation has been an active area of research for decades. Most early work involved manually encoding musical rules into generative systems or rearranging fragments of human-composed music; see [8] for an extensive overview. Recent research has favored machine learning systems which automatically extract patterns from corpora of human-composed music.

Many early machine learning-based systems focused on modeling simple *monophonic* melodies, i.e., music where only one note can be sounding at any given point in time [9–11]. More recently, research has focused on *polyphonic* generation tasks. Here, most work represents polyphonic music as a *piano roll*—a sparse binary matrix of time and pitch—and seeks to generate sequences of individual piano roll timesteps [12, 13] or chunks of timesteps [14]. Other work favors an *event-based* representation of music, where the music is flattened into a list of musically-salient events [1, 15, 16]. None of these methods allow for the generation of multi-instrumental music.

Other research focuses on the multi-instrumental setting and seeks to provide systems which can *harmonize* with human-composed material [17–20]. Unlike the system we develop here, these approaches all require complex inference procedures to generate music without human input. Recent work [21–23] attempts multi-instrumental music generation from scratch, but these methods are limited to generating fixed lengths, unlike our method which can generate arbitrarily-long sequences. There is also music generation research that operates on the audio domain [24, 25], though this work is largely unrelated to symbolic domain methods. The work described in this paper is methodologically similar to MuseNet [26], which was concurrent with our work.

3. DATASETS AND TASK

The *NES Music Database* (NES-MDB) [2] consists of approximately 46 hours of music composed for the sound chip on the Nintendo Entertainment System. This dataset is enticing for research in multi-instrumental music generation because (1) it is an unusually large corpus of music that was composed for a fixed ensemble, and (2) it is available in symbolic format.

3.1 NES ensemble preliminaries

The ensemble on the NES sound chip has four monophonic instrument voices: two pulse waveform generators (P1/P2), one triangle waveform generator (TR), and one noise generator (NO). [2] The first three of these instruments are melodic voices: typically, TR plays the bass line and P1/P2 are interchangeably the melody and harmony. The noise instrument is used to provide percussion.

The various instruments have a mixture of sound-producing capabilities. For example, the range of MIDI pitches which P1/P2 can generate is 33–108, while the range of TR extends an octave lower (21–108). The noise channel can produce 16 different "types" of noise which correspond to different center frequencies and bandwidths. Each instrument also has a variety of dynamics and timbral attributes. It is shown in [2] that these expressive attributes can be estimated from the score post-hoc, and hence we ignore them in this study to focus on the problem of modeling composition rather than expressive performance.

Each chiptune in NES-MDB is stored as a MIDI file, and the constituent MIDI events are quantized at audio rate (44100 *ticks* per second). Paired with code which synthesizes these MIDI files as NES audio, the files contain all of the information needed to synthesize the original 8-bit waveforms.

[2] There is an additional fifth voice capable of waveform playback that the authors of NES-MDB excluded.

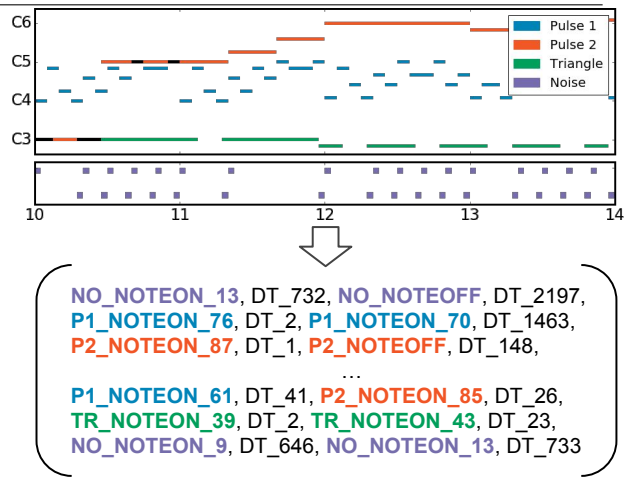


Figure 1. A visual comparison between the piano roll representation of the original NES-MDB paper [2] (**top**) and the event representation of this work (**bottom**). In the piano roll representation, the majority of information is the same across timesteps. In our event representation, each timestep encodes a musically-meaningful change.

3.2 Event-based task

In our original work on NES-MDB [2], we operated on a *piano roll* representation of the data, i.e., the MIDI information decomposed into a sparse grid across time, pitch, and instrument (top of Figure 1). Because no tempo or beat information exists in the dataset, the authors chose to discretize the time axis at a rate of 24 timesteps per second. This high rate is necessary for capturing nuanced timing information in the scores but results in much of the information being redundant across adjacent timesteps. This represents a challenge as long-term dependencies are a barrier to success for sequence modeling with machine learning.

To circumvent these issues, we design an *event-based* representation (bottom of Figure 1) similar to that used for single-instument music in [15]. Specifically, we convert each NES-MDB MIDI file into a time-ordered sequence of events, so that every entry in the sequence corresponds to a musically-salient occurrence.

To handle the rhythmic information, we add time shift (ΔT) events which represent time advancing by some dis-

Event description	Event ID(s)
Start or end of sequence	0
ΔT for 1–100 ticks (short)	1–100
ΔT for 100–1000 ticks (medium)	101–190
ΔT for $>$ 10000 ticks (long)	191–370
P1 Note Off/On	371–447
P2 Note Off/On	448–524
TR Note Off/On	525–613
NO Note Off/On	614–630

Table 1. Schematic for our event-based representation of NES-MDB, reminiscent of the one used in *Performance RNN* [15]. The 631 events in our representation are distributed among time-shift (ΔT) events (which allow for nuanced timing), and note off/on events for individual instruments (as in typical MIDI).

crete number of ticks (each tick is $\frac{1}{44100}$th of a second). To keep the number of events in our representation tractable, we quantize ΔT events in the real data to fixed gratings. We embed the multi-instrumental aspect of our problem directly into this representation by using separate note on/off events for each instrument. Contemporaneous events are always listed in the following instrument order: P1, P2, TR, NO. Our final representation consists of 631 events, of which about half encode time-related events and half note-related (Table 1). Apart from minor timing quantization, this format is a lossless transformation of the original MIDI score.

4. METHODOLOGY

To model the event sequences outlined in the last section, we adopt a *language modeling* factorization. We factorize the joint probability of a musical sequence consisting of N events $(E_1, \ldots, E_N)$ into a product of conditionals:

$$P(E_1) \cdot P(E_2 \mid E_1) \cdot \ldots \cdot P(E_N \mid E_1, \ldots, E_{N-1}). \quad (1)$$

This factorization is convenient because it allows for a simple left-to-right algorithm for generating music: sampling from the distribution estimated by the model at each timestep (conditioned on previous outputs). The goal of our optimization procedure is to find a model configuration which maximizes the likelihood of the real event sequences. Motivated by the strong results for piano music generation from the recent *Music Transformer* [1] approach, we also adopt a Transformer [27] architecture.

4.1 Transformer architecture

The Transformer [27] is an attention-based neural network architecture. In our context, this means that the model has a mechanism which explicitly biases its predictions based on a subset of musical events that have happened in the past. The model's design gives it the ability to learn which subset of past musical events to pay attention to when predicting the current event. This mechanism may be especially useful for learning patterns of repetition in music across large gaps of time.

The original Transformer architecture [27] was an encoder-decoder model designed for language translation. In this paper, we are only concerned with the decoder portion of the Transformer. Our work uses a recent extension of Transformer called Transformer-XL [28], which is designed specifically to handle longer sequences. Transformer-XL builds upon the Transformer architecture by augmenting it with a recurrence mechanism. The recurrence mechanism enables Transformer-XL to use information beyond its training segment by learning how to incorporate recurrent state from previous segments. In contrast, the original Transformer is only able to alter its predictions based on the current training segment, hence the available system memory during training is a bottleneck to its ability to learn long-term dependencies. In order to effectively use its recurrent state, Transformer-XL adopts a sophisticated position-aware mechanism so the model can generalize to different amounts of recurrent memory during generation.

The Music Transformer [1] is a different Transformer variant that also attempts to tackle long-range dependencies by using a mechanism which reduces the quadratic memory cost of attention, enabling training on longer sequences. Although similar in goal to Transformer-XL, its method is orthogonal and could, in theory, be combined with the recurrent mechanism of Transformer-XL. For simplicity, we focus on the Transformer-XL architecture as its recurrence mechanism alone is sufficient to learn long-term dependencies. Additionally, code to reproduce the Music Transformer method is unavailable.

4.2 Pre-training

Transformers are extremely high-dimensional models, and accordingly they can learn effective strategies for extremely large datasets [6]. One barrier to their application in the music domain is that most symbolic music datasets are either too small or too structurally heterogeneous. For example, the popular Bach chorales dataset [29] is structurally homogeneous (all chorales have four voices), but small (only 306 chorales). In contrast, the Lakh MIDI dataset [4] is enormous (175k songs) but heterogeneous (varying numbers of instruments per piece). The NES-MDB dataset we use in this work represents a middle ground (large and structurally-homogenous), but is still substantially smaller than the MAESTRO dataset [3] used to train Music Transformer (46 hours vs. 172 hours).

We hypothesize that we can improve the performance of our model on our NES music generation task by leveraging the musical information in the larger Lakh MIDI dataset. To test this, we propose a two-step procedure. First, we map each structurally-heterogeneous Lakh MIDI file into one which can be performed by our NES ensemble. Then, we pre-train a Transformer on this dataset, and fine-tune this pre-trained model on the NES-MDB dataset. Such transfer learning procedures are common methodology in other areas of machine learning [30], but remain hitherto unexplored in music generation research. One possible

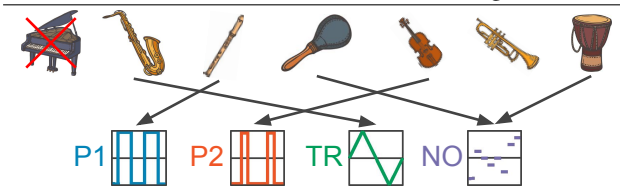


Figure 2. Illustration of our mapping heuristic used to enable transfer learning from Lakh MIDI to NES-MDB. We identify monophonic instruments from the arbitrary ensembles in Lakh MIDI and randomly assign them to the fixed four-instrument ensemble of NES-MDB.

reason for the lack of investigation into this strategy is that mapping music from one domain to another requires careful consideration of musical invariants, and hence is less straightforward than analogous methodology for other tasks (e.g., language). We consider this transfer learning protocol to be a primary methodological contribution of this work.

4.2.1 Mapping Lakh MIDI to the NES ensemble

Here we describe our protocol for mapping Lakh MIDI data into a score suitable for the four monophonic instruments of the NES ensemble. For a given example from Lakh MIDI, we first identify all of its monophonic melodic instruments (skipping the example if it has no such instruments). Then, we filter out instruments which fall outside of the range of MIDI notes that the NES ensemble is capable of producing (Section 3.1). We randomly assign these instruments to the three melodic instruments of the NES (P1/P2/TR) (Figure 2). Because there are a variable number of instruments in each Lakh MIDI example, there are potentially many possible assignments. Hence, we output multiple examples for each input Lakh MIDI example.

In addition to this strategy for melodic instruments, we also design a strategy for mapping percussive instruments in Lakh into the percussive noise instrument of the NES ensemble. We first identify percussive instruments in each Lakh MIDI example. Then, each individual percussive voice (e.g., snare drum, hi-hat) is randomly assigned to a noise "type" (1–16), emulating how the noise instrument is used by human composers to encode syncopated rhythms.

From the 175k MIDI files in Lakh MIDI, our mapping procedure produces 775k examples suitable for performance by the NES ensemble. It is straightforward to imagine similar mapping procedures for other ensembles (e.g., string quartet, vocal choir), and thus it is possible that music generation research in other domains could reuse this procedure to enable transfer learning.

5. EXPERIMENTS

We first conduct an experiment to train Transformer-XL [28] on our event representation (Section 3.2) of NES-MDB. We train the model on excerpts from the training data of 512 events; each excerpt represents around 9 seconds of music on average. Because of the recurrent attention mechanism in Transformer-XL, the model effectively has access to twice this length in its history.

We use the smaller configuration of Transformer-XL which has 12 attention layers each with 8 heads. The learning rate $2e{-}4$ used to train this model on text was found to be too high for our musical application, so we lowered it to $2e{-}5$. Training was stopped when the performance of the model on the validation data stopped improving. We trained the model using four NVIDIA Titan X GPUs with minibatches of size 30, and it reached its early stopping criteria in less than a day. [3]

5.1 Data augmentation and pre-training

To improve the performance of our model further, we employed standard music data augmentation methods as well as ones which we developed specifically for the multi-instrumental setting:

1. (Standard) Transpose melodic voices by a random number of semitones between -6 and 5 (inclusive).
2. (Standard) Adjust the speed of the piece by a random percentage between $\pm 5\%$.
3. Half of the time, remove a random number of instruments from the ensemble (leaving at least one).
4. Half of the time, shuffle the score-to-instrument alignment for the melodic instruments only (e.g., TR performs P2's part).

Finally, we experimented with pre-training our model on the Lakh MIDI dataset mapped to the NES ensemble (Section 4.2.1). To conduct this experiment, we first split the Lakh data into training and validation subsets. We then trained the model for a week on the training set (with data augmentation) and monitored performance on the validation set. Because of the extreme size of the dataset, the model only completed four epochs of training. Even after a week, the model was underfitting the training data (validation performance was still improving). We then fine-tuned this pre-trained model on the NES-MDB training data, again performing early stopping based on the validation performance. Both our pre-training and fine-tuning experiments use the same hyperparameters outlined in the previous section.

5.2 Baselines

We also measure the performance of competitive baselines on our event-based representation of NES-MDB. Our simplest baselines consist of n-gram models, i.e., statistics gathered directly from the training data of how often certain length-n sequences appear. Specifically, we build unigram (1-gram) and 5-gram models, using backoff for the latter to provide a likelihood for 5-grams which are not present in the training data. We also compare to an LSTM [31] recurrent neural network, which is a popular model for music generation. Our LSTM is configured so

[3] Full hyperparameter description and pre-trained models: `https://github.com/chrisdonahue/LakhNES`

Model	Params	Epochs	Test PPL
Random	0	0	631.00
Unigram	631	1	198.14
5-gram	9M	1	37.25
LSTM [31]	40M	18	14.11
+Data augmentation		35	12.64
Transformer-XL [28]	41M	76	3.50
+Data augmentation		350	2.74
+Pre-train (LakhNES)		250	**2.46**

Table 2. Quantitative performance of various models trained on the event-based representation (631 event types) of NES-MDB. *Params* indicates the number of parameters of each model. *Epochs* is the number of data epochs the model observed before early stopping based on the validation data. *Test PPL* represents the perplexity of the model on the test data, i.e., the exponentiation of its average negative log-likelihood on the test data. A lower perplexity indicates that the model better fits this unseen data.

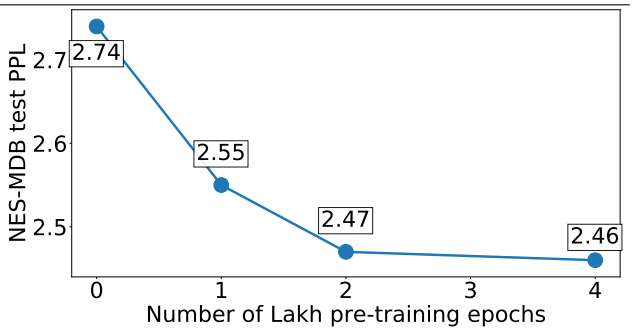


Figure 3. Measuring the performance improvement when doubling the amount of Lakh MIDI pre-training before fine-tuning Transformer-XL on NES-MDB. Each datapoint represents the result of a fine-tuning run starting from 0, 1, 2, or 4 epochs of Lakh MIDI pre-training. Additional amounts of pre-training appear to improve performance, though with diminishing returns.

that it has approximately the same number of parameters as our Transformer-XL model (1 layer, 3072 units).

6. QUANTITATIVE ANALYSIS

We report the *perplexity* (PPL) of each model on the test set in Table 2. Perplexity is calculated by first averaging the negative log-likelihood of each model across the test data, then exponentiating the average, i.e., $e^{\frac{1}{N}\sum_{i=1}^{N} -\log q_i}$, where q_i is the likelihood assigned by a given model to the i-th event. A lower perplexity on the test set indicates that a model is a good fit for unseen data, and hence increases our confidence in its ability to generate new music.

We find that Transformer-XL dramatically outperforms both the n-gram and LSTM baselines on the NES-MDB event-based task (PPL of 3.5 vs. 37.2 and 14.1 respectively). Data augmentation improves the performance of both the LSTM and Transformer-XL (by 10% and 22% respectively), and also increases the number of epochs before the models overfit. We observe that LakhNES (Transformer-XL pre-trained on Lakh MIDI and fine-tuned on NES-MDB with augmentation), achieves 10% better performance than training with data augmentation alone.

We also conduct an experiment to measure the performance effect of using different amounts of Lakh MIDI pre-training before fine-tuning on NES-MDB. Specifically, we measure the performance on the NES-MDB fine-tuning task after 1, 2, and 4 epochs of Lakh MIDI pre-training. We plot the test PPL of each model after fine-tuning in Figure 3. The results agree with our expectation that increasing the amount of pre-training improves the fine-tuned model's performance, though with diminishing returns.

7. USER STUDY

While perplexity is a useful quantitative metric for model comparison, it is not necessarily correlated with human judgements. Since we ultimately seek models which produce music that is convincing to humans, we conduct two user studies on Amazon Mechanical Turk to compare the performance of various models. In both of our user studies we compare four models (rows 3, 5, 7, 8 from Table 2): (1) 5-gram model, (2) LSTM trained with data augmentation, (3) Transformer-XL trained with data augmentation (TXL), and (4) Transformer-XL with data augmentation and Lakh MIDI pre-training (LakhNES).

7.1 Turing test

This study seeks to determine the ability of humans to distinguish between real (human-composed) and fake (computer-generated) chiptunes in a "Turing test" setting. We present human judges with pairs of examples where one example is real and the other fake, and ask them to identify the real example between the two.

We first amass collections of 5-second audio clips from all of our methods and from the real data by selecting random slices from the variable-length music. Then, we create pairs of examples where one example is real and the other fake (randomly chosen from our four methods). Given that Mechanical Turk studies are notoriously noisy, we also create control pairs where the fake data comes from a random model (i.e., we generate "music" by selecting events uniformly at random—row 1 in Table 2).

We ask human judges to annotate 800 batches each consisting of 10 randomly-ordered pairs, where fake data in 2 of the pairs came from the control set and fake data in 8 of the pairs came from our four methods. For their judgments to be included in our results, workers were required to complete at least 3 batches and achieve 100% accuracy on the 6 control examples in those batches—a worker answering randomly would only be included 1.6% of the time. After filtering, each method was evaluated around 180 times. We report accuracy in Figure 4.

In this setting, a lower accuracy indicates that a given model's results sound more human-like, because they were incorrectly identified as human-composed more often. An

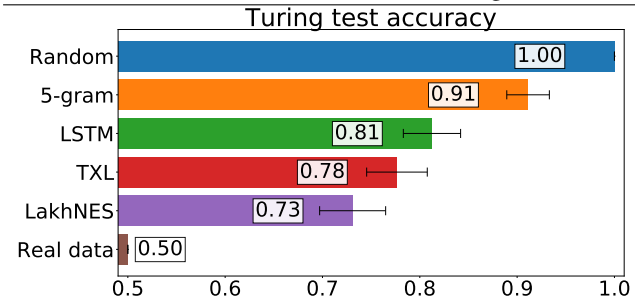


Figure 4. Human accuracy at distinguishing computer-generated examples from human-composed ones (error bars are standard error). Users were presented with pairs of clips (one human, one computer) and tasked with identifying which was composed by a human. Random examples are used as a control and we filtered annotators with accuracy less than 1 on those pairs. A lower accuracy is better as it indicates that the annotators confused a particular model with the real data more often.

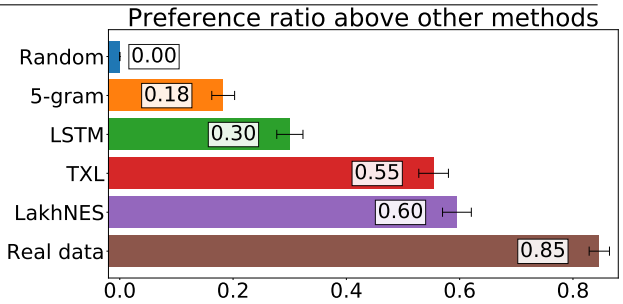


Figure 5. Proportion of comparisons where humans preferred an example from each model over an example from another random model (error bars are standard error). Users were presented with pairs of clips from different methods and asked which they preferred. Pairs of random data and human-composed clips are used as a control and we filtered annotators who preferred random. A higher ratio is better as it indicates that the annotators preferred results from that method more often than another.

ideal generative model would achieve 50% accuracy (although it is possible in theory to generate music which sounds "more human" than human-composed music). We find that LakhNES (Transformer-XL with pre-training) was mistakenly identified as human more often than both our 5-gram model ($p < .0001$ by t-test with normal approximation) and our LSTM ($p = .07$). It also outperformed Transformer-XL without pre-training, but the difference was not statistically significant ($p = .32$).

Overall, these results suggest that there is still a sizable gap between human-composed and computer-generated chiptunes. Subjectively speaking, we feel that the melodies and harmonies produced by LakhNES are promisingly human, but its inability to maintain rhythmic consistency is often a dead giveaway in a Turing test. We suspect that our model could be improved by using a beat-based event representation, however the current model can be bootstrapped with human-specified rhythmic material to manually address rhythmic consistency issues (Section 8).

7.2 Preference test

In addition to our Turing test, we also conduct a preference-based user study, given that human-ness is not necessarily a predictor of general preference. We present human judges with pairs of examples from two different methods, and ask them which of the two they "prefer".

Here we construct pairs of 10-second clips from two different (randomly-chosen) methods. These clips are twice as long as those used in Section 7.1 allowing longer-term structure to influence preference decisions. As in our Turing test, we construct randomly-ordered batches consisting of 10 pairs. In each batch, 8 of the pairs are created by sampling two methods without replacement from a set of five (four computer-generated and the real data), while 2 pairs always compare randomly-generated clips to real data (control). We ask human judges to assign preference to these batches, filtering out workers who even once indicated that they preferred random examples to real data.

After filtering, each of the five methods was involved in around 400 comparisons in total. We report the ratio of "wins" for each method in Figure 5, i.e., the proportion of times a method was preferred over any of the other four.

We find that LakhNES outperforms all other generative methods, though is preferred significantly less often than the real data. Human judges preferred chiptunes generated by LakhNES over the real data in 26% of comparisons (vs. only 10% of the time for the LSTM). We find this to be a promising indicator of Transformer's potential on this task.

8. PAIRING LAKHNES WITH HUMANS

In addition to generating chiptunes from scratch, LakhNES can be used for a number of tasks to assist human composers. For example, LakhNES can be "primed" on human-composed material and then asked to continue the material, providing a method for composers to quickly expand on their ideas. Composers can also provide fixed rhythmic material and use LakhNES to generate the rest of the score. We explore these use cases in our sound examples: `https://chrisdonahue.com/LakhNES` . When generating all of our sound examples (besides those in our user study), we found that limiting the entropy of the model by using a sampling temperature of .95 and top-k sampling [32] with $k = 32$ improved results qualitatively.

9. CONCLUSION

In this paper we presented LakhNES, a method for learning to generate multi-instrumental music. We developed an event-based representation suitable for this task. Training powerful language models on this representation results in compelling multi-instrumental music generation. We show that we can further improve results both quantitatively and qualitatively by pre-training on a cross-domain dataset. LakhNES can be used to both generate chiptunes from scratch and collaborate with human composers.

10. ACKNOWLEDGEMENTS

Thanks to Cheng-Zhi Anna Huang, Cheng-i Wang, and Jennifer Hsu for helpful discussions regarding this work. This work was supported by UC San Diego's Chancellors Research Excellence Scholarship program. GPUs used in this research were donated by NVIDIA.

11. REFERENCES

[1] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music Transformer: Generating music with long-term structure. In Proc. *ICLR*, 2019.

[2] Chris Donahue, Huanru Henry Mao, and Julian McAuley. The NES Music Database: A multi-instrumental dataset with expressive performance attributes. In Proc. *ISMIR*, 2018.

[3] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In Proc. *ICLR*, 2019.

[4] Colin Raffel. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. PhD thesis, Columbia University, 2016.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018.

[6] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, 2019.

[7] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Transfer learning for music classification and regression tasks. In Proc. *ISMIR*, 2017.

[8] Gerhard Nierhaus. *Algorithmic composition: paradigms of automated music generation*. Springer Science & Business Media, 2009.

[9] Peter M Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 1989.

[10] Michael C Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, 1994.

[11] Douglas Eck and Jürgen Schmidhuber. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In Proc. *Neural Networks for Signal Processing*, 2002.

[12] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In Proc. *ICML*, 2012.

[13] Daniel D Johnson. Generating polyphonic music using tied parallel networks. In Proc. *International Conference on Evolutionary and Biologically Inspired Music and Art*, 2017.

[14] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. In Proc. *ISMIR*, 2017.

[15] Ian Simon and Sageev Oore. Performance RNN: Generating music with expressive timing and dynamics. `https://magenta.tensorflow.org/performance-rnn`, 2017.

[16] Huanru Henry Mao, Taylor Shin, and Garrison Cottrell. DeepJ: Style-specific music generation. In Proc. *International Conference on Semantic Computing*, 2018.

[17] Moray Allan and Christopher Williams. Harmonising chorales by probabilistic inference. In Proc. *NIPS*, 2005.

[18] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. In Proc. *ISMIR*, 2017.

[19] Gaëtan Hadjeres and François Pachet. DeepBach: A steerable model for Bach chorales generation. In Proc. *ICML*, 2017.

[20] Yujia Yan, Ethan Lustig, Joseph VanderStel, and Zhiyao Duan. Part-invariant model for music generation and harmonization. In Proc. *ISMIR*, 2018.

[21] Hao-Wen Dong and Yi-Hsuan Yang. Convolutional generative adversarial networks with binary neurons for polyphonic music generation. In Proc. *ISMIR*, 2018.

[22] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. In Proc. *ICML*, 2018.

[23] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In Proc. *AAAI*, 2018.

[24] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. In Proc. *ICLR*, 2018.

[25] Sander Dieleman, Aäron van den Oord, and Karen Simonyan. The challenge of realistic music generation: modelling raw audio at scale. In Proc. *NeurIPS*, 2018.

[26] Christine McLeavy Payne. MuseNet. `https://openai.com/blog/musenet/`, 2019.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Proc. *NIPS*, 2017.

[28] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. *arXiv:1901.02860*, 2019.

[29] Hermann Hild, Johannes Feulner, and Wolfram Menzel. HARMONET: A neural net for harmonizing chorales in the style of JS Bach. In Proc. *NIPS*, 1992.

[30] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 2010.

[31] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[32] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In Proc. *ACL*, 2018.

TAKING FORM: A REPRESENTATION STANDARD, CONVERSION CODE, AND EXAMPLE CORPUS FOR RECORDING, VISUALIZING, AND STUDYING ANALYSES OF MUSICAL FORM

Mark Gotham
Cornell University
Department of Music

Matthew T. Ireland
University of Cambridge
Sidney Sussex College

ABSTRACT

We report on new specification standards for representing human analyses of musical form which enable musicians to represent their analytical view of a piece either on the score (where an encoded version is available) or on a spreadsheet. Both of these representations are simple, intuitive, and highly human-readable. Further, we provide code for converting between these formats, as well as a nested bracket representation adopted from computational linguistics which, in turn, can be visualised in familiar tree diagrams to provide 'at a glance' introductions to works. Finally, we provide an initial corpus of analyses/annotations in these formats, report on the practicalities of amassing them, and offer tools for automatic comparison of the works in the corpus based on the content and structure of the annotations. We intend for this resource to be useful to computational musicologists, enabling study of form at scale, and also useful pedagogically to all teachers, students, and appreciators of music from whom projects of this kind can be rather disconnected. The code and corpus can be found at `https://github.com/MarkGotham/Taking-Form`

1. INTRODUCTION

Marking up scores is a fundamental part of life for any musician working in a score-based medium including – but not entirely limited to – Western classical music. The specifics vary widely from analytical observations to instrument fingerings, but the wider goals can be understood in the same terms: commenting on the music to clarify, visualise, remind, and reinforce a particular understanding of the music in question or plan for its execution.

Regrettably, these annotations are rarely stored, kept, and shared effectively. For instance, an ensemble may assiduously mark up a set of parts, but at the end of the hire period, publishers require them to erase those annotations. (Admittedly this is routinely flouted.) Some conductors travel with their own sets of parts for this reason; others waste rehearsal time conveying markings that could well have been in the score in the first place.

Those score markings represent one of many forms of siloed knowledge that we cannot easily share and repurpose. The same is true of most musical analysis which is published in articles and monographs which time-pressed musicians for the most part simply do not read, and from which the information is hard to extract automatically.

The digital age presents potential solutions to many of these problems, and computational musicology is beginning to answer the call with solutions that centre on representing musico-analytical information in a structured fashion. For instance, formats designed to enable the representation, study, and re-purposing of Roman-numeral analyses include the 'TAVERN', 'ABC', and 'RomanText' projects among others [5, 13, 19]; Automated extraction of performance data include [6, 7], and 'Tuttitempi'; [1] and even pencil-on-paper performers' markings have become an extricable dataset [1].

A key part of the equation here is the opportunity for musical visualization; prior work in this area sees a close relationship between efforts to create format standards for representing musical structure on the one hand, and visualising those structures on the other. This has included handling note-by-note annotations ab initio (such as Dezrann [8] and Verovio [14]) as well as a range of ad hoc visualizations for other research projects like the 'Ribbon' formal analysis of the Josquin Research Project [16]. [2]

This expands the more established practice of visualizing audio analysis in which Fourier or wavelet transforms visualise spectral content (routine in commercial softwares); beat trackers visualise the micro-timing of performed tempo; [3] and even the recorded waveform itself visualises the dynamic profile (an easily forgotten, but eminently useful resource). [4]

2. FORM

Form is a particularly pertinent candidate for stronger representation. The range of use cases includes computational study and visualizations for education in the widest sense:

[1] `https://tuttitempi.com/`
[2] `http://josquin.stanford.edu/`
[3] For instance, see Sonic Visualiser [2]
[4] See [4] for a summary of other relatively recent audio visualizations.

not only of those formally involved in traditional modes of teaching and learning, but anyone seeking to attain or maintain a stronger grasp of the work in question.

Discussion of form necessarily involves comparisons across the full range of the work in question. Keeping all of this information in mind is a challenge and thus form is a parameter particularly in need of schematic representations: while a harmonic or melodic device can often be reproduced exactly, that is almost never possible with form. This has led to a range of creative representations of large-scale structures with clear, at-a-glance summaries.

This paper reports on an attempt to represent formal analyses such that they can be stored, visualised, and studied effectively. We accommodate any number of internal divisions, and envisage this being used to mark up analytical views of everything not specified in the score: that is divisions right down to the level of hypermetrical / phrase groupings. In principle, this could be continued further, to represent the structure of the measure level and beat groupings. [5] That may well be useful for metrically complex works; we include one example in the 'Miscellany' corpus (see Section 3) but do not explore it further here as those details are usually clear from the notation in the common practice repertoire we are primarily targetting. Instead we focus on representing analytical views of those un-notated, hypermetrical and formal levels above the single measure.

We propose three similarly expressive standards for representing analyses, and we provide code for converting between those representations. *On-score mark-up* (Section 2.2) forms the most natural user interface for musicians accustomed to making such annotations; *Abstract syntax trees* (Section 2.1) present a final visualization format; and the *tabular standard* (Section 2.3) is a flexible intermediary encoding useful for overcoming some of the complexities inherent in annotating a score with labels that denote position in a hierarchical structure. We now begin at the 'extremes', as it were, with the (final) visualization format and (initial) on-score mark-up, before proceeding to 'join the dots', discussing the intermediary standard.

2.1 Abstract Syntax Trees (ASTs)

In dealing with form, we face the specific problem of representing hierarchical information, and thus we respond to an explicit call from [15] for a 'Standard Format' for 'Hierarchical Analyses and Representations', including form. Previous work includes Craig Sapp's automation and visualization of 'Hierarchical Key Analysis' [18], and the 'Variations Audio Timeliner' [6] approaches form specifically (though it not open source). There is also a well established precedent for using 'tree' structures (borrowed from computational linguistics) to represent this hierarchical information in music.

Perhaps the most famous linguistics-music cross-over, 'A Generative Theory of Tonal Music' [11], applied a

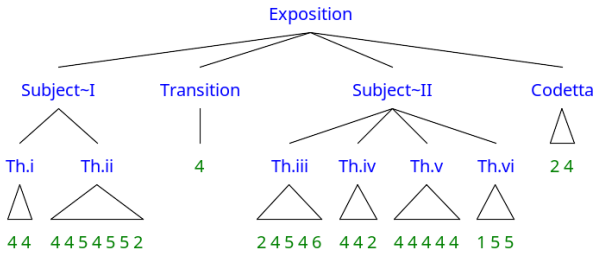


Figure 1. Example tree visualisation: the exposition from the first movement of Schubert's Symphony No.8. The abbreviations are 'Subject' for 'Subject Group', and 'Th.' for 'Theme'.

version of ASTs to discuss metrical and grouping well-formedness and preference at the note level in Schenkerian terms. [17] has applied the same logic to the different 'levels' of harmonic identity, from the chord symbol (such as 'C') through Roman-numerals ('I') to Riemannian functions ('T'). [7]

Figure 1 provides an example of such an AST representation of a passage of musical form: the exposition from the first movement of Schubert's Symphony No.8. The figure includes structural groupings from the highest level down to the level of measure grouping: the numbers of the lowest level refer to the number of measures in each successive phrase grouping.

This paper provides a specification format for producing such visualisation on the basis of well-structured representations of form and phrase in music, as well as code for converting between this representation format, various kind of tree visualizations, and, crucially, encoded scores. With these tools, we can mark observations directly onto scores (which is pedagogically invaluable) and keep them in a structured format for representations and analysis.

In the specification format proposed, each mark-up annotation (labelled on a score) maps directly to a node in the AST. The level (distance from the root node) in the AST must be explicitly encoded within the annotation.

2.2 On-score mark-up

There are many ways to encode this information. This paper reports on a single standard for doing so which is highly flexible both in terms of the types and terms of divisions used, and also the input method. We begin with perhaps the most user-friendly method: marking formal divisions directly onto encoded scores. All that is required is a music notation software which will export to the interoperable standard musicXML. Most notation software support this, including both commercial and free/open-source options like MuseScore.

To enter a marking, simply create a 'stave text' as described in the following instructions which combine the

[5] Our computational framework could be extended to calculate and visualise any arbitrary summary statistic, measuring any lowest level divisions.

[6] variations.sourceforge.net/vat/

[7] For more on datasets for and visualisations of GTTM and Schenkerian reductions see [10, 12] and the work of Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo, including [9] and Masatoshi Hamanaka's XML markups of musical examples from (and using the tree structure representation of) GTTM [11] `http://gttm.jp/gttm/`

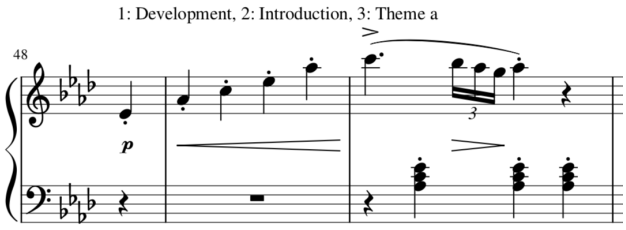


Figure 2. Example file markup in the first movement of Beethoven's Piano Sonata op.2 no.1.

definition of the minimal well-formedness constraints with 'how to' style instructions. Figure 2 illustrates what this looks like on a score.

- First, click on any note at the right measure and beat position for the marking you want to enter.
- Enter a new textual marking at that position (most software provide CMD+T (Mac) / CNTRL+T (Windows) as a shortcut for this).
- Begin each on-score mark-up with an initial character to specify a 'level number' for the marking in question, followed by a colon. For instance, 'Exposition', 'Development' and 'Recapitulation' will all be on the top level of division (number 1) in sonata form movements, so '1: Exposition'. All other levels continue the divisions from here, (e.g. '2: First Subject Group').
- Where you wish to indicate a division, but have no name for the span in question, use the correct level number and a placeholder text like '4: X'.
- In practice, many entries like '1: Exposition' and '2: First Subject Group' will begin at the same time. To indicate these multiple, simultaneous level entries, insert one text entry with all the component parts divided by a comma (','). For instance, many sonatas will begin with the long string: '1: Exposition, 2: First Subject Group, 3: Theme a, 4: Sentence, 5: Presentation, 6: Basic Idea'.

All entries are relative to each other, so level numbers are given by finding the directly relevant parallel. These will fall into a few basic types of comparison:

- A span and its first phrase-division will generally be used together, with the division at +1 level. For a phrase, we might have '4: Period, 5: Antecedent'.
- The next division of that phrase ('Consequent') is at the same level of the first division ('Antecedent'), thus '5: Consequent'.
- The next phrase outside of this span returns to the initial level, so '4: Period'.
- When we eventually get to larger structural boundary then we have to find an entry at a comparable level, which will involve proportionately wider view. For the 'Recapitulation' we're starting again from the top level (so '1: Recapitulation'), while for the second subject group it's level 2 ('2: Second Subject Group').

Note that while the numbers assigned to the top levels will be consistent within and even across pieces ('Exposition', 'Development' and 'Recapitulation' will almost always appear as a set at level 1), those at lower levels will vary depending on the number of level divisions above them. For instance, the same phrase grouping structure may appear at multiple levels across a piece: at lower levels in a long and richly-structured exposition, but at nominally 'higher' levels in a short Coda with fewer divisions (see Figure 1).

This system has been designed to minimise the potential for confusion. Among the alternatives we considered and rejected was a system of multiple comparative marks with '+' for breaking up the existing span, '-' backing up one level, '=' for the same level and '*' for reverting to the highest level. This may appear to be an improvement until you try to indicate the start of a development, for instance, and have to work out the number of successive '-'s needed to reach the right level.

Further work could explore the possibility of encoding similar annotations within a formal context-free grammar. In that case, the user would specify the start of the scope of each annotation, allowing a top-down parser to infer the AST structure without the user having to explicitly specify the depth of each annotation. Informal user trials suggested this recursive mark-up format to be less intuitive and less flexible than the format proposed above, and so we have not explored it further in this initial work.

In any case, the process is necessarily somewhat complicated. To keep everything in order, some users may prefer to work directly on spreadsheets (for which instructions follow in the next section) or at least to keep an informal spreadsheet on the go at the same time.

2.3 Tabular standard

See `https://www.github.com/MarkGotham/Taking-Form` for code to extract these score markings and set them out in a tabular format that is also (perhaps even more) human-readable. Equally, some users will prefer to work directly on spreadsheets. In our experience, the tabular format provides a faster and more flexible user interface for adding, deleting, and especially for modifying annotations.

In the tabular standard:

- Generally, there is one row per measure. Exceptions include the relatively common case of 1st / 2nd time bars (for which use measure 'numbers' 112a / 112b etc.) and the extremely rare case where two structural boundaries fall within the same measure (use multiple rows).
- Each text entry indicates the start of a span. Again, we suggest using text like 'Exposition' where a

```
Sentence    Presentation    Basic Idea      [Exposition
                            Basic Idea          [Subject~I
            Continuation    Fragmentation           [Th.i 4 4]
                            Cadence                 [Th.ii 4 4 5 4 5 5 2]
                                                ]
                                                [Transition 4]
                                                [Subject~II
                                                    [Th.iii 2 4 5 4 6]
                                                    [Th.iv 4 4 2]
                                                    [Th.v 4 4 4 4 4]
                                                    [Th.vi 1 5 5]
                                                ]
                                                [Codetta 2 4]
                                            ]
```

Figure 3. An example of the tabular representation format, setting out a generic division of a sentence (level N) into presentation and continuation phases (level N+1) which, in turn, divide into two 'Basic Idea's in the first case, and 'Fragmentation' and 'Cadence' in the latter (all at level N+2).

Figure 4. An example of the tabular representation format, setting out form, theme, and phrase groupings in the exposition of Schubert 8/i (corresponding to Figure 1).

name is appropriate, and 'X' for any moment that begins a new span, but is not so easily labelled.

- That marking entry remains in effect until a change in the same column (e.g. replacing 'Exposition' with 'Development' or one 'Period' with another 'Period', for instance).
- The columns go from larger to smaller units and users may employ as many columns as needed to get from the large formal architecture (e.g. expositions of c.50 bars) down to few-bar groupings. We make no assumptions here. While a '4' grouping will often divide into 2+2, it might be better expressed as an undivided 4 (and occasionally as an asymmetric 3+1). Assuming that '4' indicates 2+2 would leave us without a description for the undivided 4.
- Again, the final levels of grouping will appear in different columns of the document, depending on the number of intervening levels.
- The 'beat' column is available for registering exactly where a new theme starts, accommodating anacruses for instance. When inputting directly to tabular, the user may wish to indicate beats with negative numbers to indicate both the basic measure range and the exact beat position at a glance. If in doubt, we advise using the downbeat ('beat 1'), especially if the accompaniment part begins there (the accompaniment is a part of the span too, after all). The code for converting to bracket representation involves a simplification that eliminates beat information.

The representation standard (in both the on-score and tabular entry systems) also accepts 'equal division', which may be useful for sections which need two labels for exactly the same span. For instance, a section might be listed as 'A' but also consist entirely of a 'Compound Period'. Similarly, the 'First theme' may very well be a 'Sentence'. Assign these designations consecutive level numbers (columns) even though they represent the same span.

Finally, while the standard accommodates any naming system, we recommend naming themes successively, and not restarting the count for the second subject group (continuing instead with 'Theme C', for instance) in order to enable succinct and unambiguous reference back to these themes later on.

2.4 Nested Bracket Notation

The linguistics standard for representing syntax trees is a system of nested brackets. Figure 4 sets out an example of this applied to musical form. In this format, the first entry in a bracket is the parent node, labelling the overall span, and each subsequent entry is a child node (an internal division). These child nodes may divide further with additional brackets recursively.

Our specification requires only:

- Matched brackets: each open '[' must be paired with a closing ']'.

There are no further, formal constraints.

Our converter outputs this as a simple string, which is all that's required to meet the syntax criteria for the various existing code libraries for visualising nested bracket notation with trees. However, once again, some users may wish to begin directly at this level, in which case we would advise one additional constraint for readability:

- Add a new level of indentation to the bracket file to indicate the structural level in question.

Figure 4 reflects this more visually graspable version.

3. CORPUS

We complement this new standard and code base with a corpus of examples. The preparation of that corpus warrants comment as a potential model for other projects of this kind.

The analyses began in the classroom, with students each assigned their own movement(s) to work on. The instructors then marked those student analyses, providing a 'corrected' version based on the student work as well as written feedback, drawing their attention to important moments and considerations. This process thus created a corpus of student-initiated, instructor-corrected analyses which went

through a final round of finessing and proof-reading to ensure grammatical accuracy and also consistency before release.

Future projects could extend this to a final round explicitly consolidating one scholar's analyses for additional internal consistency and quality control. In any case, the process benefits from involving a large number of people in suitable capacities and in a time-effective manner. Students get high-quality feedback from highly invested researcher-teachers, as well as direct integration into research projects. Researcher-teachers, in turn, benefit from knowing that their marking time is more than doubly valuable: they are providing great feedback and building a corpus at the same time.

The corpus combines tabular and nested-bracket representations to illustrate the range outlined above. The tabular representations consist of:

- Beethoven's 'first period' piano sonatas (nos1–15): 15 sonatas, 54 movements. [8]
- The first movements of all the Mozart piano sonatas (15 sonata movements).

The nested-bracket format demonstrates that format alternative, as well as other formal representations of the music. Specifically, while the tabular representations represent measures by number, these bracket representations register the number of measures within the spans in question directly (the conversion code accommodates both).

We are keen to stress that the standard supports any analyses and terminologies consistent with the minimal formal constraints itemised above. As discussed, we have elected in this corpus to name the themes alphabetically here to minimise confusion with the numbered levels, but again, this format supports any naming system the user cares to apply or devise. We provide a more miscellaneous set of repertoire and formal ideas in the corpus' 'Miscellany' folder to this effect. For instance, the analysis of the first movement from Bach's Brandenburg Concerto No.6 uses this template to set out the form in terms of changes to the canon distances.

Yet more importantly, we emphatically do not intend the analyses themselves to be in any way definitive, but rather a proof of concept, and a first offering. We welcome multiple analyses of the same work to focus our disciplinary discussion of what gives rise to formal labels and to the shifting balance of priorities. That said, creating more analyses of different works is probably a keener priority at the outset of this field. [9]

4. APPLICATIONS

The existence of tree structure representations based on formal analyse enables a number of applications in the visualisation of musical form and the comparison of form between scores. Both the visualization and the automated comparison help to focus questions of divergence between two analyses.

In this project, we have primarily focused on off-score visualizations to enable at-a-glance overviews and the development of global intuitions for structure. Equally, these formal observations could be returned to scores for different kinds of visual clarification. For instance, they enable the automated marking-up of phrase-ending barlines more strongly in scores, which is a popular annotation type among conductors. Moreover, we can toggle these annotations on and off at will: this is an important, and fundamentally digital-age flexibility.

Turning to automated comparisons, a distance metric between trees can be defined, which in turn enables clustering of scores with similar form. That process is discussed in the following section.

4.1 Tree edit distance and clustering

Zhang and Shasha [20] provide an algorithm for calculating the edit distance between two trees, t_1 and t_2. This metric is based on the minimum number of *insert*, *delete* and *update* operations (each of which have an associated cost for a given node) required to transform t_1 into t_2. We adopt their approach but allow the user to specify a function to describe the cost of each operation for a particular node, to enable the distance metric to be tuned for a particular corpus. In the examples that follow, the cost of inserting, deleting or updating a node is weighted more highly for nodes closer to the root of the tree. The cost function should also use the semantics of each node: the default comparator function in the code provided weights operations on a placeholder node '[X]' lower than the equivalent operation on any other node. Thus inserting a placeholder node '[X]' at level 6 in the tree is less costly than inserting a 'Coda' at level 1 in the tree, for example.

The concept of tree edit distance has been applied to a range of problems such as automated melody recognition, plagiarism detection in software, and finding similarities between sequences of RNA and DNA. We have explored two applications of tree edit distance, involving comparisons between two analyses, and between N analyses. We set these out in turn.

Between two analyses. A summary of the differences in tree structure up to some depth (l_{max}) can be collapsed into strings in a simple language. We base our notation on a language proposed by Cunningham for summarising control structures in programming languages. [10] Figure 5 illustrates how this simplified version of nested-bracket notation (without labels) can be used to visualise differences in form up to a given depth (l_{max}) in the tree.

In our simplified language: '||' denotes the divisions between nodes at level 1 in the tree; '.' represents a structure at level 5, or l_{max} if $l_{\text{max}} < 5$; '{' and '}' denote the start and end (respectively) of the span of a node at level 2 (if level $2 < l_{\text{max}}$); '[' and ']' denote the start and end (respectively) of the span of a node at level 3 (if

[8] Of these 15 sonatas, 9 are in 4 movements, 6 are in 3.

[9] [3]'s BPS-FH dataset of first movements from Beethoven Piano Sonatas. BPS-FH focusses on harmony but also includes some formal designations, and thus a balance between the potential benefits of cross-referencing and of extending the collective corpus.

[10] http://c2.com/doc/SignatureSurvey/

ANALYSIS 1 {..}{..}{..}{..}‖ {..}{....}{..}
ANALYSIS 2 {..}{.. .}{..}‖'{..}{....}{..}

Figure 5. Two analyses encoded as strings in a simple language. This enables edit distance-based comparison of their musical form up to level 3 in the abstract syntax tree.

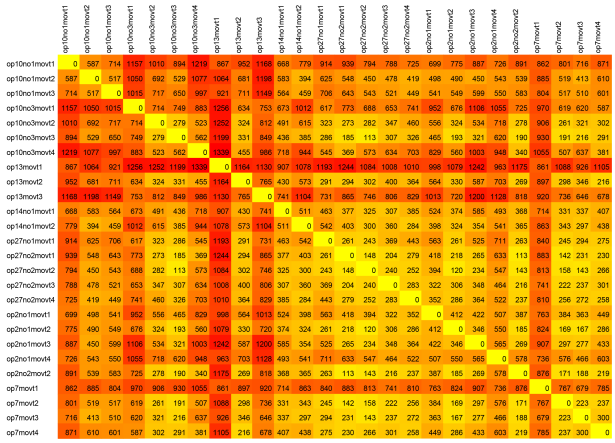


Figure 6. Heat map of tree edit distance between a selection of works in the corpus provided. The colour scale ranges from yellow (most similar/lowest tree edit distance) to red (most different/highest tree edit distance).

level $3 < l_{\max}$); '(' and ')' denote the start and end (respectively) of the span of a node at level 4 (if level $4 < l_{\max}$); ' denotes an anacrusis or longer introductory passage.

The process of calculating the tree edit distance allows correct alignment of the two strings. Our current analysis code highlights in green those nodes that are present in one analysis but not the other, and in red those nodes for which differences exist at a level greater than $l_{\max}$.

Between N analyses. Expanding this principle further, tree edit distance can be used to compare the form of all N scores in a corpus. Pairwise similarities in form can thus be identified. Figure 6 shows the tree edit distance between a selection of works in the corpus provided, visualised as a heat map.

5. SUMMARY AND OUTLOOK

This paper reports on a new specification standard for representing musical form, a corpus of example movements, and a set of code for converting between a range of representation and visualisation formats, handling score annotation to tabular or nested bracket representation, and tabular to nested bracket representation.

Among the possible improvements to this model, we envisage developments to both the user-interface and the representation structure. We consider the current opportunity to work on free and open source notation software to be a radical improvement to the student experience, enabling them to work directly on 'the music', listening back to it at will, and recording their observations in a digital format. That said, we are still limited to the availability of encoded scores and this is one musical area in which working with PDFs would be helpful, and could be practical.

As projects such as PeachNote exemplify, while OMR cannot yet reliably generate performance-ready scores, it is generally robust enough to support use cases like score-to-audio matching. Thus, as long as we still have many more PDFs than encoded scores available, it would be worth considering an alternative user-interface which offers a drag-and-drop system, for placing annotation labels onto the score. Here, we only need OMR robust enough to extract measure lines accurately and pair them with the annotation. Using pre-made (rather than free-text) labels would also solve the issue of ambiguous user-entries, though it would also limit the range of possible answers.

Turning to the representation standard, one priority for improvement is the inclusion of ambiguity. At present, we cannot admit multiple variant readings in a single file (though nothing stops analysts from producing multiple variant files). Formal judgments can arise through identification of melodic and / or harmonic closure, textural and / or dynamic changes, and much more besides. When forced into a single reading, we have to make difficult decisions between these competing priorities.

Furthermore, this ambiguity motivates a second direction for computational research based on this kind of corpus: while we can clearly pursue questions exploring proportions and the like, taking the analyses at face value, we can also explore which score elements correlate most strongly with formal designations, where analyses (dis)agree, and what gives rise to these shifts in priority.

This paper began with an overview of some recently implemented representation and visualization formats for other musical parameters; a key frontier in this field will be the combination of those standards. Given the lack of a consensus over a primary successor to XML for score encodings (MNX, MEI, ...), this much more sparsely populated field of analytical representations may be doubly slow in determining its future direction. That said, representations of harmony and form by reference to their measure and beat positions are easy to combine, perhaps representing harmonies as leaf nodes (assuming they change at least as frequently as formal sections).

Finally, we also motivated this research with its potential breadth of appeal to musicians outside the scholarly, computational community. This should be a key consideration for determining future research priorities. Integrating diverse groups in the production of corpora (also featured in this text and project) may be one way to road test new ideas for ease and popularity of use at the outset. We invite other researchers building corpora to experiment with variants on this process and whatever the exact standards they use, to focus on building up a large and versatile meta-corpus for the benefit of all.

6. ACKNOWLEDGEMENTS

This research was supported by a grant from Cornell University's Active Learning Initiative (ALI). We wish to thank the ALI, and all Cornell students who took part.

7. REFERENCES

[1] Eamonn Bell and Laurent Pugin. Approaches to handwritten conductor annotation extraction in musical scores. In *Proceedings of the 3rd International Workshop on Digital Libraries for Musicology*, DLfM 2016, pages 33–36, New York, NY, USA, 2016. ACM.

[2] C. Cannam, C. Landone, and M. Sandler. Sonic Visualiser: An open source application for viewing, analysing, and annotating music audio files. In *Proceedings of the ACM Multimedia 2010 International Conference*, pages 1467–1468, Firenze, Italy, October 2010.

[3] Tsung-Ping Chen and Li Su. Functional harmony recognition of symbolic music data with multi-task recurrent neural networks. In Emilia Gómez, Xiao Hu, Eric Humphrey, and Emmanouil Benetos, editors, *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, pages 90–97, 2018.

[4] Nicholas Cook. Methods for analysing recordings. In Nicholas Cook, Eric Clarke, Daniel Leech-Wilkinson, and John Rink, editors, *The Cambridge Companion to Recorded Music*, Cambridge Companions to Music, pages 221–245. Cambridge University Press, 2009.

[5] Johanna Devaney, Claire Arthur, Nathaniel Condit-Schultz, and Kirsten Nisula. Theme and Variation Encodings with Roman Numerals (TAVERN): A new data set for symbolic music analysis. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, pages 728–734, 2015.

[6] Johanna Devaney and Hubert Léveillé Gauvin. Representing and linking music performance data with score information. In *Proceedings of the 3rd International workshop on Digital Libraries for Musicology, DLfM 2016, New York, USA, August 12, 2016*, pages 1–8, 2016.

[7] Johanna Devaney and Hubert Léveillé Gauvin. Encoding music performance data in humdrum and MEI. *Int. J. on Digital Libraries*, 20(1):81–91, 2019.

[8] Mathieu Giraud, Richard Groult, and Emmanuel Leguy. Dezrann, a web framework to share music analysis. In Sandeep Bhagwati and Jean Bresson, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, pages 104–110, Montreal, Canada, 2018. Concordia University.

[9] Nami Iino, Mayumi Shimada, Takuichi Nishimura, Hideaki Takeda, and Masatoshi Hamanaka. Proposal of an annotation method for integrating musical technique knowledge using a GTTM time-span tree. In Ioannis Kompatsiaris, Benoit Huet, Vasileios Mezaris, Cathal Gurrin, Wen-Huang Cheng, and Stefanos Vrochidis, editors, *MultiMedia Modeling - 25th International Conference, MMM 2019, Thessaloniki, Greece, January 8-11, 2019, Proceedings, Part I*, volume 11295 of *Lecture Notes in Computer Science*, pages 616–627. Springer, 2019.

[10] Phillip B. Kirlin. A data set for computational studies of Schenkerian analysis. In *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, pages 213–218, 2014.

[11] Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music*. The MIT Press, Cambridge, MA, 1983.

[12] Alan Marsden, Satoshi Tojo, and Keiji Hirata. No longer 'somewhat arbitrary': Calculating salience in GTTM-style reduction. In *Proceedings of the 5th International Conference on Digital Libraries for Musicology*, DLfM '18, pages 26–33, New York, NY, USA, 2018. ACM.

[13] Markus Neuwirth, Daniel Harasim, Fabian C. Moss, and Martin Rohrmeier. The Annotated Beethoven Corpus (ABC): A dataset of harmonic analyses of all Beethoven string quartets. *Frontiers in Digital Humanities*, 5:16, 2018.

[14] Laurent Pugin, Rodolfo Zitellini, and Perry Roland. Verovio: A library for engraving MEI music notation into SVG. In Hsin-Min Wang, Yi-Hsuan Yang, and Jin Ha Lee, editors, *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, pages 107–112, 2014.

[15] David Rizo and Alan Marsden. A standard format proposal for hierarchical analyses and representations. In *Proceedings of the 3rd International Workshop on Digital Libraries for Musicology*, DLfM 2016, pages 25–32, New York, NY, USA, 2016. ACM.

[16] Jesse Rodin, Craig Sapp, and Clare Bokulich. Josquin research project [digital resource], 2010.

[17] Martin Rohrmeier. Towards a generative syntax of tonal harmony. *Journal of Mathematics and Music*, 5(1):35–53, 2011.

[18] Craig Stuart Sapp. Visual hierarchical key analysis. *Computers in Entertainment (CIE)*, 3(4):1–19, October 2005.

[19] Dmitri Tymoczko, Mark Gotham, Michael Scott Cuthbert, and Christopher Ariza. The Romantext format: a flexible and standard method for representing Roman numeral analyses. In *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands*, 2019.

[20] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems *SIAM journal on computing*, 18(6):1245–1262, 1989.

LEARNING COMPLEX BASIS FUNCTIONS FOR INVARIANT REPRESENTATIONS OF AUDIO

Stefan Lattner,[1] **Monika Dörfler,**[2] **Andreas Arzt**[3]

[1] Sony Computer Science Laboratories (CSL), Paris, France
[2] NuHAG, Faculty of Mathematics, University Vienna
[3] Institute of Computational Perception, JKU Linz

ABSTRACT

Learning features from data has shown to be more successful than using hand-crafted features for many machine learning tasks. In music information retrieval (MIR), features learned from windowed spectrograms are highly variant to transformations like transposition or time-shift. Such variances are undesirable when they are irrelevant for the respective MIR task. We propose an architecture called Complex Autoencoder (CAE) which learns features invariant to orthogonal transformations. Mapping signals onto complex basis functions learned by the CAE results in a transformation-invariant "magnitude space" and a transformation-variant "phase space". The phase space is useful to infer transformations between data pairs. When exploiting the invariance-property of the magnitude space, we achieve state-of-the-art results in audio-to-score alignment and repeated section discovery for audio. A PyTorch implementation of the CAE, including the repeated section discovery method, is available online. [1]

1. INTRODUCTION

Learning from audio data most commonly involves some prior processing of the raw sound signals. The most popular features are derived from a spectrogram, which consists of the magnitude values of the Fourier transform of a windowed signal of interest. In a Fourier transform, a signal is projected onto sine and cosine functions of different frequencies. One of the main reasons for the spectrogram to be more useful than the usage of the Fourier coefficients in their complex form is the fact that the magnitude spectrum of a signal is invariant to a translation of the original signal.

This invariance to translation, desirable for most learning problems in audio, results from the fact that cosine and sine represent the real and imaginary parts, respectively, of the *complex eigenvectors* of translation. More generally, the eigenvectors of an orthogonal transformation (e.g., translation, rotation, reflection, but most general all permutations - "shuffling pixels") constitute an orthonormal basis of complex vectors with corresponding eigenvalues of magnitude 1. Hence, as we shall see in detail, the absolute value of a signal's coefficients with respect to this basis is invariant to that transformation. We harness this invariance property for learning representations invariant to different orthogonal transformations.

In particular, transposition-invariance is an essential property for several MIR tasks, including alignment tasks, repeated section discovery, classification tasks, cover song detection, query by humming, or representations of acapella recordings with pitch drift. Different methods have aimed at learning transposition-invariant representations. For example, in [33] close time steps in chromagrams are cross-correlated in order to calculate distances between pitch classes, and in [20], successive n-grams of constant-Q transformed (CQT) representations of audio are compared using a Gated Autoencoder (GAE) architecture. Most similar to our approach, the transposition-invariant magnitudes of Fourier transformations applied to chromagram-like representations of audio are facilitated in [3] and [23]. However, instead of using 2D Fourier transforms with fixed basis functions, we learn the relevant basis functions starting from CQT representations of audio. This learning of basis functions has some advantages over using pre-defined bases. For example, the number of basis elements necessary to discriminate between signals can be reduced compared to a common Fourier transform (e.g., for transposition- and time-shift invariance we use $M = 256$ basis functions for input dimensionality $N = 3840$, while usually $N = M$). Furthermore, our approach is generic and has the potential to learn other musically interesting invariances (e.g., towards tempo-change, diatonic transposition, inversion, or retrograde).

The contribution of this paper is a simple training method for learning invariant representations from data pairs and its application to two MIR tasks. First, we show that when using the features learned by the Complex Autoencoder (CAE) from audio in CQT representation, we can improve the state-of-the-art in a transposition-invariant repeated section discovery task in audio. Second, the CAE

[1] https://github.com/SonyCSLParis/cae-invar

features prove useful in an audio-to-score alignment task, where we show that most of the time, they yield better results than Chroma features and features calculated with a GAE. We also compare the CAE with a GAE in classifying rotated MNIST digits, based on rotation-invariant features learned by the CAE. The reason we also perform experiments on MNIST is that it allows us to show the efficacy of the model with respect to rotation-invariance. Furthermore, the class labels available in the MNIST dataset help to highlight the different clusters in the rotation-invariant space (see Figure 4).

In particular for translations, the model can be interpreted as measuring distances in the data. Training the CAE for transposition and time-shift invariance on short windows of audio in CQT representation, therefore, leads to representations of rhythmic structures and tonal relationships present in the windows, what we exploit in the repeated section discovery task. Representing rhythmic structures is less critical in music alignment tasks; it can even be disadvantageous when the aligned signals differ in tempo. We show in the alignment task that it is sufficient to train the CAE only for transposition-invariance (i.e., time-shift transformation) on rather short n-grams of audio in CQT representation. This is because compared to repeated section discovery, where rhythmic patterns can help to identify similar parts, in the alignment task, a dynamic time-warping algorithm keeps track of the respective positions in the music pieces.

The CAE can be trained in an unsupervised manner on data pairs obeying the relevant transformations. Thereby, we obtain a "magnitude space" and a "phase space", as it is known from a Fourier transform. The "magnitude space" of the CAE is invariant to all the learned transformations. Remarkably, the *phase shifts* a projected signal undergoes during a transformation (i.e., the relative vector in the "phase space" of the CAE) are discriminative with respect to the type and the distance of a transformation. This is an interesting property which could be exploited for determining types of relations between musical fragments in structure analysis tasks.

The paper is structured as follows. In Section 2 existing work related to the proposed method is discussed. In Section 3 we describe the model and its mathematical background and Section 4 describes the general training procedure. In Section 5, we show results on three different tasks: discovery of repeated themes and sections, audio-to-score alignment, and classification of MNIST digits. We end the paper with a conclusion and a discussion of possible directions for future work (Section 6).

2. RELATED WORK

Generally, mid-level representations in neural networks are highly variant to transformations in the input. The most common and well-known way to obtain shift-invariance in convolutional architectures is max-pooling [4]. However, full shift-invariance can only be achieved step-wise by applying max-pooling over several layers. A whole line of research therefore aims to obtain representations invariant to different kinds of transformations using other approaches. Inspiration for the proposed model was drawn from [25], where complex basis functions are learned using a GAE. An approach similar to ours is to facilitate harmonic functions or wavelets, either in weight initialization [29, 37], for modulating learned filters using Gabor functions [22], or for using fixed wavelets in scattering transforms [5, 32]. Similarly, harmonic functions can be pre-defined, e.g., to obtain rotation invariance in convolutional architectures [36], or learned, e.g., by assuming "temporal slowness" of features in videos [21, 28], while pitch-invariant timbral features are learned in [30] by enabling convolution through the frequency domain.

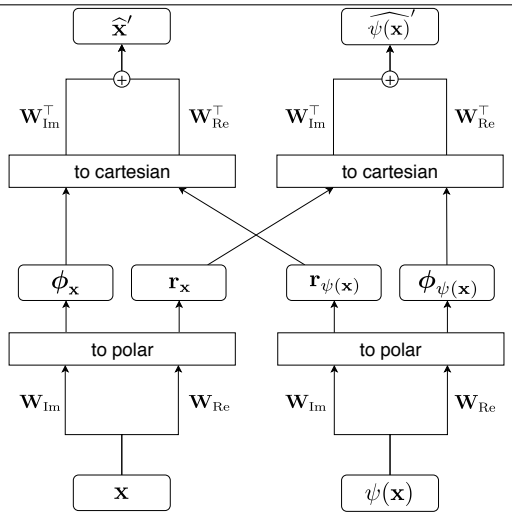


Figure 1: Schematic illustration of reconstruction during training. Both the input $\mathbf{x}$ and its transformed counterpart $\psi(\mathbf{x})$ are projected onto complex basis pairs $\{\mathbf{W}_{\mathrm{Re}}, \mathbf{W}_{\mathrm{Im}}\}$ and expressed in polar form. Then, using the swapped magnitude vectors $\{\mathbf{r}_{\psi(\mathbf{x})}, \mathbf{r}_{\mathbf{x}}\}$ and the original phase vectors $\{\boldsymbol{\phi}_{\mathbf{x}}, \boldsymbol{\phi}_{\psi(\mathbf{x})}\}$, the data is reconstructed by performing the inverse operations.

Most of the approaches mentioned so far (including our approach) aim at invariances to relatively simple, affine transformations. Invariances to more complex, non-linear transformations are usually achieved by redundancy (e.g., an object is presented from different camera angles or under different lighting conditions), which typically requires bigger architectures. That way, invariance can be learned by an explicit transformation of the input [16], by enforcing similarity in the latent space [24], or by using a Siamese architecture and pre-defined transformation sets [18]. Other methods involve rotating convolution kernels during training [38] and dealing with input deformations using learned, dynamic convolution grids [8]. In [9] an end-to-end CNN which acts on raw audio learns Gabor-like filters similar to those extracted by the CAE, see Figure 2.

3. MODEL AND MATHEMATICAL BACKGROUND

We aim at learning orthogonal transformations encoding certain invariances of a class of signals which are known or

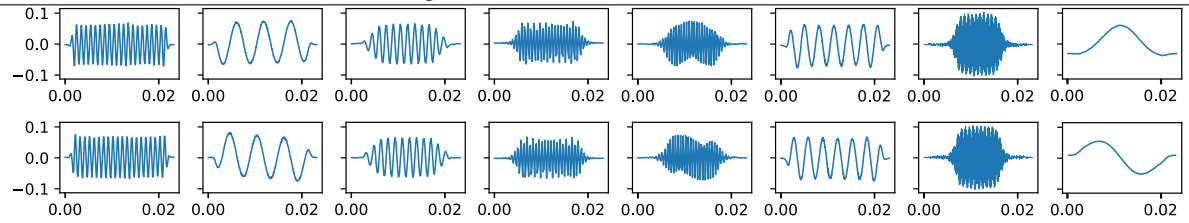


Figure 2: Some examples of real (top) and imaginary (bottom) basis vectors learned from audio signals (time in seconds).

assumed to be useful for a particular learning task at hand. To this end, we leverage the particular properties of orthogonal transformations, which we now describe. A transformation $\psi : \mathbb{R}^N \to \mathbb{R}^N$ is orthogonal if $\langle \psi(\mathbf{x}), \psi(\mathbf{y}) \rangle = \langle \mathbf{x}, \mathbf{y} \rangle$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$. By $\langle \mathbf{x}, \mathbf{y} \rangle$ we denote the inner product on $\mathbb{R}^N$ or $\mathbb{C}^N$, respectively. Orthogonal transformations are distinguished by the fact that they possess a diagonalization with eigenvalues which all have absolute value 1. Hence, in any non-trivial case, the eigenvalues are complex and so are the corresponding eigenvectors. More precisely, if ψ is orthogonal, there exists a unitary matrix [2] $\mathbf{W}$ and eigenvalues λ_j, $j = 1, \ldots, N$, with $|\lambda_j| = 1$ for all j, such that

$$\psi(\mathbf{x}) = \mathbf{W}^* \mathbf{D} \mathbf{W} \mathbf{x} \tag{1}$$

Here $\mathbf{D}$ denotes the diagonal $N \times N$ matrix with the eigenvalues λ_j in the diagonal. We hence have the following statement.

Proposition 1. *If an orthogonal transformation $\psi : \mathbb{R}^N \to \mathbb{R}^N$ is diagonalised by a unitary matrix $\mathbf{W}$, then the feature vector given by $|\mathbf{W}\mathbf{x}|$ for all $\mathbf{x} \in \mathbb{R}^N$ is invariant to ψ. In other words, we have $|\mathbf{W}\mathbf{x}| = |\mathbf{W}\psi(\mathbf{x})|$ for all $\mathbf{x} \in \mathbb{R}^N$.*

Proof. According to (1) and since $\mathbf{W}\mathbf{W}^*\mathbf{x} = \mathbf{x}$, we have

$$\mathbf{W}\psi(\mathbf{x}) = \mathbf{D}\mathbf{W}\mathbf{x}, \tag{2}$$

which can be written coordinate-wise as

$$\langle w_j, \psi(\mathbf{x}) \rangle = \lambda_j \langle w_j, \mathbf{x} \rangle, \; j = 1, \ldots, N,$$

where w_j denotes the $j-$th row of the complex, unitary matrix $\mathbf{W}$. Hence, since $|\lambda_j| = 1$ for all j, we have $|\langle w_j, \psi(\mathbf{x}) \rangle| = |\langle w_j, \mathbf{x} \rangle|$ for all $\mathbf{x} \in \mathbb{R}^N$ and thus $|\mathbf{W}\mathbf{x}| = |\mathbf{W}\psi(\mathbf{x})|$ as claimed. □

In CAE-learning, we can only deal with real weights and are usually interested in learning less than N basis vectors. Hence, we split an $M \times N$-dimensional submatrix of the unitary, complex matrix of eigenvectors $\mathbf{W}$ into a real and an imaginary part $\mathbf{W}_{\text{Re}} \in \mathbb{R}^{M \times N}$ and $\mathbf{W}_{\text{Im}} \in \mathbb{R}^{M \times N}$ which map $\mathbf{x}$ onto the real and imaginary part of $\mathbf{W}\mathbf{x}$, respectively. The complex data $\mathbf{W}_{\text{Re}}\mathbf{x} + i\mathbf{W}_{\text{Im}}\mathbf{x}$ is then expressed in its polar form by a phase vector $\boldsymbol{\phi}_{\mathbf{x}} \in [0, 2\pi)^M$ and a magnitude vector $\mathbf{r}_{\mathbf{x}} \in \mathbb{R}^M_{\geq 0}$.

According to Proposition 1, assuming that $\mathbf{W}_{\text{Re}} + i\mathbf{W}_{\text{Im}}$ consist of orthonormal eigenvectors of ψ leads to the magnitude of projections of a signal $\mathbf{x}$ and a transformed version $\psi(\mathbf{x})$ onto these eigenvectors to be equal. This property is imposed during training by expressing $\mathbf{W}\mathbf{x}$ and $\mathbf{W}\psi(\mathbf{x})$ in their respective polar forms

$$\boldsymbol{\phi}_{\mathbf{x}} = \text{atan2}(\mathbf{W}_{\text{Re}}\mathbf{x}, \mathbf{W}_{\text{Im}}\mathbf{x}), \tag{3}$$

and

$$\mathbf{r}_{\mathbf{x}} = \sqrt{(\mathbf{W}_{\text{Re}}\mathbf{x})^2 + (\mathbf{W}_{\text{Im}}\mathbf{x})^2}, \tag{4}$$

and swapping the magnitude vectors before reconstruction.

Accordingly, we reconstruct $\mathbf{x}$ given its own phase representation $\boldsymbol{\phi}_{\mathbf{x}}$ and the magnitude representation of the transformed signal $\mathbf{r}_{\psi(\mathbf{x})}$ as follows (see Figure 1):

$$\widehat{\mathbf{x}}' = \mathbf{W}_{\text{Re}}^\top(\mathbf{r}_{\psi(\mathbf{x})} \cdot \sin \boldsymbol{\phi}_{\mathbf{x}}) + \mathbf{W}_{\text{Im}}^\top(\mathbf{r}_{\psi(\mathbf{x})} \cdot \cos \boldsymbol{\phi}_{\mathbf{x}}). \tag{5}$$

Likewise, we reconstruct the transformed signal $\psi(\mathbf{x})$ as

$$\widehat{\psi(\mathbf{x})}' = \mathbf{W}_{\text{Re}}^\top(\mathbf{r}_{\mathbf{x}} \cdot \sin \boldsymbol{\phi}_{\psi(\mathbf{x})}) + \mathbf{W}_{\text{Im}}^\top(\mathbf{r}_{\mathbf{x}} \cdot \cos \boldsymbol{\phi}_{\psi(\mathbf{x})}). \tag{6}$$

The CAE is then trained by minimizing the symmetric reconstruction error

$$\frac{1}{N}\sum_i^N (x_i - \widehat{x}'_i)^p + \frac{1}{N}\sum_j^N (\psi(x_j) - \widehat{\psi(x_j)}')^p, \tag{7}$$

where $p \in \{1, 2\}$ has shown to work well in practice. Training on sufficiently many transform pairs thus leads to learning the weights of the unitary matrix $\mathbf{W}$, which diagonalises ψ. While the magnitudes of the coefficient vectors $\mathbf{W}\psi(\mathbf{x})$ are equal to $\mathbf{W}\mathbf{x}$, the transformation itself is then represented by the differences in the phase vectors $\Delta\boldsymbol{\phi} = \boldsymbol{\phi}_{\mathbf{x}} - \boldsymbol{\phi}_{\psi(\mathbf{x})}$ (see Figure 4(b)). As an example of complex basis vectors learned by the CAE, see Figure 2, where the CAE was trained on time-shifted audio signals in the time domain, yielding complex Gabor-like filters.

4. TRAINING

For all the experiments described below, we choose 256 complex basis vectors and train the model for 500 epochs with a learning rate of 1e-3. We use a batch size of 1000, and we sample 100k transformations per epoch, generally picking random instances from the train set to be transformed. The training data is standardized, and 50% dropout is used on the input. We set $p = 1$ (see Equation 7) for the audio experiments, and $p = 2$ for the MNIST experiment. In the alignment experiment, we also penalize the mean of norms of all basis vectors and the deviation of the individual basis vectors' norms to the average norm over all basis vectors. In the MNIST experiment, the norm of all basis vectors is set to 0.4 after every batch. For information about the training data see the respective experiment section below.

[2] A complex matrix $\mathbf{W}$ is unitary, if $\mathbf{W}^* = \mathbf{W}^{-1}$.

Algorithm	F_{est}	P_{est}	R_{est}	$F_{\text{o(.5)}}$	$P_{\text{o(.5)}}$	$R_{\text{o(.5)}}$	$F_{\text{o(.75)}}$	$P_{\text{o(.75)}}$	$R_{\text{o(.75)}}$	$\mathbf{F_3}$	P_3	R_3	Time (s)
CA (ours)	52.53	63.10	50.29	63.58	64.60	62.68	67.20	68.73	65.82	**52.16**	**62.51**	49.78	**69**
GAE intervals [20]	**57.67**	**67.46**	59.52	58.85	61.89	56.54	68.44	72.62	64.86	51.61	59.60	**55.13**	194
VMO deadpan [34]	56.15	66.80	57.83	**67.78**	**72.93**	**64.30**	**70.58**	**72.81**	**68.66**	50.60	61.36	52.25	96
SIARCT-CFP [7]	23.94	14.90	**60.90**	56.87	62.90	51.90	-	-	-	-	-	-	-
Nieto [27]	49.80	54.96	51.73	38.73	34.98	45.17	31.79	37.58	27.61	32.01	35.12	35.28	454

Table 1: Different precision, recall and f-scores (adopted from [34], details on the metrics are given in [6]) of different methods in the Discovery of Repeated Themes and Sections MIREX task, for symbolic music and audio. The $\mathbf{F_3}$ score constitutes a summarization of all metrics.

5. EXPERIMENTS

5.1 Discovery of Repeated Themes and Sections

In the MIREX task "Discovery of Repeated Themes and Sections", [3] the performance of different algorithms to identify repeated (and possibly transposed) patterns in symbolic music and audio is tested. The commonly used JKUPDD dataset [6] contains 26 motifs, themes, and repeated sections annotated in 5 pieces by J. S. Bach, L. v. Beethoven, F. Chopin, O. Gibbons, and W. A. Mozart. We use the audio versions of the dataset and preprocess them the same way as the training data described below.

The CAE is trained on 100 random piano pieces of the MAPS dataset [12] (subset MUS) at a sampling rate of 22.05 kHz. We choose a constant-Q transformed spectrogram representation with a hop size of 1984. The range comprises 120 frequency bins (24 per octave), starting from a minimal frequency of 65.4 Hz. The spectrogram is split into n-grams of 32 frames. The set of transformations applied to the data during training $\Psi_{\text{pshift, tshift}}$ contains transposition by $[-24, 24]$ frequency bins and time shifts by $[-12, 12]$.

After training, all n-grams of the JKUPDD dataset are projected into the transformation-invariant magnitude space. Using these representations, a self-similarity matrix is built for each piece using the reciprocal of the cosine distance. The matrices are then filtered with an identity matrix of size 10×10. Then, their main diagonals are set to zero. Finally, the matrices are first normalized and then centered by subtracting their medians.

For finding repeated sections, the method proposed in [20] is adopted, which finds diagonals in a self-similarity matrix using a threshold. As we normalized the matrices to zero median, the threshold chosen in this experiment is close to zero (i.e., 0.01).

5.1.1 Results and Discussion

Table 1 shows the results of the experiment. Using our method, we could slightly outperform the Gated Autoencoder approach proposed in [20]. By visual inspection of the self-similarity matrix, we noted very precise diagonals at repetitions, while almost no similarity is indicated on other parts (this is different from the self-similarity plots provided in [20]). This selectivity, which may also result from the cosine distance, probably contributes to the slightly higher precision of the proposed method.

[3] `http://www.music-ir.org/mirex/wiki/2017:Discovery_of_Repeated_Themes_&_Sections`

ID	Dataset	Files	Duration
CE	Chopin Etude	22	$\sim$ 30 min.
CB	Chopin Ballade	22	$\sim$ 48 min.
MS	Mozart Sonatas	13	$\sim$ 85 min.
RP	Rachmaninoff Prelude	3	$\sim$ 12 min.
B3	Beethoven 3	4	$\sim$ 52 min.
M4	Mahler 4	4	$\sim$ 58 min.

Table 2: The evaluation data set for the alignment experiments (see text).

5.2 Invariant Audio-to-Score Alignment

The task of synchronising an audio recording of a music performance and its score has already been studied extensively (see e.g. [10, 11, 14, 15, 17, 26]). Here, we compare synchronisation results using the proposed method (CAE) to traditional Chroma features and the GAE features introduced in [20], which were used for music alignment in [2].

For the alignment experiments we follow [2], using the same setup and the same data (see Table 2 for a summary). *CB* and *CE* consist of 22 recordings of excerpts of the Ballade Op. 38 No. 1 and the Etude Op. 10 No. 3 by Chopin [13], *MS* contains performances of the first movements of the piano sonatas KV279-284, KV330-333, KV457, KV475 and KV533 by Mozart [35], and *RP* consists of three performances of the Prelude Op. 23 No. 5 by Rachmaninoff [1]. Finally, *B3* and *M4* are annotated recordings of Beethoven's 3^{rd} and Mahler's 4^{th} symphonies. Note that CB, CE, MS, and RP consist of piano music, while B3 and M4 consist of orchestral music, but we will use the same model for the whole data set, which was trained on piano music only.

The scores are provided in the MIDI format, with the global tempi set such that the scores roughly match the average length of the given performances, i.e., both representations have the same average tempo, but there still exist substantial differences in local tempi. The scores are then synthesized with the help of *timidity* [4] and a publicly available sound font. The resulting audio files are used as score representations for the alignment experiments. To compute the alignments, a multi-scale variant of the dynamic time warping (DTW) algorithm (see [26] for a detailed description of DTW) is used, namely FastDTW [31] with the radius parameter set to 50.

The CAE is trained the same way and on the same data as described in Section 5.1 but here we choose a CQT hop size of 448. Furthermore, for this experiment, we use an

[4] `https://sourceforge.net/projects/timidity/`

		'Un-transposed' Data			Transp.
DS	**Metric**	**Chroma**	**GAE**	**CAE**	**CAE**
CB	1st Quartile	15 ms	**10 ms**	**10 ms**	10 ms
	Median	34 ms	22 ms	**21 ms**	21 ms
	3rd Quartile	80 ms	39 ms	**37 ms**	38 ms
	Err. $\leq$ 50 ms	64%	83%	**84%**	84%
	Err. $\leq$ 250 ms	85%	**94%**	**94%**	94%
CE	1st Quartile	13 ms	**10 ms**	**10 ms**	9 ms
	Median	29 ms	21 ms	**19 ms**	18 ms
	3rd Quartile	56 ms	36 ms	**32 ms**	30 ms
	Err. $\leq$ 50 ms	71%	87%	**90%**	91%
	Err. $\leq$ 250 ms	94%	**96%**	**96%**	97%
MS	1st Quartile	7 ms	**6 ms**	**6 ms**	6 ms
	Median	16 ms	13 ms	**12 ms**	12 ms
	3rd Quartile	31 ms	25 ms	**22 ms**	22 ms
	Err. $\leq$ 50 ms	85%	90%	**91%**	92%
	Err. $\leq$ 250 ms	98%	**100%**	**100%**	99%
RP	1st Quartile	17 ms	14 ms	**9 ms**	9 ms
	Median	43 ms	34 ms	**20 ms**	21 ms
	3rd Quartile	113 ms	90 ms	**55 ms**	69 ms
	Err. $\leq$ 50 ms	55%	63%	**74%**	70%
	Err. $\leq$ 250 ms	91%	90%	**95%**	93%
B3	1st Quartile	20 ms	25 ms	**17 ms**	18 ms
	Median	48 ms	54 ms	**39 ms**	42 ms
	3rd Quartile	108 ms	104 ms	**83 ms**	99 ms
	Err. $\leq$ 50 ms	52%	47%	**59%**	56%
	Err. $\leq$ 250 ms	88%	90%	**91%**	88%
M4	1st Quartile	46 ms	50 ms	**42 ms**	46 ms
	Median	110 ms	129 ms	**99 ms**	110 ms
	3rd Quartile	278 ms	477 ms	**255 ms**	290 ms
	Err. $\leq$ 50 ms	27%	25%	**29%**	27%
	Err. $\leq$ 250 ms	73%	66%	**75%**	72%

Table 3: Comparison of the proposed features CAE to Chroma features and features computed via a gated autoencoder GAE. The first three columns show results on normal, i.e., un-transposed data. The rightmost column shows the average result of alignments of the original performances to scores in 12 different transpositions.

n-gram size of 8. The set of transformations applied to the data during training Ψ_{pshift} are transpositions by $[-24, 24]$ frequency bins.

5.2.1 Results and Discussion

In the alignment experiments, we compare the proposed CAE features to the results presented in [2], where Chroma features and features computed via a gated autoencoder (GAE) were compared to each other. Table 3 gives an overview of the results. The first three columns show that the proposed CAE features consistently outperform the other two methods in the normal alignment setting (i.e., without any transpositions). Additionally, the rightmost column shows that for CAE, the results essentially stay the same, even when the alignment is computed with transposed versions of the score. This demonstrates the invariance to transpositions, which is a serious advantage over the Chroma features.

As has been shown in [2], the GAE features are highly sensitive to tempo differences between the score representation and the performance. To see if the proposed CAE features suffer from the same problem, we repeated this experiment and performed alignments on artificially slowed-down and sped-up score representations. The results are shown in Table 4. For all tested features, the degree to which the tempi of the score representation and the performance match influences the alignment quality. The experiments suggest that CAE is less sensitive to differences in tempi than GAE, but the Chroma features still have the advantage over GAE in this matter. We also conducted experiments with more extreme tempi, which further confirmed this trend. The reason for the higher robustness to tempo differences of the CAE features over the GAE features may be found in the way the GAE features are computed. In a GAE, two inputs $\{\mathbf{x}_{t-n,\dots,t}, \mathbf{x}_{t+1}\}$ are compared to one another, and the features are sensitive to the position and order of events in $\mathbf{x}_{t-n,\dots,t}$. When training a CAE only for transposition-invariance, the resulting features represent mainly distances in the frequency-dimension of the input and tend to be invariant to the position of events in time.

5.3 Classification of MNIST digits

We test the ability of the CAE to learn rotation-invariance in 2D images using randomly rotated MNIST digits (the dataset was first described in [19]). Given the set of rotations Ψ_{rot} with rotation angles $[0, 2\pi)$ about the origin of the images. For any MNIST instance $\mathbf{x}_k$, we create a rotated version $\psi_i(\mathbf{x}_k)$ and a further rotated version $\psi_j(\psi_i(\mathbf{x}_k))$, where $\psi_i, \psi_j \in \Psi_{\text{rot}}$, resulting in pairs $\{\psi_i(\mathbf{x}_k), \psi_j(\psi_i(\mathbf{x}_k))\}$. After the CAE is trained on 50k such pairs, single randomly rotated instances are projected into the magnitude space. On these projections, a logistic regression classifier is trained to predict the class labels. We test different train set sizes (sampled from the main train set with balanced class distribution). 50-fold cross-validation is used, where evaluation is always performed on 10k test instances, independent of the train set size. For comparison, we perform k-nn classification on the randomly rotated images (i.e., the input space), and unrotated images directly. We choose logistic regression for the magnitude space and k-nn classification for the input space because they showed the overall best results for those representations. This choice, as well as the overall experiment setup, reflects that in [25].

5.3.1 Results and Discussion

Figure 3 shows the results of the rotated MNIST classification task. The error rates of *magnitudes GAE* were obtained by a GAE architecture which was extended to learn basis functions, as reported in [25]. Classification on the magnitude space of the CAE (*magnitudes CAE*) leads to substantially better results than those of the GAE, even though only 256 basis elements are used in the CA, compared to 1000 in the GAE. This is probably due to the explicit training of the CA to learn an invariant magnitude space, while the magnitude space of the GAE is learned indirectly during the learning of transformations. Overall, classification on the rotation-invariant magnitude spaces performs much better than on the input space of rotated images in particular for small train set sizes. The difference in performance between *images (original)* and *magnitudes CA* reflects the gap between a theoretically optimal rotation-invariant representation (as *images (original)* are

DS	Metric	Chroma			GAE			CAE		
		$\frac{2}{3}$ T.	Base T.	$\frac{4}{3}$ T.	$\frac{2}{3}$ T.	Base T.	$\frac{4}{3}$ T.	$\frac{2}{3}$ T.	Base T.	$\frac{4}{3}$ T.
CB	Error $\leq$ 50 ms	54%	64%	67%	47%	83%	33%	**80%**	**84%**	**85%**
	Error $\leq$ 250 ms	82%	85%	85%	87%	**94%**	84%	**91%**	**94%**	**94%**
CE	Error $\leq$ 50 ms	69%	71%	73%	40%	87%	38%	**85%**	**90%**	**88%**
	Error $\leq$ 250 ms	90%	94%	94%	**93%**	**96%**	80%	**93%**	**96%**	**95%**
MS	Error $\leq$ 50 ms	79%	85%	75%	84%	90%	74%	**86%**	**91%**	**76%**
	Error $\leq$ 250 ms	98%	98%	97%	**99%**	**100%**	**98%**	**99%**	**100%**	**98%**
RP	Error $\leq$ 50 ms	53%	55%	56%	43%	63%	37%	**67%**	**74%**	**63%**
	Error $\leq$ 250 ms	92%	91%	87%	82%	90%	85%	**95%**	**95%**	**91%**
B3	Error $\leq$ 50 ms	**44%**	52%	**36%**	–	47%	–	39%	**59%**	33%
	Error $\leq$ 250 ms	**83%**	88%	**82%**	–	90%	–	82%	**91%**	**82%**
M4	Error $\leq$ 50 ms	**26%**	27%	**24%**	–	25%	–	24%	**29%**	22%
	Error $\leq$ 250 ms	**75%**	73%	**71%**	–	66%	–	72%	**75%**	65%

Table 4: Results on score representations with different tempi (higher is better). *Base T.* refers to a globally set tempo that ensures that the duration of the score representation is roughly equal to the duration of a typical performance. $\frac{2}{3}$ *T.* and $\frac{4}{3}$ *T.* refer to score representation with the tempo set to $\frac{2}{3}$ and $\frac{4}{3}$ of the base tempo. The two metrics used are the percentage of events that are aligned with an error lower or equal 50 ms and 250 ms (i.e. higher is better). The missing numbers for GAE were not provided in [2].

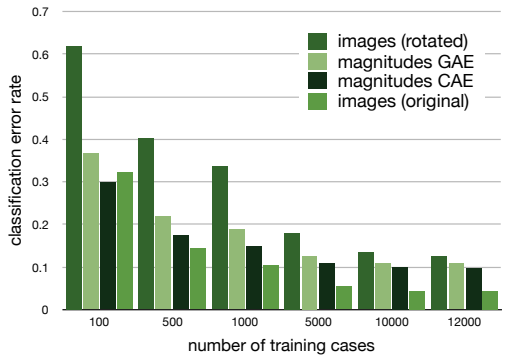
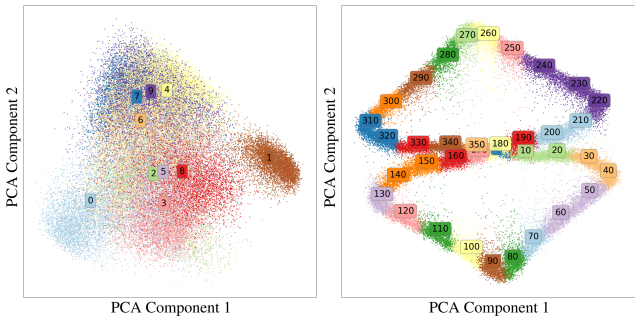


Figure 3: Classification error rates in the input space (images) and the magnitude space (magnitudes) on the rotated MNIST dataset with different train set sizes. "Images (original)" denotes the results of the unrotated MNIST dataset for comparison.



(a) (b)

Figure 4: PCAs of rotated MNIST digits in the magnitude space (a) and the phase-difference space (b) (best viewed in color). The magnitude space represents the data in the absence of the transformations leading to clusters of the digit classes (colored and labeled accordingly). The phase-difference space represents the transformations between images, independent of their identity (colors and labels denote rotation angles quantized into 36 bins).

not rotated), and the representations learned by the CA. On 100 training cases, logistic regression would outperform the k-nn classification on the input spaces, while for all other train set sizes k-nn is superior over logistic regression. Thus, we obtain slightly worse classification performance of *images (original)* on 100 training cases compared to *magnitudes CA*.

Figure 4 shows PCAs of the randomly rotated MNIST digits projected into the magnitude space and the phase-difference space of the CAE. The clusters in the magnitude space indicate that images with the same content (i.e., class label) yield similar projections, independent of their rotations. The clusters in the phase-difference space show that phase differences clearly represent the transformations in the data.

6. CONCLUSION AND FUTURE WORK

The empirical results in this work show that for music alignment, structure analysis, and invariant classification tasks, the features learned by the CAE have advantages over other features, like Chroma features, and features learned by a GAE. As opposed to Chroma features, the CAE features are transposition-invariant, and generally perform better in the alignment task. Compared to the features learned by a GAE, the CAE features are more robust to differences in tempo between alignment data.

Future work should involve investigating the use of the "phase-difference" space of the CAE. For example, qualifying transformations between sections in music could lead to a richer musical structure analysis (e.g., determining mutually transposed parts, or finding sections with similar rhythm but different tonality).

The learned bases could also be used in scattering transforms (i.e., as convolutional filters). As opposed to conventional scattering transforms, where the bases are fixed in general, learned bases may help reducing model sizes or to cover different invariances. Using rotation-invariant filters in convolutional settings may lead to rotation-invariant architectures, similar to what is proposed in [36].

7. ACKNOWLEDGMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skodowsa-Curie grant agreement No. 765068. Monika Dörfler is supported by the Vienna Science and Technology Fund (WWTF) project SALSA (MA14-018).

8. REFERENCES

[1] Andreas Arzt. *Flexible and Robust Music Tracking*. PhD thesis, Johannes Kepler University Linz, 2016.

[2] Andreas Arzt and Stefan Lattner. Audio-to-score alignment using transposition-invariant features. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Paris, France, 2018.

[3] Thierry Bertin-Mahieux and Daniel P. W. Ellis. Large-scale cover song recognition using the 2d fourier transform magnitude. In Fabien Gouyon, Perfecto Herrera, Luis Gustavo Martins, and Meinard Müller, editors, *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*, pages 241–246. FEUP Edições, 2012.

[4] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 111–118. Omnipress, 2010.

[5] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1872–1886, 2013.

[6] Tom Collins. Discovery of repeated themes and sections. `http://www.music-ir.org/mirex/wiki/2017:Discovery_of_Repeated_Themes_%26_Sections`, 2017.

[7] Tom Collins, Andreas Arzt, Sebastian Flossmann, and Gerhard Widmer. Siarct-cfp: Improving precision and the discovery of inexact musical patterns in point-set representations. In *ISMIR*, pages 549–554, 2013.

[8] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 764–773. IEEE Computer Society, 2017.

[9] S. Dieleman and B. Schrauwen. End-to-end learning for music audio. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968, May 2014.

[10] Simon Dixon and Gerhard Widmer. MATCH: A music alignment tool chest. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 492–497, London, UK, 2005.

[11] Daniel P.W. Ellis and Graham E. Poliner. Identifying 'cover songs' with chroma features and dynamic programming beat tracking. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 4, pages 1429–1432, Honolulu, Hawaii, USA, 2007.

[12] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010.

[13] Werner Goebl. The vienna 4x22 piano corpus, 1999. `http://dx.doi.org/10.21939/4X22`.

[14] Maarten Grachten, Martin Gasser, Andreas Arzt, and Gerhard Widmer. Automatic alignment of music performances with structural differences. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 607–612, Curitiba, Brazil, 2013.

[15] Ning Hu, Roger B. Dannenberg, and George Tzanetakis. Polyphonic audio matching and alignment for music retrieval. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA, 2003.

[16] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2017–2025, 2015.

[17] Cyril Joder, Slim Essid, and Gaël Richard. A comparative study of tonal acoustic features for a symbolic level music-to-score alignment. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Dallas, Texas, USA, 2010.

[18] Dmitry Laptev, Nikolay Savinov, Joachim M. Buhmann, and Marc Pollefeys. TI-POOLING: transformation-invariant pooling for feature learning in convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 289–297. IEEE Computer Society, 2016.

[19] Hugo Larochelle, Dumitru Erhan, Aaron C. Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In Zoubin Ghahramani, editor, *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, volume 227 of *ACM International Conference Proceeding Series*, pages 473–480. ACM, 2007.

[20] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. Learning transposition-invariant interval features from symbolic music and audio. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27*, 2018.

[21] Quoc V. Le, Will Y. Zou, Serena Y. Yeung, and Andrew Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, pages 3361–3368. IEEE Computer Society, 2011.

[22] Shangzhen Luan, Chen Chen, Baochang Zhang, Jungong Han, and Jianzhuang Liu. Gabor convolutional networks. *IEEE Trans. Image Processing*, 27(9):4357–4366, 2018.

[23] Matija Marolt. A mid-level representation for melody-based retrieval in audio collections. *IEEE Transactions on Multimedia*, 10(8):1617–1625, 2008.

[24] Tadashi Matsuo, Hiroya Fukuhara, and Nobutaka Shimada. Transform invariant auto-encoder. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, pages 2359–2364. IEEE, 2017.

[25] Roland Memisevic and Georgios Exarchakis. Learning invariant features by harnessing the aperture problem. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 100–108. JMLR.org, 2013.

[26] Meinard Müller. *Fundamentals of Music Processing*. Springer Verlag, 2015.

[27] Oriol Nieto and Morwaread M Farbood. Identifying polyphonic patterns from audio recordings using music segmentation techniques. In *Proc. of the 15th International Society for Music Information Retrieval Conference*, pages 411–416, 2014.

[28] Bruno A Olshausen, Charles Cadieu, Jack Culpepper, and David K Warland. Bilinear models of natural images. In *Electronic Imaging 2007*, pages 649206–649206. International Society for Optics and Photonics, 2007.

[29] Wanli Ouyang and Xiaogang Wang. Joint deep learning for pedestrian detection. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 2056–2063. IEEE Computer Society, 2013.

[30] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik M. Schmidt, Andreas F. Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, pages 637–644, 2018.

[31] Stan Salvador and Philip Chan. FastDTW: Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[32] Laurent Sifre and Stéphane Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 1233–1240. IEEE Computer Society, 2013.

[33] Thomas C Walters, David A Ross, and Richard F Lyon. The intervalgram: an audio feature for large-scale melody recognition. In *Proc. of the 9th International Symposium on Computer Music Modeling and Retrieval (CMMR)*. Citeseer, 2012.

[34] Cheng-i Wang, Jennifer Hsu, and Shlomo Dubnov. Music pattern discovery with variable markov oracle: A unified approach to symbolic and audio representations. In Meinard Müller and Frans Wiering, editors, *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, pages 176–182, 2015.

[35] Gerhard Widmer. Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artificial Intelligence*, 146(2):129–148, 2003.

[36] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 7168–7177. IEEE Computer Society, 2017.

[37] Zhuoyao Zhong, Lianwen Jin, and Zecheng Xie. High performance offline handwritten chinese character recognition using googlenet and directional feature maps. In *13th International Conference on Document Analysis and Recognition, ICDAR 2015, Nancy, France, August 23-26, 2015*, pages 846–850. IEEE Computer Society, 2015.

[38] Yanzhao Zhou, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Oriented response networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 4961–4970. IEEE Computer Society, 2017.

FOLDED CQT RCNN FOR REAL-TIME RECOGNITION OF INSTRUMENT PLAYING TECHNIQUES

Jean-François Ducher
CICM – MUSIDANSE –
Université Paris 8/
IRCAM (UMR9912 STMS)
ducher@ircam.fr

Philippe Esling
IRCAM (UMR9912 STMS)
esling@ircam.fr

ABSTRACT

In the past years, deep learning has produced state-of-the-art performance in timbre and instrument classification. However, only a few models currently deal with the recognition of advanced *Instrument Playing Techniques* (IPT). None of them have a real-time approach of this problem. Furthermore, most studies rely on a single sound bank for training and testing. Their methodology provides no assurance as to the generalization of their results to other sounds. In this article, we extend state-of-the-art convolutional neural networks to the classification of IPTs. We build the first IPT corpus from independent sound banks, annotate it with the JAMS standard and make it freely available. Our models yield consistently high accuracies on a homogeneous subset of this corpus. However, only a proper taxonomy of IPTs and specifically defined input transforms offer proper resilience when addressing the "*minus-1db*" methodology, which assesses the ability of the models to generalize. In particular, we introduce a novel *Folded Constant Q-Transform* adjusted to the requirements of IPT classification. Finally we discuss the use of our classifier in real-time.

1. INTRODUCTION

Throughout modern history, western composers have diversified and refined *Instrument Playing Techniques* (IPTs) in order to foster innovation in the timbre space [14]. In folklore and oral traditions, IPTs sometimes stand out as a distinctive feature of the musical style [16, 22]. Therefore, their identification could contribute to the more general task of style recognition in the process of browsing in music databases. Moreover, interactive computer music systems (for instance in the field of improvisation or score-following) could hugely benefit from the development of real-time IPT classifiers [24].

In the last two decades, the MIR community has produced a lot of research in the field of timbre recognition, but there has been little effort in IPT classification, often considered as its *last frontier* [17].

One major cause of this gap in research is the lack of IPT sound banks. Lostanlen [17] has recently addressed the question of IPT recognition but limited his experiment to samples from isolated notes in a unique sound bank.

Here, our aim is to build a real-time classifier of IPT from solo recordings. Our system should be reactive to possibly rapid changes in the technique. Therefore, the preprocessing of the audio has to maintain temporal coherence and induce as little latency as possible. For instance, any segmentation of the audio (such as proposed by [21]) in order to subsume our task into a problem of classification of isolated notes would be irrelevant. Our study focuses on the cello but the methodological issues we raise are similar for other instruments and the process we use to build and train the classifier could be generalized as long as the IPTs of these instruments are included in a sufficient number of sound banks.

We show that trying to categorize cello IPTs in a unidimensional way produces weak results. The classifier performs well on homogeneous sets of data but generalizes poorly. Therefore we introduce a taxonomy of the playing techniques of the cello along 4 axes (named *exciter/vibrator*, *left-hand*, *waveform*, and *interaction position*). We aim to build a single network which classifies audio sequences in a multi-task [4] manner according to these axes. Then, we implement a rule-based system on top of this network, in order to simplify the model and yield a classification along the 18 main IPT categories.

In order to train our classifier, we produce a large corpus of labeled synthetic data with 5 IPT sound banks and their proprietary samplers. This corpus is annotated using the JAMS standard [11]. We make it available to the MIR community.

We adapt state-of-the-art models successfully used for instrument classification [11,16] to the multi-task and low latency requirements of IPT recognition. Front-end classifiers along the 4 IPT dimensions are built as fully-connected (FC) or recurrent layers on top of deep convolutional neural networks (CNNs). All tested system configurations achieve high accuracies on homogeneous subsets of our annotated corpus. This alone provides no indication of their ability to generalize to other databases or actual solo recordings. Therefore, we adapt the *minus-1db* methodology presented by Livshin [15] to the needs of our system. When subject to this methodology, we

show that RNN front-ends generalize better than FC. Our adapted *Folded Constant-Q Transform (FCQT)* also yields more stable performance than Log-Mel-Spectrograms. Finally, we assess the reactivity of our models for each of the 4 IPT dimensions.

2. RELATED WORK

2.1 Deep Learning and MIR

Following their success in the field of computer vision [10], deep learning techniques have been quickly adopted by the MIR community. Instead of using sets of hand-designed audio descriptors [1,7,15], these techniques rely on basic representations of the audio signal and let the algorithm learn suitable features for a given task. Convolutional (CNN), recurrent (RNN) neural networks and their combinations have been among the most popular architectures used in MIR. Convolutional layers seek local correlations within their input by training sets of convolutional kernels. CNNs are built by stacking such layers with *pooling* layers[1] at increasingly bigger scales. Therefore, they can detect large and complex patterns while being computationally efficient. RNNs have been developed in order to forecast or classify temporal sequences. As their hidden units have connections from one time step to the next, they can carry information through various temporal states. Long Short-Term Memory (LSTM) units, where gates enable to control this flow of information, have been proficiently used in MIR [6].

2.2 Instrumental Timbre Classification

Early research in instrument classification relied on samples of isolated notes played with ordinary techniques. In most studies, experiments were performed with a single sound bank. This practice overlooked variability related to the instrument model, player or recording environment. A detailed review of generalization issues by Livshin [15] shows that the performance of classifiers trained and tested with a single sound bank gives no hint on their accuracy when confronted to new sounds. He suggests to use several independent sound banks, pick one for testing while training the classifier on the rest joined together. Then, the experiment has to be repeated with all the possible test banks. This methodology named *minus-1db* provides more reliable indications on the ability of the classifier to generalize. In the case of instrument classification from solo recordings, it translates into a *leave-1-CD-out* policy.

To our knowledge, very few studies follow the methodological principles of Livshin, and, as such, can be regarded as being state-of-the-art in this matter.

Patil and al. [21] proposed a classifier built upon a support vector machine applied to spectro-temporal receptive fields. Trained on isolated notes of the RWC database to classify 6 instruments, this model reached 98.7% accuracy. Its resilience was assessed on a proprietary database of soli, which were first segmented using a harmonicity-based method. With the *leave-1-CD-out* methodology, accuracy still reached 88.1%.

Lostanlen and Cella [18] used two separate solo instrument databases to train and test a deep CNN to classify 8 instruments. Their network relied on the CQT of the audio signal. Through proper optimization of their convolution strategy, their system reaches average accuracies of 74%, against 61.4% for a decision tree forest applied to a large set of audio descriptors.

Regarding predominant instrument classification in polyphonic textures, Han and al. [11] achieved state-of-the-art F1-scores with a deep CNN applied to log-mel-spectrograms. This study was performed with two independent subsets of the IRMAS database for training and testing.

2.3 IPT Classification

IPT classification studies have been carried out on the clarinet [19], the snare drum [27], and the electric guitar [5]. The first two studies pose methodological issues since they perform training and evaluation with a single database. Chen and al. [5] focus on the detection in electric guitar solos of five techniques which all have an impact on the melodic contour. This feature is key to the design of their classifier. Therefore, their research can hardly be generalized to other IPTs.

Lostanlen and al. [17] tackle the issue of IPT recognition in a transversal manner. They work with samples from isolated notes belonging to 143 IPTs from 16 instruments. Their query-by-example system relies on a variant of the *k-nearest neighbors* algorithm where the metric used is subject to a training process. Applied to Mel-Frequency Cepstral Coefficients enriched by second-order scattering coefficients, it reaches *rank-5* accuracy of 61%. Yet, again, they train and test their system on a single sound bank.

3. IPT TAXONOMY

As we will show in Section 5., trying to classify IPTs without taking into account their multi-dimensional structure results in a poor ability to generalize. This motivates our newly introduced taxonomy.

3.1 Theoretical Background

A proper IPT taxonomy requires identifying what exciter, vibrator and resonator are selected (Schaeffer [26]), and what modification and excitation movements are undertaken (Cadoz [3]). Taking the example of the cello and following Feron [8] :

- among the possible exciters are the bow hair, bow wood, as well as various parts of the hand (finger, nail, knuckle). The natural vibrators are the four strings, but the body can be involved[2];
- modification movements are mainly the ornaments and other IPTs realized by the left hand (e.g. *vibrato*, *glissandi*, *harmonics*);

1 Pooling reduces the size of the output of the convolution (*called a feature map*) by downsampling ; generally, the maximum value of a local neighborhood is taken (*max-pooling*)

2 The *resonator* is assumed to remain unchanged (body).

- excitation movements should be characterized by their position (e.g. *sul tasto* or *ponticello),* length (e.g. *staccato*), eventual periodicity (e.g. *jettato, tremolo*), and the amount of speed and pressure involved (e.g. *flautando* vs. *pressured*).

3.2 Availability of Data

Proper definitions of all these IPTs should then be provided in order to annotate a recorded corpus of audio in a consistent manner.

However, the cost of such a study would be prohibitive. Therefore, we decided to rely upon available sound banks and their IPT definitions.

We identified 5 IPT sound banks which had different players and recording setups: EastWest Quantum Leap (EWQL), Vienna String Library (VSL), IRCAM Solo Instruments (ISI), Virtual Orchestra (VO) and ConTimbre (CONT). These banks suffer from two drawbacks. First, as expected, the absence of standardized definitions causes gaps in the realization of given IPTs between them, even for such a basic feature as the vibrato of an *ordinario* class. Second, each of the sound banks includes only a fraction of all the technical possibilities mentioned above. Several IPT combinations, albeit perfectly playable, are not available (e.g. *harmonic trills*).

3.3 Proposed Taxonomy

We match the list of IPTs in our sound banks with the theoretical approach in section 3.1. Bearing in mind our real-time constraint, we want to prevent an inflation of the number of classifiers and parameters in our model.

Therefore, we retain only 4 dimensions in our taxonomy (see Table 1). The first axis refers to the *exciter/vibrator* couple, which has a strong impact on the harmonicity and noisiness of the resulting sound. The second axis refers to how the *left hand* shapes the pitch contour. The third axis, called *waveform*, classifies IPTs depending on the nature and length of the bow/string interaction. The last axis refers to the *position* of the *interaction* with the string, which induces different spectral envelopes in the sound.

Axis 1 *Exciter/Vibrator*	Axis 2 *Left-hand*	Axis 3 *Waveform*	Axis 4 *Int. Position*
bow hair/string (*ordinario*)	NONE	NONE	NONE
bow wood/string (*con legno*)	*vibrato*	sustained	ordinario
finger/string (*pizzicato*)	non *vibrato*	*staccato*	*sul tasto*
finger/string+body (*pizz. Bartok*)	*glissando*	*spiccato, battuto*	*sul ponticello*
pressured bow/string	trill	*marcato, sfz*	harmonics
hand or knuckle hit on body		*tremolo*	

Table 1. IPT taxonomy (axes, classes) proposed in this study. This taxonomy is partly hierarchical in the sense that classification along Axes 2-4 is optional and dependent upon classification on Axis 1. When no classification is desirable, the *NONE* class is used in the training process[3].

3 For instance, a pizzicato will never be classified along the 3rd (waveform) axis. But it still could be classified along the 2nd axis (e.g. glissando) or the 4th (e.g. harmonics).

Some IPTs which would require a separate axis (should we aim at an exhaustive taxonomy for musicological purposes) were forced onto an existing dimension. Pressured bowing was included in axis 1 as it results in strong inharmonicity and noisiness. Harmonics, both natural and artificial, were included in axis 4, as they are most commonly played *sul ponticello*, but imply specific spectral envelopes. Finally, string classification[4] was excluded from the taxonomy, as well as the use of mutes.

On each dimension, each class of the taxonomy had to be represented at least in two sound banks for the *minus 1-Db* methodology to be implemented.

4. EXPERIMENTS

4.1 Building the Databases

4.1.1 Sequence Generation Principles

Bearing in mind our goal to generalize to actual solo recordings, we have generated series of audio sequences which simulate such recordings. This simulation tool was developed as a series of Max/Msp patches[5] controlling the proprietary samplers[6] of our 5 sound banks. The generated sequences include randomly generated notes and chords of all available IPTs. The chords are bounded by the playability of the instrument (as presented in [28]) and the specific ranges of the sound banks.

In the remainder of this article, we will use the term *database X* to designate the sequences which were generated with this process applied to the *sound bank X.*

4.1.2 Data Augmentation

We augment the data to increase the robustness of the classifier to variations in the recording environment and tuning of the instrument. Therefore, we generate sequences with a randomly detuned reference A4 in a 20Hz range around 440Hz (through transposition of the original samples). We also use various levels and types of reverb in the samplers. Finally, we average the stereo channels provided by the sampler to get the final signal.

After augmentation, the sequences represent 13.5 hours of music and over 4 Gb of AIF files sampled at 44100Hz and encoded at 16 bits PCM.

4.1.3 JAMS Standard Annotation

Our simulation tool exports both the annotation files with the JAMS format [13] and the corresponding audio. IPT classification along the 4 axes is provided with the *tag_open* namespace. Onset times and note/chord pitches were added when available, using *onset* (resp. *pitch_contour*) namespaces. Both audio and annotation files are made available to the MIR community[7].

4 All notes above G2 can be played on several strings. String change is regarded as a component of intra-class variability.

5 Available upon request

6 e.g. UVI Workstation for ISI, Vienna Instruments for VSL.

7 https://drive.google.com/open?id=1HYqHxxd2ZDkU2TL_1EXa6WNv9lY37hU9

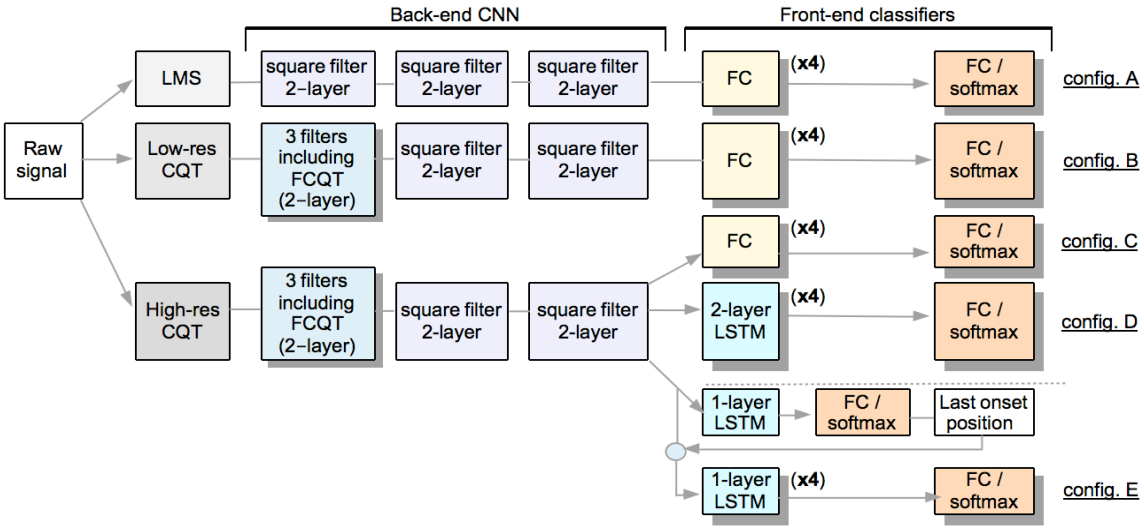


Figure 1. System architecture (incremental configurations A to E). A is directly inspired by [11]. B introduces CQT along with adapted filtering in the convolutional process. The resolution of CQT is increased in C. Recurrent layers process the output of the CNN in D. Finally, in configuration E, we experiment with the joint classification of onsets.

4.2 System Architecture and Configuration

4.2.1 Preprocessing

As shown in Figure 1, we have tested several possibilities[8] for the spectral transform.

Log-mel-spectrograms (LMS): following [11], we first downsample the signal to 22,050Hz, then compute LMS on 128 bins, with FFT windows of 2048 samples and hopsize of 512 samples (~23ms).

Adapted low-res CQT: following [18], we compute a 12 bin-per-octave CQT from C3(~130Hz)[9] to B10(~15.8kHz). Hop size is 1024 samples (~23ms). Only the logarithm of the amplitude of the CQT is retained. In order to preserve the temporal coherence of the preprocessed signal, we halve the Q-factor of the bins in the C3-B3 octave. In our adapted CQT, the size of the analysis windows are limited to 2850 samples (~64ms).

Adapted high-res CQT: to better account for IPTs such as *glissandi* and *vibrato*, we also experiment with a doubled bin-per-octave resolution above C5. The total number of bins of the CQT increases from 84 to 144. Since analysis windows are bounded, we cannot extend this resolution to the lower two octaves without deteriorating further the corresponding Q-factors.

In all configurations, we filter out low-energy frames (with average LMS or CQT < -79dB), normalize the data and cut it into fixed-length sequences of 60 frames (~1.4 s). Short sequences are less likely to include the attack of long notes, which is critical information for the network. A sequence length of 60 frames is an empirical compromise between this loss of information and the computing cost of longer sequences.

4.2.2 Folded Constant Q-Transform (F-CQT)

We introduce *F-CQT* as a generalization of the *pitch spiral* method [18] in order to capture efficiently the spectral envelope of a signal. It is obtained by first changing the pitch order of the CQT chromatic bins to match the cycle of fifths. The reshuffled CQT is then folded in 2 dimensions so as to put successive octaves on adjacent lines, in the same manner as the pitch spiral.

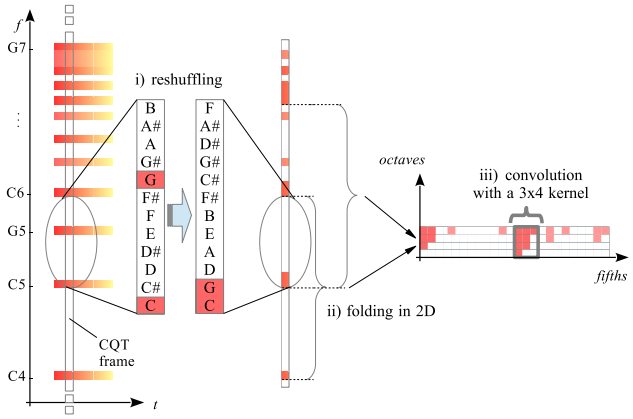


Figure 2. *F-CQT* example for a C4 note: (i) each CQT frame is reshuffled on an octave-per-octave basis, folded (ii) using 2-octave wide half-overlapping bands, in order to avoid side effects. A kernel (iii) of size 12 captures 8 out of the first 12 harmonics including the fundamental.

As shown in Figure 2, bins related to the harmonics $f_{2^i 3^j} = f_1 2^i 3^j$ of a given fundamental f_1 remain in its close neighborhood. Therefore, a small convolutional kernel of 3 *fifths* x 4 *octaves*[10] will capture 8 out of the first 12 harmonics. This is achieved without resorting to a computationally expensive harmonic-CQT [2]. Convolution with such a kernel can be seen as a 1D frequency-wise convolution of the usual CQT with a disjoint filter: the *F-CQT filter*. To capture the same

8 We use the Librosa [20] implementation for LMS and CQT.

9 It has been assumed that the lower octave of the cello would be analyzed with enough detail without the fundamental frequency being reported.

10 Tradeoff between the number of harmonics captured and the size of the kernel. A higher number of octaves results in blurring the picture with several harmonics on the same bin.

harmonics with a regular 1D-kernel would require a much bigger kernel size (43 parameters instead of 12).

4.2.3 CNN Back-end

We assume that the nature of our task is somewhat similar to the recognition of instruments in a challenging environment such as polyphonic textures. Therefore, we follow the main characteristics of the CNN architecture presented in [11], while adapting its capacity to our data.

The proposed CNN is made of 3 modules which operate at increasing scales. In each module we stack two identical convolutional layers, with batch normalization and *Rectified Linear Unit* (ReLU) activation. A max-pooling layer and dropout at 0.25 probability are implemented at the output of the module. Following the architecture design of [11], we use as baseline square (3x3) filters[11] for all layers. However it has been suggested in recent research [23] that domain-specific filtering could improve CNN performance, especially in the deeper layers. Therefore, we evaluate the use of three separate filters, namely our *F-CQT*, the baseline (3x3) and a (6x2) filter; the latter is designed to capture longer patterns such as *vibrato* or *trills*. The concatenation of 8 feature maps for each filter is used as input of the second module. Max-pooling layers are in charge of downsampling the features while the number of feature maps increases[12]. The output of the CNN is a 10 step long sequence of 64 maps with a single feature (one step equals six frames ~125ms).

4.2.4 IPT Classifiers (Front-end)

In configurations A to C, IPT classifiers are built with a fully-connected (FC) layer of 32 neurons with *ReLU* activation, followed by another FC layer with *softmax* activation. The latter comprises as many neurons as there are classes in the corresponding axis [10].

In configuration D (resp. E), we replace the first FC layer with a double (resp. single) layer of 32 unidirectional LSTM cells with an input of 64 features per temporal step.

In configuration E, following [12], an additional classifier with the same design is jointly trained to locate the attack of the last note of the sequence. Its eleven classes coincide with the 10 steps of the sequence plus one: this additional class is used to categorize sequences of long notes where the attack occurs prior to the beginning of the sequence. The prediction of this onset classifier is concatenated with the original features and used as input to the 4 IPT classifiers.

A rule-based system computes an 18-class linear classification from the network predictions along the 4 IPT axes. The same rules are applied to the ground truth. An 18-class accuracy is provided as an additional assessment of the performance of the system.

11 2D filters are noted: N *time frames* x M *frequency bins*.

12 Detailed architecture available here:https://drive.google.com/open?id=1GvS6VQ3iJP6e9MBajEPL0VOXva-iuUmS

4.3. Training Configuration

The system is trained by minimizing the sum of the cross-entropy loss function of the classifiers. Mini-batch gradient descent is performed with ADAM optimization [10] and exponential decay of the learning rate[13].

5. RESULTS

5.1 Direct classification (18 classes)

We first attempted to build directly a classifier among 18 cello IPTs, chosen simply because they were the most represented IPTs in our data. Our network architecture (called A_{18}) was similar to configuration A, but with a single 18-class classifier front-end. This effort resulted in excellent accuracies when the classifier was tested on homogeneous subsets of our corpus (see Table 2). However, these results collapsed when the *minus-1db* methodology was implemented. Not only average accuracies dropped below 50% but they were very irregular from one test base to the other. This indicated a poor ability to generalize.

Database excluded from training	18-class accuracy	
	Homogeneous corpus	Heterogeneous corpus *(minus-1db)*
CONT	92,9%	49,9%
EWQL	94,0%	30,3%
ISI	95,2%	51,1%
VSL	97,3%	44,0%
VO	93,4%	32,4%
Average A_{18}	94,6%	41,5%

Table 2. 18-class accuracy of configuration A_{18} with a single 18-class IPT classifier front-end. Results are given for 5 different training/testing subsets and averaged on 3 alternative learning schedules.

5.2 Introduction of our taxonomy

Trained and tested on 5 homogeneous subsets of our data, our models still yield in average 18-class accuracies over 90% in all the configurations (see Table 3). In the framework of the *minus-1db* methodology, they now exhibit much greater resilience. Their 18-class accuracies, averaged on all test bases, are above 50% in most configurations. These accuracies are also less sensitive to the choice of the test base, as shown by Figure 3 in the case of configuration D.

Configuration	Parameter count	18-class accuracy	
		Homogeneous corpus	Heterogeneous corpus *(minus-1db)*
A (LMS)	184K	93,5%	49,6%
B (Low-res CQT)	180K	90,1%	52,2%
C (High-res CQT)	181K	91,2%	54,3%
D (2 layers LSTM)	150K	91,3%	57,6%
E (Joint onset class.)	142K	91,7%	57,6%

Table 3. Parameter count and 18-class accuracy of configurations A-E, averaged on 5 different homogeneous or heterogeneous training/testing subsets and 3 alternative learning schedules.

On all axes but the *interaction position*, whatever the resolution, CQT with adapted filtering performs better than 128 bins log-mel-spectrograms (see Table 4). For

13 Detailed training parameters available at the same URL as 12.

instance, configuration C achieves better average and 18-class accuracy than A with roughly the same parameter count (Student t-test resp. p=0.046 and 0.004).

Accuracy / Configuration	Average on all axes	Exciter/ vibrator	Left-hand	Wave-form	Int. Position
A (LMS)	72,6%	84,2%	66,6%	77,1%	62,7%
B (Low-res CQT)	73,6%	85,6%	71,1%	78,1%	59,7%
C (High-res CQT)	74,7%	86,0%	72,6%	78,6%	61,7%
D (2 layers LSTM)	**76,0%**	**87,0%**	73,4%	**79,9%**	**63,5%**
E (Joint onset class.)	75,6%	86,7%	**73,8%**	79,5%	62,2%

Table 4. *Minus-1db* framework : per-axis and average accuracies in each configuration, for all 5 test databases and 3 alternative learning schedules.

We hypothesized that increasing the resolution of the CQT beyond the tempered scale would improve system accuracy for such IPTs as *vibrato* or *glissando*. Our experiment confirms that configuration C yields better average accuracies than configuration B on the *left-hand* axis (p=0.023). This results in better 18-class accuracy (p=0.045).

Configuration D with a 2-layer LSTM front-end achieves better average and 18-class accuracy than C with a fully-connected front-end (resp. p=0.039 and 0.02). Configuration E with joint onset classification but single-layer LSTMs also achieves better 18-class accuracy than C (p=0.016) with even lower parameter count. In both configurations E and D, all axes exhibit average improved performance compared to C but detailed results show discrepancies from one test base to the other.

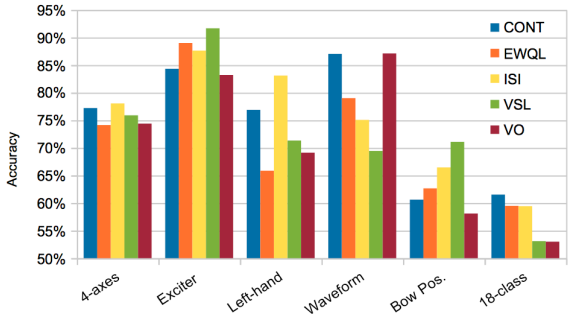


Figure 3. Accuracy per base and axis (configuration D).

In all tested configurations, the best accuracies are observed on the *exciter/vibrator* axis, while the worst-performing axis is *interaction position*. This statement is valid across most test databases, as seen in Figure 3 for configuration D. Bow position classification is likely to be a very difficult task even for a human expert. In the medium register, the choice of the string has a strong impact on the timbre, which makes the bow position harder to guess. In the higher register, *sul tasto* (with stronger emphasis or lower rank harmonics) and *ordinario* may be hard to distinguish. Finally, this is the axis where variability due to the instrument model is likely to be most perceivable.

5.3 Reactivity study

Our real-time classifier has to be as reactive as possible to sudden changes in the play of the cellist. To assess that reactivity, we select sequences in our test database where a change of IPT just occurred on the last note (or chord) of the sequence. We compute the average accuracy of the system as each time frame goes by. As Figure 4 exhibits, *exciter* is the axis where the classifier is most reactive, achieving >70% accuracy within 70ms of the attack. Unsurprisingly, it takes much longer for the network to categorize *left-hand* activity (e.g. *vibrato, trills)* or discriminate between *waveforms,* which often requires the note to be released (e.g. *staccato*). Finally, not only the *bow-position* axis yields poorer overall accuracy, but it is also the least reactive.

When the attack of the note gets out of the 60 frame-wide analysis window (see Figure 4, right side), the system has to categorize IPTs without information about the attack. However, its performance is not harmed as one could expect. Indeed, the actual length of the note provides information about the technique used. Longer notes tend, for instance, to be produced with the bow and to be vibrated or trilled.

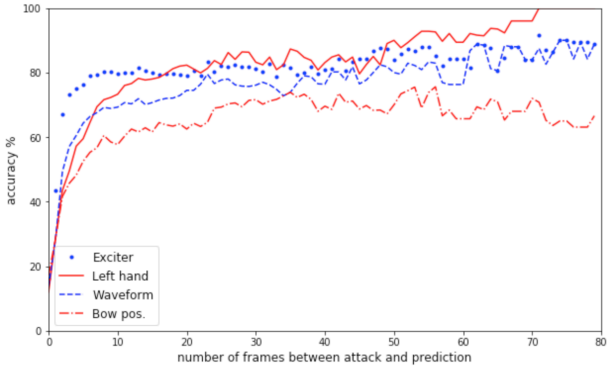


Figure 4. Average accuracy for sequences with an IPT change vs. time lapsed between change and prediction (10 frames~0,23s). Test base: ISI, configuration D.

6. CONCLUSIONS

In this article, we have extended state-of-the-art methods regarding instrument recognition to the real-time classification of IPTs from cello solo recordings. First experiments in the framework of the *minus-1db* methodology show a good resilience of models which are based on a meaningful taxonomy and process an adapted CQT through the proper combination of deep CNN and LSTM layers. Our methodology, from the realization of the databases to the architecture and training of the networks, can be extended with little effort to other string instruments. Other orchestral instruments first require an adaptation of the IPT taxonomy, which could be grounded on the same principles as ours [3,8,26].

To further assess the ability of our models to generalize, a database of contemporary cello solo recordings has been built, some of which will be annotated with the JAMS standard and used for testing. Finally, several unsupervised adaptation techniques, such as Maximum Classifier Discrepancy [25] or back-propagation through Gradient Reversal Layer [9], will also be tested in this environment.

7. REFERENCES

[1] D.G. Bhalke, C.B. Rama Rao, and D.S. Bormane: "Automatic Musical Instrument Classification using Fractional Fourier Transform based-MFCC Features and Counter Propagation Neural Network", *Journal Int. Inform. Syst.,* Vol. 46.3, pp. 425–446, 2016.

[2] R.M. Bittner, B. McFee, J. Salamon, P. Li, J.P. Bello: "Deep Salience Representations for f_0 Estimation in Polyphonic Music", *Proc. International Society for Music Information Retrieval (ISMIR)*, 2017.

[3] C. Cadoz: "Instrumental gesture and musical composition", *Proc. Int. Computer Music Conf.*, 1988.

[4] R. Caruana: "Multitask Learning". *Machine Learning*, Vol. 28, 1997.

[5] Y.-P. Chen, L. Su, and Y.-H. Yang: "Electric Guitar Playing Technique Detection in Real-World Recording Based on F0 Sequence Pattern Recognition", *Proc. ISMIR*, 2015.

[6] K. Choi, Fazekas G., K. Cho and M. Sandler: "A Tutorial on Deep Learning for Music Information Retrieval", arXiv:1709.04396v2, 2018.

[7] S. Essid: "Classification Automatique des Signaux Audio-fréquences: Reconnaissance des Instruments de Musique", PhD Thesis, Université Pierre et Marie Curie, Paris, 2005.

[8] F.-X. Féron, B. Carat: "Catégorisation des Actions Instrumentales dans Pression pour un(e) Violoncelliste de Lachenmann", *Manifeste 2014, Conférence "Composer (avec) le geste",* https://medias.ircam.fr/x23f46d, 2014.

[9] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand and V. Lempitsky: "Domain-Adversarial Training of Neural Networks", *Journal of Machine Learning Research*, Vol. 17, p. 1-35, 2016.

[10] I. Goodfellow, Y. Bengio, and A. Courville: *Deep Learning*, MIT Press, Cambridge MA, USA, 2017.

[11] Y. Han, J. Kim, and K. Lee: "Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music". *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 25, no. 1, pp. 208–221, 2017.

[12] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, D. Eck: "Onsets and Frames: Dual-Objective Piano Transcription", *Proc. ISMIR*, 2018.

[13] E.J. Humphrey, J. Salamon, O. Nieto, J. Forsyth, R.M. Bittner, J.P. Bello: "JAMS : a JSON Annotated Music Specification for Reproducible MIR Research", *Proc. ISMIR*, 2014.

[14] S. Kostka: *Materials and Techniques of Post Tonal Music,* Taylor&Francis, pp. 216-223, 2016.

[15] A. Livshin: "Automatic Musical Instrument Recognition and Related Topics. Acoustics", PhD Thesis, Université Pierre et Marie Curie, Paris VI, 2007.

[16] D.M. Lloyd: "A Classical Clarinetists Guide to Klezmer Music", PhD Thesis, Ohio State University, pp. 37-43, 2017.

[17] V. Lostanlen, J. Andén, and M. Lagrange: "Extended Playing Techniques: the Next Milestone in Musical Instrument Recognition". *5th International Workshop on Digital Libraries for Musicology*, Paris, France, 2018.

[18] V. Lostanlen, C.-E. Cella: "Deep Convolutional Networks on the Pitch Spiral for Music Instrument Recognition", *Proc. ISMIR*, 2016.

[19] M.A. Loureiro, H. Bastos de Paula and H.C. Yehia: "Timbre Classification Of A Single Musical Instrument". *Proc. ISMIR*, 2004.

[20] B. McFee, C. Raffel, D. Liang, D.P.W. Ellis, M. McVicar, E. Battenberg, and O. Nieto: "Librosa: Audio and Music Signal Analysis in Python", *Proc. of the 14th Python in Science Conf. (SCIPY)*, 2015.

[21] K. Patil, M. Elhilali, "Biomimetic Spectro-Temporal Features for Music Instrument Recognition in Isolated Notes and Solo phrases", *EURASIP J. Audio Speech Music Process*, 2015.

[22] C. Perret: "Une Rencontre entre Musique Savante et Jazz, Musique de Tradition Orale et les œuvres aux Accents Jazzistiques d'Érik Satie, Darius Milhaud, Igor Stravinsky et Maurice Ravel", *Volume !*, Vol. 2:1, pp. 43-67, 2003.

[23] J. Pons, O. Slizovskaia, R. Gong, E. Gómez and X. Serra, "Timbre Analysis of Music Audio Signals with Convolutional Neural Networks", *25th European Signal Processing Conference (EUSIPCO)*, 2017.

[24] R. Rowe: *Interactive Music Systems: Machine Listening and Composing*, Chapter 5, MIT Press, Cambridge MA, USA, 1993.

[25] K. Saito, K. Watanabe, Y. Ushiku, T. Harada: "Maximum Classifier Discrepancy for Unsupervised Domain Adaptation", arXiv:1712.02560, 2018.

[26] P. Schaeffer: *Traité des objets musicaux*, Editions du Seuil, pp. 52-53, 1966.

[27] A.R. Tindale, A. Kapur, G. Tzanetakis, and I. Fujinaga: "Retrieval of Percussion Gestures using Timbre Classification Techniques", *Proc. ISMIR*, 2004.

[28] J. Wiederker: *The contemporary Cello*, pp. 59-61, Ed. L'Oiseau d'or, Sainte-Geneviève-des-Bois, France.

HUMDRUMR: A NEW TAKE ON AN OLD APPROACH TO COMPUTATIONAL MUSICOLOGY

Nathaniel Condit-Schultz
Georgia Institute of Technology
natcs@gatech.edu

Claire Arthur
Georgia Institute of Technology
claire.arthur@gatech.edu

ABSTRACT

Musicology research is a fundamentally humanistic endeavor. However, despite the productive work of a small niche of humanities-trained computational musicologists, most cutting-edge digital music research is pursued by scholars whose primary training is scientific or computational, not humanistic. This unfortunate situation is prolonged, at least in part, by the daunting barrier that computer coding presents to humanities scholars with no technical training. In this paper, we present `humdrumR` ("hum-drummer"), a software package designed to afford computational musicology research for both advanced and novice computer coders. Humdrum is a powerful and influential existing computational musicology framework including the *humdrum syntax*—a flexible text data format with tens of thousands of extant scores available—and the Bash-based *humdrum toolkit*. `HumdrumR` is a modern replacement for the humdrum toolkit, based in the data-analysis/statistical programming language R. By combining the flexibility and transparency of the humdrum syntax with the powerful data analysis tools and concise syntax of R, `humdrumR` offers an appealing new approach to would-be computational musicologists. `HumdrumR` leverages R's powerful metaprogramming capabilities to create an extremely expressive and composable syntax, allowing novices to achieve usable analyses quickly while avoiding many coding concepts that are commonly challenging for beginners.

1. INTRODUCTION

Though digital musicology has been a productive area of research for several decades (e.g., [2, 8, 12, 16, 18–20, 22–24, 26, 28, 29]), it remains a niche field within the musical-side of academia. In fact, most cutting-edge, scientific music research has been pursued by researchers with primary training in computer science and psychology. Fortunately, recent years have seen a flourishing of "digital humanities" research in general, with increasing numbers of traditional humanities scholars adopting computational approaches. Humanist music scholars' deep, nuanced knowledge of musical culture, structure, and practice could be an invaluable asset to the computational/empirical music research community. Unfortunately, the training necessary for fruitful computational scholarship is absent from most music curricula, which cover neither the fundamental methodological principals of empirical research nor the necessary coding skills. The need to assist, and convince, music scholars to learn programming has been an area of active discussion for some time [7], in particular there is a need for software tools crafted to support their learning and research goals [32].

To make computational research truly appealing and accessible to traditional humanists and seasoned computational researchers alike, we must juggle three conflicting factors: flexibility, power, and usability. User-friendly interfaces like the Josquin Research Project's Analysis Tools [1], Theme Finder [2], or rapscience.net's Visualizer [3] can be utilized by scholars with no special training. However, such tools allow only variations of hard-coded analyses applied to limited, fixed databases. On the opposite extreme, any skilled programmer can code their own symbolic music data formats and analysis/parsing software "from scratch"—as many computational projects [4, 11] have done—, allowing for the unlimited power of the programming language of their choice, but requiring substantially more effort and experience. The most prominent modern computational musicology toolkit—`music21` [10]—, in our estimation, falls too close to the later extreme, proving quite daunting to novice coders.

This paper describes `humdrumR` ("hum-drummer"), a software toolkit for symbolic musicological data analysis intended to be appealing and accessible to traditional musicologists while also being useful to more experienced computational researchers. Learning lessons from the successes and failures of existing tools, `humdrumR` strikes a powerful new balance between flexibility, power, and usability:

1. Using the humdrum data syntax, `humdrumR` is extremely flexible and general in scope, allowing users to study any type of performance-art data that can be represented symbolically—musical scores, dance steps, trumpet fingerings, etc.

[1] http://josquin.stanford.edu/search/
[2] http://www.themefinder.org/
[3] http://rapscience.net/Analysis/gui.html

2. Based in the R programming language, `humdrumR` is extremely powerful, capable of complex data manipulation and interfacing with R's statistics, visualization, and machine learning libraries.

3. `humdrumR` is designed to present a relatively low barrier of entry for non-technical researchers, offering a concise, expressive syntax for applying music analyses while avoiding difficult coding paradigms.

In 2005, musicologist and psychologist Nicholas Cook observed that to truly engage in computational musicology "proper," scholars must achieve "sufficient understanding of the symbolical processing and data representation on which it's based" [7]. We believe that the combination of humdrum and R represents an ideal avenue for computational novices to develop this "sufficient" understanding: enough to pursue quality computational research but without having to engage with general coding paradigms that are irrelevant to their research.

In this paper, we first review the relevant philosophical and technical features of humdrum and R, noting how `humdrumR` incorporates these features (sections 2 and 3). We next contrast humdrum(R) coding philosophy and style with that of `music21` (section 4). Finally, we describe the principle features of the `humdrumR` package and `humdrumR` syntax (section 5), including numerous code examples.

2. HUMDRUM

Humdrum [4] is a system for computational musicology research, created by David Huron [17] (circa 1995) and maintained by Craig Sapp at Stanford's Center for Computer Assisted Research in the Humanities. Though no one digital musicology framework has ever truly dominated the field, humdrum is certainly among the most widely used and influential systems, being cited as the direct inspiration for some of its most successful competitors—`music21` [10] and MusicXML [13]—and with tens of thousands of scores available in humdrum format. Humdrum actually has two distinct components: the humdrum syntax and the humdrum toolkit.

The *humdrum toolkit* is a collection of Unix command-line tools for parsing and analyzing humdrum data, largely written in Bash and AWK but with more recent "extra" commands written in `C++`. [5] The humdrum toolkit is fundamentally entwined with the Bash shell, in particular the `grep` and `sed` commands, which rely heavily on *regular expressions* to parse and filter tokens. The humdrum toolkit also includes a number of analysis tools, notably the sophisticated windowing and n-gram tool `context` and the pattern finder `patt`. Using the toolkit, basic parsing and analysis can be achieved quickly and easily in Bash command pipes, but true Bash scripting is required for even mildly complicated tasks. In practice, after initial parsing, humdrum users must transition into a higher-level programming environment (Python, R, MATLAB, Julia, etc.) to achieve more complex data manipulation, statistical testing, or data visualization. The burden of parsing and manipulating humdrum data in the higher-level language falls on the user, substantially increasing the need for coding skills and prolonging the workflow. To make matters worse, installation of the humdrum toolkit can be fairly complicated, especially for Windows users, who must install a Unix emulator to use the toolkit at all.

`HumdrumR` is a successor to the humdrum toolkit, replacing all the original toolkit's functionality while adding significantly more. However, `humdrumR` uses the original humdrum syntax specification, and is thus compatible with all existing humdrum data. In fact, both the technical design and methodological philosophy of the humdrum syntax are fundamental to `humdrumR`.

2.1 The Humdrum Syntax

The humdrum syntax is an extremely general scheme for representing musicological data in plain-text. The syntax is basically tabular (tab-delineated columns) but with a few additional complexities:

- Columns of data are interpreted as *spines*, which can dynamically start, end, split, or merge, creating *spine paths*. Spine paths allow the syntax to represent many complex cases from music notation: For example, if the upper staff of a piano score splits temporarily into two voices (one with beams up, the other with beams down), this can be encoded by splitting the spine representing that staff into two columns (with a `*^` token) before merging them again after the split passage (`*v` tokens).

- Humdrum records are tagged as *interpretation records* (representing local metadata) or *data records* (notes, chords, etc.). By simply interspersing interpretation and data records, metadata can be concisely associated with specific data points, passages, or sections. For example, key information can be quickly read, edited, or added to scores by simply inserting lines containing tokens like `*f#:` (f# minor). This approach contrasts with the more verbose, hierarchical attributes and tags used in XMLs.

- Finally, each "cell" (i.e., each line-column coordinate) in a humdrum file can contain multiple data tokens—a feature commonly used to represent chords but with myriad other potential use cases.

The humdrum syntax provides a data *structure* but says nothing about how information is actually encoded in data tokens. Rather, specific *interpretation* schemes must be defined. A few well-specified interpretations for music are widely used, most notably the `**kern` representation. However, users can freely define new interpretations, with the appropriate degree of rigour/precision/complexity for the task at hand. Thus, humdrum is not limited to traditional Western notation, but rather affords the study of non-Western, vernacular, or avant-garde musics, as any type of

[4] http://www.humdrum.org

[5] Sapp also maintains a `C++` development library for humdrum tools called humlib (https://humlib.humdrum.org/).

musical feature, data, or metadata to be easily encoded in the same place. This presents a stark contrast with nearly all other music encoding schemes, including *MuseData* [15], abc [30], TinyNotation [9, ch. 16], LilyPond [25], and MusicXML, all of which are limited to representing music as, or in terms of, Western *music notation*—for instance, representing diatonic note names. Though `**kern` represents music in similar terms, the humdrum syntax in general has no such limitation. [6] Though the actively developing MEI [14] encoding standard *can* be extended to incorporate arbitrary music information, the emphasis of the MEI community is nonetheless representing music notation(s) for purposes of library science and score preservation and dissemination, not analysis.

Data encoding schemes inevitably balance read/writeability with power and flexibility [9, ch. 16]). Notation systems such as abc notation and TinyNotation achieve nearly effortless human read/writeability, but with severely limited representational possibilities. [7] On the opposite extreme, MusicXML and MEI can encode extremely complex, sophisticated score information, but are difficult to read/write directly. Humdrum achieves a balance between these extremes: though not quite as extendable as MEI, the humdrum syntax nonetheless affords a huge variety of approaches to encoding data. On the other hand, though `humdrum`'s top-down, tab-delineated format certainly makes editing slightly more cumbersome than editing abc or TinyNotation, most humdrum interpretations are comparably readable.

Like other easy notation schemes (abc, tinyNotation, LilyPond), humdrum interpretations often embed multiple pieces of information in each token. For instance, the `**kern` token `(4.aL/` encodes information about slurs (`(`), rhythm (`4.`), pitch (`a`), beaming (`L`), and stem direction (`/`). This "dense" approach allows a large window of musical time (for instance, several measures of multipart music) can be seen on a single screen. This contrasts with, for instance, MEI which spreads specific pieces of information across nested tags, making it impossible to see more than a few notes in a single screen. Such complicated tokens can also be written and edited rapidly, once you are familiar with the encoding. However, `kern`'s "dense" approach is but one option given humdrum's flexible syntax: information can be spread across multiple spines/columns (like *MuseData*) in whatever manner is most appropriate for data analysis. For example, the MCFlow corpus of rap transcriptions [6] encodes eight pieces of information across eight separate spines.

Human read/writeability is not just important to the process of data creation and curation, but is in fact essential to humdrum's entire methodological philosophy: humdrum emphasizes epistemological transparency by forcing users to engage directly with their symbolic data representations, even as they are filtered and transformed. Indeed, in traditional humdrum work flows, we apply repeated transformation to humdrum data tokens while maintaining, and visualizing the simple, readable humdrum syntax with each transformation. This dramatically improves the process of debugging and makes the series of steps from input to output clearer for novice users. Humdrum, thus, truly supports Cook's [7] call for a direct understanding of symbolic representations and processes. Consistent with this philosophy, `humdrumR` commands also reconstruct and display readable humdrum data even as the data is manipulated, maintaining a clarity and transparency which is easily lost when coding with complex data structures.

Though alternative data encodings have proven highly useful in contexts of research (*MuseData*, abc, MEI), file interchange (MusicXML), composition and engraving (TinyNotation, LilyPond, MEI), and performance (MIDI), the humdrum syntax offers an optimal combination of flexibility, read/writeability, and epistemological transparency, and is thus an ideal target for a new computational musicology toolkit. Fortunately, our focus on a single encoding scheme is not a limiting factor, as software to translate between `**kern` and most important representations—including MIDI, MusicXML, and MEI—is already available. In fact, fruitful cross-fertilization between musical data ecosystems is common: for instance, MEI's extraordinary Verovio score viewer has already been adapted as the Verovio Humdrum Viewer [8], making it easier than ever to visualize and edit humdrum data.

3. R

R [27] is a free, cross-platform, open-source "environment for statistical computing and graphics"—a domain specific programming language designed specifically for data analysis. R has a large ecosystem of data-analysis and statistics packages, most available through the Comprehensive R Archive Network (CRAN); As of now, the only programming environment with a comparable ecosystem for data analysis is Python. However, being less popular than Python for *general* programming, R's ecosystem is comparatively focused and uncluttered, with a higher quality floor. R's standard library ("base" R) itself contains analysis and visualization features which will satisfy the needs of many musicological research projects. Even when using external libraries, R users rarely encounter dependency issues—in fact, the process of installing R, `humdrumR`, and its dependencies is trivial on any operating system, even for beginner programmers, with no need for external package managers, virtual environments, "sandboxes," etc.

R excels at exploratory, ad hoc data analysis in short scripts or in the read-eval-print loop (REPL), making it easy to quickly manipulate, filter, and visualize data "on the fly." Indeed, a focus on simple scripting and "constrained" projects, without concern for more general software development issues, is core to R (and `humdrumR`) philosophy, making R an excellent avenue for learning programming purely for data analysis. This coding paradigm

[6] Conversely, humdrum/`**kern` is not as optimized for representing the details of music engraving as LilyPond, MEI, or MusicXML.

[7] However, `music21` includes an API for extending TinyNotation, useful to those with the requisite coding skills.

[8] http://verovio.humdrum.org/

is well suited to the constrained, stand-alone projects typical of theory-driven musicological research projects. The R ecosystem also includes a number of useful, free, software tools for enhancing the productivity, reproducibility, and presentation/sharing of research conducted in R. Notably, RStudio is a free, high quality integrated development environment for R. R is also one target of the Jupyter project [21], with RStudio too incorporating a suite of mark-up, notebook, and presentation tools. Finally, RStudio's `shiny` package [5] provides an easy means of creating interactive R data visualizations in the browser.

Though R is generally best suited to a combination of procedural and functional styles of programming, it nonetheless includes Object-Oriented Programming features suitable for defining simple classes. R's `S4` object system is oriented around multiple dispatch—generic functions can be defined which call specific methods based on the types of any or all of the function's arguments, not just their first argument. In languages featuring multiple dispatch (Julia, Common Lisp, Smalltalk, etc.), methods are not bound within classes. As a result, the object system operates in the background: novice users benefit from the features afforded by the object system—for instance, common functions like `summarize` can be applied to nearly any type of R object, getting useful results—without ever having to consciously engage with the system.

One of the core philosophies of R is "vectorization": treating data collections (especially vectors and arrays) as conceptually singular objects. `HumdrumR` leans into this philosophy, allowing users to think and operate on `humdrumR` data collections holistically. One concrete realization of this approach is that `humdrumR` users can completely avoid explicit iteration (i.e., loops): Iteration is abstracted by the `humdrumR` API, allowing users to decide solely *what* processes to apply to data tokens, not *how* to apply them to each token. `HumdrumR` defines a number of useful data classes, yet the R approach makes these classes an implementation detail that novice programmers need not understand.

3.1 Metaprogramming

The final key to the R ecosystem's concise data analysis syntax is its subtle use of metaprogramming [31, ch. 17–21]. In particular, numerous R packages use a shared domain specific language for specifying statistical models using R "formula" [31] objects.[9] Created using the `~` operator, R formulae capture ("quote") surrounding R expressions as well as their local namespace. R's metaprogramming features allow the programmatic manipulation of these formulae, for instance, using the `update` routine. Again, though metaprogramming is essential to R code, only advanced developers will ever need to explicitly engage with metaprogrammming concepts.

[9] For example, `Y ~ X*Z + (1|G)` describes a linear model predicting the variable `Y` using predictors `X` and `Z`, and the interaction between `X` and `Z`, with random-effect intercepts specified for each level of the grouping factor `G`.

4. music21

`Music21` is a Python library for symbolic music generation and analysis, with an extensive set of tools extending well beyond the capabilities of the humdrum toolkit, and which easily integrates with Python's extensive ecosystem (statistics and graphics libraries, etc.). Though `music21` and `humdrumR` overlap significantly in use case, `humdrumR` offers a fundamentally different coding philosophy and style.

Python syntax is famously simple to read/learn, yet the Pythonic coding style of `music21` nonetheless presents challenges to would-be computational musicologists. Working with `music21` requires one to engage directly with a hierarchy of complex classes (with *numerous* attributes and methods) and write many explicit control and looping structures, including (in typical analyses) multiple nested `for` loops. In contrast to humdrum, which relies on plain-text strings to encode information, `music21` parses musical scores into numerous complex data objects. Notably, `music21.Note` contains a rich set of attributes and methods for describing "notes." This complexity, though highly useful to experienced coders, is a barrier to entry for novices, for whom the practical reality of carrying out a computational musicology project is all but impossible. This is in part due to the style of `music21`'s User's guide, which necessarily gets bogged down explaining detailed functionality of how to represent, extract and manipulate low-level features (e.g., parts, notes, etc.) at the expense of explaining larger-scale processes like, for instance, how to search through a corpus of music to compare n-grams. Finally, `Music21`'s object hierarchy primarily represents musical score features—representations for arbitrary extra-musical data (e.g., dance steps, fingerings) or musical metadata (e.g., formal labels, manual annotations) is not supported.

`Music21`'s *Stream*-based object model is (purposely) extremely flexible [1]. As discussed above, the humdrum syntax includes some scope for flexible variation (using spine paths, for instance), yet `humdrumR`'s data backend (section 5.1) nonetheless always has the same structure; thus, one can always assume the same data structure and thus that certain commands/routines will always work. In contrast, `music21` users must always first determine the Stream-hierarchy of their data: are notes nested within measures, within parts, or within chords, etc? Similarly, `music21`'s highly sophisticated data classes (like `music21.Note`) are fairly inflexible, whereas humdrum's plain-text tokens are completely flexible. Again, this later approach is consistent with `humdrumR` philosophy, forcing users to think about and grapple with the transparent details of how their musical information is encoded and *not* about details of data structure.

As dynamically typed, interpreted languages, explicit loops in either R or Python are notoriously slow. However, whereas `music21` makes much explicit use of `for` loops, `humdrumR` enables users to exclusively use "vectorized" R and an optimized split-apply-combine backend, to achieve fast execution of most commands. As a re-

sult, humdrumR is generally much faster than music21.

5. HUMDRUMR

The humdrumR library defines a number of data object classes and a suite of functions for manipulating these classes. Most significant is the humdrumR class itself, which encapsulates a corpus of parsed humdrum files, and serves as the primary interface through which users interact with humdrum data. HumdrumR includes a complete humdrum syntax parser, which reads any valid humdrum data—including all valid spine paths—into a humdrumR data object. Invalid humdrum files are automatically flagged and skipped, with line-by-line syntax error reports generated on request. Consistent with the humdrumR philosophy, the syntax for reading files is concise and powerful: users simply specify one or more regular expressions matching files on their local disc. For instance, the command

```
readHumdrum("Bach/Chorales/chor.*krn")->chor
```

validates, reads, and parses all files matching the regular expression "chor.*krn" in the directory "Bach/Chorales" (370 files on our machine), wraps them in a humdrumR corpus object, and assigns this object to the variable chor. A set of summary functions are included to quickly describe the size, content, and structure of a loaded humdrumR data objects. In addition, an extensive suite of functions and classes for representing and manipulating pitch and rhythm data is also included—these tools reproduce much of the functionality of the original humdrum toolkit—as well as music21's core class hierarchy—, with some significant improvements and additions.

A great strength of the original humdrum toolkit is its use of the Bash | ("pipe") operator to concisely chain a series of operations—a syntax that is highly accessible to novice programmers. In recent years, the piping approach has become popular in R [3], especially magrittr's %>% pipe operator. HumdrumR too incorporates a pipe operator—%hum>%—which appears in all our subsequent code examples.

5.1 Data Model

R's primary native data structure is the tabular data.frame. HumdrumR utilizes a popular extension of the base R data.frame, the data.table [10]: an optimized data.frame which achieves database manipulation performance comparable to Python's Pandas module, including an extremely fast split-apply-combine routine. The HumdrumR class stores humdrum data in a list of data.tables, with each individual data token assigned to a single row. Data and metadata for each token are encoded in named columns of the data.table, called *fields*. The original string is encoded in the Token field, with global and local metadata associated with that token spread across other fields. Additional fields encode the "location" (which file, spine, path, record, etc.) of each token, encoding the structure of the original humdrum data so that it can be reconstructed for visual inspection after each modification. Users can freely create new fields; for instance, **kern tokens can be parsed into various pieces of information, each saved into a separate field: For example, the commands

```
chor %hum>% as.recip -> chor$Duration
chor %hum>% as.midi  -> chor$MIDI
```

create two new fields—Duration and MIDI—by applying the functions as.recip [11] and as.midi to the default Token field. These new fields can then be referenced like any other field in subsequent calls.

5.2 API

Much of R's expressive power arises from a subtle usage of metaprogramming to manipulate data.frames: Several base R functions—including subset, with, and within—allow users to input arbitrary R expressions which are then *evaluated within the data set* using the table's named fields as a local namespace. HumdrumR extends this paradigm to humdrum data, allowing users to apply arbitrary expressions to humdrum data stored in the underlying data.table back-end. Users capture expressions as R formulae and humdrumR API applies them to the data. These expressions can refer to any field in the data, including fields created by the user. The command

```
chor %hum>% ~table(MIDI[Duration == "4"])
```

(using the MIDI and Duration fields defined in the previous code block) extracts all MIDI values where the corresponding rhythm is a quarter note—using the standard R indexing ([]) operator—and then applies the base R table function, to tabulate these MIDI values. In a sense, this approach is an abstraction of function definition: One creates an expression—equivalent to the *body* of a function—but specifies no function arguments, as all data fields from the humdrum data are automatically passed into the expression if referenced.

The true power of the humdrumR arises through special *keyword* formulae which modify the API's behavior. Most notably, the by keyword can be used to split-apply-combine humdrum data. For instance, the command

```
chor %hum>% c(~table(MIDI[Duration == "4"]),
              by ~ File)
```

applies the exact same processing as the previous code block except the expression is applied separately to each file in the corpus, creating 370 separate tables. Since the humdrumR data fields include all data and metadata in the dataset, *any* field can be used to group data.

Other keyword formulae afford complex windowing, including n-grams, overlapping fixed-length windows, and various dynamic windowing possibilities. Finally, another set of keywords can be used to directly manipulate R's built-in visualization settings. Since formulae (including keyword formulae) are first-class objects in R, all of these expressions can be easily saved, composed, manipulated,

[10] https://cran.r-project.org/web/packages/data.table/index.html

[11] "Recip" is short for "reciprocal"—the humdrum term for standard Western duration categories (eighth-notes, sixteenth-notes, etc.).

and combined. For instance, we could save the tabling expression used above and use it in combination with various keyword formulae:

```
tabQuarters <- ~table(MIDI[Duration == "4"])
chor %hum>% c(tabQuarters, by ~ File)
chor %hum>% c(tabQuarters, by ~ Spine)
chor %hum>% c(tabQuarters, by ~ Clef)
```

The humdrumR API also includes routines for filtering and indexing a humdrum corpus. The standard R indexing operators ([] and [[]]) can be used to select pieces, records, or spines, either numerically or by matching regular expressions. More sophisticated filtering can be achieved through the use of humdrumR formulae: For instance, the command

```
filterHumdrum(chor,
              ~Token %~% '[EeAa]-',
              by ~ File)
```

selects all the files in the data set which contain the notes E♭ or A♭.

To bring it all together, a simple, yet complete humdrumR analysis script might look like this:

```
readHumdrum("Bach/Chorale/chor.*krn")->chor
chor %hum>% (~ as.midi(Token))
     %hum>% (~ Pipe - 60)
     %hum>% c(doplot ~hist(Pipe,
                           title = Clef,
                           xlim = c(30, 80)),
              by ~ Clef,
              mfcol ~ c(2,2))
```

These commands load the Bach chorale dataset, convert the original data tokens to MIDI numbers, subtract these numbers by 60 (to center them on middle C), then create a separate histogram for each clef in the data (in a 2x2 grid [12]). Note the use of the base R hist (histogram) function, including the use of the title and xlim (x-axis limits) arguments to give each plot a meaningful title and place all plots on the same scale.

5.3 Developer Tools

HumdrumR is designed to be highly extensible. Even novice users can quickly begin to create and save their own routines as R functions, or simpler yet, as combinations of humdrumR formulae. However, humdrumR also includes several features to support development by more sophisticated coders. All humdrumR data classes are intended to serve as extensible bases for further development—for instance, developers might choose to implement counterpoint analysis algorithms using humdrumR's basic tonalInterval class and its wealth of useful methods (transposition, inversion, etc.). However, the most significant tool for developers is humdrumR's Regular-Expression Method Dispatch System (REMDS). Interacting with humdrum data requires extensive string manipulation, typically using regular expressions, as one works to extract the information one is interested in from humdrum's "dense" character tokens. Using the REMDS, developers need only define normal R functions to manipulate the information they are concerned with and regular expressions to match that information. The REMDS can then be used to create generic functions which read an input string and dispatch the appropriate method based on matching regular expressions—what's more, these methods can (optionally) be applied "in place," only affecting the substring which matches the desired regular expression. For example, the humdrumR pitch module defines a number of functions which translate specific pitch encodings (note names, solfege, intervals, etc.) to/from humdrumR's common tonalInterval pitch representation. The REMDS is then used to generate generic pitch translation functions which call the desired method when a specific regular expression is matched. For instance, the function as.midi can be applied to strings containing a variety of pitch representations, even when embedded with other information (i.e., rhythm, beaming):

```
as.midi("4.cc#J") # "cc#" => **kern  =>  73
as.midi("4.C#5J") # "C#5" => **pitch =>  73
as.midi("4.-M9J") # "M9"  => **mint  => -14
as.midi("4.soJ")  # "so"  => **solfa =>  7
```

This approach allows users to use humdrumR functions without having to explicitly manipulate strings or use regular expressions, one of the major barriers to learning in the original humdrum toolkit. Many of humdrumR's own functions (like as.midi) are written using the REMDS, and developers can utilize it to significantly reduce coding effort when defining new functions.

6. FUTURE

The package source of HumdrumR 0.3.0 is currently available on github (natsguitar/humdrumR); when development solidifies, version 1.0.0 will be made available on CRAN under the terms of the GNU General Public License. However, releasing code is not enough to support humanist scholars interested in coding—it is imperative to provide high quality documentation and learning materials in a style that is digestible for users who may be new to computer programming. The most important contribution of the original humdrum project was neither the syntax nor the toolkit, but Huron's extensive user guide [17]. [13] The Humdrum User Guide offers a gentle introduction to empirical/computational research from a humanistic perspective, walking readers through the practical and philosophical details and challenges of digital humanities work and the conceptual transformations necessary to convert humanistic thought into concrete code, using examples from real musicology projects. HumdrumR too will ultimately be accompanied by a *humdrumR User Guide*, including interactive online content. Our goal is to not just teach the mechanics of operating software in a friendly, hands-on format, but also the conceptual framework needed to think about music as data, introducing key scientific principles/methods (data sampling, statistics, hypotheses, etc.) while maintaining a holistic, humanistic perspective.

[12] The keyword mfcol is a base R graphics parameter which controls the layout of plots in a grid.

[13] http://www.humdrum.org/guide/

7. REFERENCES

[1] Christopher Ariza and Michael Scott Cuthbert. The music21 Stream: A New Object Model for Representing, Filtering, and Transforming Symbolic Musical Structures. In *Proceedings of the International Computer Music Conference*. Citeseer, 2011.

[2] Claire Arthur. Taking harmony into account. *Music Perception: An Interdisciplinary Journal*, 34(4):405–423, 2017.

[3] Stefan Milton Bache and Hadley Wickham. *magrittr: A Forward-Pipe Operator for R*, 2014. R package version 1.5.

[4] John Ashley Burgoyne, John Wild, and Ichiro Fujinaga. An Expert Ground Truth Set for Audio Chord Recognition and Music Analysis. In A Klapuri and C. Leider, editors, *Proceedings of the 12th International Society for Music Information Retrieval (ISMIR) Conference*, pages 633–638, Miami, FL, 2011.

[5] Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie, and Jonathan McPherson. *shiny: Web Application Framework for R*, 2018. R package version 1.2.0.

[6] Nathaniel Condit-Schultz. The Musical Corpus of Flow: A digital corpus of rap transcriptions. *Empirical Musicology Review*, 11(2):124–147, 2016.

[7] Nicholas Cook. Towards the Complete Musicologist? Invited talk at ISMIR, 2005.

[8] Tim Crawford and David Lewis. Early modern cover song detection: finding concordances in mixed-notation corpora of early music. In *Music Encoding Conference, Tours France*, 2017.

[9] Michael Scott Cuthbert. music21: User's Guide, 2011.

[10] Michael Scott Cuthbert and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In *Proceedings of the International Society of Music Information Retrieval*. Proceedings of the International Society for Music Information Retrieval, 2010.

[11] Trevor de Clercq and David Temperley. A Corpus Analysis of Rock Harmony. *Popular Music*, 30(01):47–70, January 2011.

[12] Johanna Devaney, Claire Arthur, Nathaniel Condit-Schultz, and Kirsten Nisula. Theme and Variation Encodings with Roman Numerals: A New Data Set for Symbolic Music Analysis. In Meinard Müller and Frans Wiering, editors, *Proceedings of the 16th International Society for Music Information Retrieval (ISMIR) Conference*, pages 728–734, Malaga, Spain, 2015.

[13] Michael Good. MusicXML: An internet-friendly format for sheet music. In *XML Conference and Expo*, pages 03–04, 2001.

[14] Andrew Hankinson, Perry Roland, and Ichiro Fujinaga. The music encoding initiative as a document-encoding framework. In *Proceedings of the International Society for Music Information Retrieval*, pages 293–298, 2011.

[15] Walter B. Hewlett. *MuseData: Multipurpose Representation*, chapter 27, pages 404–407. MIT Press, 1997.

[16] David Huron. Note-Onset Asynchrony in J. S. Bach's Two-Part Inventions. *Music Perception*, 10(4):435–443, July 1993.

[17] David Huron. *Music Research Using Humdrum: A User's Guide*. Stanford, California: Center for Computer Assisted Research in the Humanities, 1999.

[18] David Huron. Tone and Voice: A Derivation of the Rules of Voice-Leading from Perceptual Principles. *Music Perception*, 19(1):1–64, September 2001.

[19] David Huron and Deborah A. Fantini. The Avoidance of Inner-Voice Entries: Perceptual Evidence and Musical Practice. *Music Perception*, 7(1):43–47, October 1989.

[20] Nori Jacoby, Naftali Tishby, and Dmitri Tymoczko. An Information Theoretic Approach to Chord Categorization and Functional Harmony. *Journal of New Music Research*, 44(3):219–244, 2015.

[21] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. Jupyter notebooks-a publishing format for reproducible computational workflows. In *ELPUB*, pages 87–90, 2016.

[22] David Lewis, Tim Crawford, and Daniel Müllensiefen. Instrumental idiom in the 16th century: Embellishment patterns in arrangements of vocal music. In *Proceedings of the International Society for Music Information Retrieval*, 2016.

[23] Kristen Masada and Razvan Bunescu. Chord Recognition in Symbolic Music Using Semi-Markov Conditional Random Fields. In *Proceedings of the 18th International Society for Music Information Retrieval Conference*, pages 272–278, 2017.

[24] Eric Nichols, Dan Morris, Sumit Basu, and Christopher Raphael. Relationships Between Lyrics and Melody in Popular Music. In *Proceedings of the 10th International Society for Music Information Retrieval (ISMIR) Conference*, pages 471–476. ISMIR, 2009.

[25] Han-Wen Nienhuys and Jan Nieuwenhuizen. LilyPond, a system for automated music engraving. In *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*, volume 1, pages 167–171, 2003.

[26] A. D. Patel and J. R. Daniele. Stress-Timed vs. Syllable-Timed Music? A Comment on Huron and Ollen (2003). *Music Perception: An Interdisciplinary Journal*, 21(2):273–276, 2003.

[27] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018.

[28] David Temperley and T. de Clercq. Statistical Analysis of Harmony and Melody in Rock Music. *Journal of New Music Research*, 42(3):187–204, 2013.

[29] Paul Von Hippel and David Huron. Why Do Skips Precede Reversals? The Effect of Tessitura on Melodic Structure. *Music Perception*, 18(1):59–85, October 2000.

[30] Chris Walshaw. *ABC2MTEX: An easy way of transcribing folk and traditional music, Version 1.0*. University of Greenwich, London, 1993.

[31] Hadley Wickham. *Advanced R*. Chapman and Hall, 2 edition, 2019.

[32] Frans Wiering and Emmanouil Benetos. Digital Musicology and MIR: Papers, Projects and Challenges. In *Proceedings of the International Society for Music Information Retrieval*, 2013.

TUNES TOGETHER: PERCEPTION AND EXPERIENCE OF COLLABORATIVE PLAYLISTS

So Yeon Park[1] Audrey Laplante[2] Jin Ha Lee[3] Blair Kaneshiro[1]
[1]Center for Computer Research in Music and Acoustics, Stanford University, USA
[2]École de bibliothéconomie et des sciences de l'information, Université de Montréal, Canada
[3]Information School, University of Washington, USA
{syjpark, blairbo}@stanford.edu

ABSTRACT

Music is well established as a means of social connection. In the age of streaming platforms, personalized playlists and recommendations are popular topics in music information retrieval. We bring the focus of music enjoyment back to social connection and examine how technologies can enhance interpersonal relationships, specifically through the context of the collaborative playlist (CP). We conducted an exploratory study of CP users and non-users ($N = 65$) and examined speculative and experienced purposes and outcomes of CPs, as well as general perspectives on music and social connectedness. We derived a CP Framework with three purposes—Practical, Cognitive, and Social—and two connotations—Utility and Orientation. Both users and non-users shared similar perspectives on music-related activities and CP user outcomes. Projected and actual CP purposes differed between groups, however, as did perception of music's role in connectedness in recent years. These results highlight the importance of music-based social interactions for both groups.

1. INTRODUCTION

Music has traditionally prompted social cohesion through mutually engaging properties such as cooperation and group empathy [26]. The importance of music's social implications is underscored by research on social interactions in online music sharing [8], in specific social contexts [2, 12, 13, 33], and in prototype designs [32, 34]. Social aspects are even highlighted in research with broader scopes. In exploring music information needs and behaviors through a large-scale user survey, researchers find "there is a strong social component to people's experience of and interaction with music" [39]. Such studies underpin the importance of technology's mediation in music. Yet, there is a relative dearth of research in current collaborative technologies for our most intimate and social experiences in music. Moreover, music's socially engaging traits are increasingly jeopardized by technologies that propagate individualized music consumption [17, 22].

Collaborative playlists (CPs), made possible through access-based consumption (i.e., streaming), are increasingly gaining traction [3]. For one, Spotify has allowed users to co-create and co-modify a playlist since 2008 [19]. Despite the importance of music's social qualities and increasing popularity of music co-consumption platforms, there is little research looking explicitly at today's phenomenon of collaborative playlisting. In contrast to related topics on recommendation systems and personal playlists, we lack an understanding of how CPs are used and enjoyed. Therefore, we can neither evaluate nor improve current systems in terms of meeting user needs and desires. Given these social benefits, and the fact that 86% of individuals in large music markets are purported to consume music via streaming platforms [24], characterizing the current state of how users feel about and interact with collaborative music platforms in an effort to bring "social" back into music is certainly a relevant topic in MIR research.

To address these needs, we explored the perception and experience of CP engagement by building upon prior work that identified behaviors and sentiments related to CPs [36]. We analyzed responses to selected questions from a larger survey to address the following research questions:

- RQ1: What are the distinct purposes and outcomes of CPs?
- RQ2: How do purposes and outcomes differ from those non-users predict CP usage would engender?
- RQ3: How do music perceptions, values, and habits differ between CP users and non-users?

Ultimately, an understanding of designing HCI through music co-consumption with better characterization of collaborative behaviors and needs from a user-centered perspective can help build HCI principles that can influence the landscape of human collaborations.

2. RELATED WORK

In the past, music listening was almost inevitably a social activity, through jukeboxes, radio, or the family record player in the living room. Only when music playback devices became more affordable and portable did music listening become an activity that could be enjoyed individually [17, 18, 22]. But these new practices did not displace

social practices around music. Research shows that music is still enjoyed socially (i.e., used as a social agent) to reinforce existing relationships and establish new ones [15,22]. Music preferences, especially during adolescence, play an important role in identity formation individually but also as a group of friends [18]; they convey information about a person's or group's values and beliefs. Music social practices are not limited to listening with others. They also include talking about music with friends and introducing them to new music [7]. When music collections were essentially or at least primarily physical, people shared music and prepared compilations with or for their friends [7], and mostly shopped for albums in music stores with others rather than alone [14]. Now that music has migrated online, have these social practices migrated too? Have music streaming services' collaborative and sharing features given rise to new social practices?

The advent of peer-to-peer (P2P) file sharing services (e.g., Napster, Gnutella) marked a turning point in music distribution and consumption. Researchers examined the social practices of users in these "online communities". Although these services offered ways for users to connect (e.g., chatrooms), few users "actively [sought] out chat or information sharing" [16]; interactions between users were "relatively infrequent"; and ties between them were mostly weak [37]. Brown and Sellen [7], who compared music sharing online and offline, concluded that when music was shared online, the social component of the activity was "stripped away". A study on iTunes sharing revealed that design decisions—such as partial user identification, and using a subnet ecosystem rather than P2P or in-person—impacted how social aspects of music sharing were supported [42].

More recently, music streaming services have enabled ubiquitous access-based consumption [3], and consequently have fueled research focused on selection, discovery, and listening through personal music collections [11, 20, 27, 39]. Streaming services have also provided many new affordances for supporting social music practices, one being the possibility of creating CPs. Even works on general music enjoyment and practices highlight "the growing need for tools to support collaborative music seeking, listening, and sharing" [29]. As such, works on creating social music technologies have been particularly numerous. Prototypes that aim to heighten the "extensive social functions" that music serves have been developed in the form of physical devices [32, 35] and digital platforms [5, 23, 25, 30, 31, 34]. Other works consider music recommendation based on group preferences [6, 9, 10] or integrate the collaborative functionality with other social components, such as interpersonal conversations [4], conflict management [41], and synchronous enjoyment [40]. These studies provide insights into the various aspects tackled or addressed in designing CP products. However, to the best of our knowledge, there have been no user studies on collaborative playlists or literature that considers long-term usage and outcomes relating to commercially available CP platforms. Therefore, we know very little about how CPs are used and perceived.

3. METHODS

3.1 Survey Design

Building upon past work [36], we designed a survey comparing CP users and non-users. We defined a CP as "a list of songs that multiple users have created using a digital platform"; CPs are distinguished from personal playlists in that they are also modified by other users. Our survey comprised open-ended and multiple-choice questions on perceived or experienced CP motivations, purposes, and outcomes; changes in behavior resulting from CP engagement; characteristics of users' favorite CPs; and impacts of CPs and music on social connectedness. We recruited participants through an introductory university music class, online music communities (e.g., Music group on Reddit), and social media (e.g., Twitter, Facebook). Anyone 18 years or older and fluent in English was eligible to participate. We provided no compensation for participating. Ethics approval was obtained from the Institutional Review Board of Stanford University.

3.2 Analyses

We focused on a subset of questions regarding CP purposes and outcomes, as well as music-related activities and music's role in social connectedness (questions are listed in Table 1). A three-step approach was used to analyze free-text responses (Q1). First, we decomposed each response into individual ideas and used affinity diagrams to group ideas in a data-driven fashion. We then categorized and labeled each of the groupings that emerged. Finally, the original responses were re-coded based on the identified groupings, and we computed counts and percentages of CP user and non-user responses that fell into each grouping. Quoted responses reference participants as "U" for users and "N" for non-users, followed by anonymized numbers.

Statistical analyses and data visualizations were conducted in R [38]. Differences between CP users' and non-users' word counts (Q1) and ordinal responses (Q2–Q4) were assessed using two-tailed Wilcoxon rank-sum tests. For Q2–Q4, each question comprised multiple responses (Table 1), so these p-values were corrected for multiple comparisons, on a per-question basis, using False Discovery Rate (FDR). We report significant ($p < 0.05$) and marginally significant ($0.05 \leq p < 0.10$) results, and FDR-corrected p-values.

4. RESULTS

We collected complete responses from 65 participants; 58% ($N = 38$) were CP users. Of users 42% were female ($N = 16$), and of non-users 41% were female. Participants ranged in age from 18–64 years (median 21 years); 89% ($N = 58$) resided in North America, with remaining participants from Europe ($N = 4$), Asia ($N = 2$), and South America ($N = 1$); and 86% ($N = 56$) were students. All CP users used Spotify to engage in CP activities, and Spotify was dominant among this group for other music consumption activities. Non-users were more varied in their choice of platforms (e.g., Apple Music, Pandora).

	Topic	Question	Response	Respondents
Q1	Purposes	What purpose(s) does/might a collaborative playlist serve for you?	Free-text	Users, interested non-users
Q2	Outcomes	Collaborative playlist(s) have/could... (10 statements, e.g., Diversify music library, Require less effort to enjoy music, Influence music taste positively).	Ordinal	Users, all non-users
Q3	Social connection through music	Please select the option that best represents your opinion on the following statements over the past 5 years: (4 statements, e.g., Personally, connecting with others through music has declined).	Ordinal	Users, all non-users
Q4	Importance of music activities with others	How important are these activities to your social relationships? (6 statements, e.g., Listening to recorded music with others, Sharing music with others).	Ordinal	Users, all non-users

Table 1. Survey topics, questions, response types, and respondents.

4.1 Purposes (Q1)

In answering RQ1 and RQ2, we unpacked free-text responses on CP purposes from all users as well as non-users who expressed interest in joining and/or initiating a CP (total $N = 55$).[1] Five main categories emerged from the affinity diagramming analysis (Figure 1): Three categories relating to purpose (Practical, Cognitive, and Social) and two relating to connotation (Utility and Orientation). Every response could be classified under at least one category, and many responses implicated multiple categories (i.e., category membership was not mutually exclusive). The subcategories emerged from our set of responses but are not exhaustive, and therefore may be expanded further.

Responses categorized as **Practical** implicated both the playlist object itself (the *artifact*) and the experience of playlist creation (the *process*). Users cited specific events (e.g., "party" (U21), "road trip" (U3)) or themes (e.g., "workout" (U16), "Christmas music" (U17)) as CP purposes; one non-user response ("serve as an outlet for entertainment" (N8)) suggested CP creation could itself be intrinsically enjoyable. **Cognitive** responses involved learning and discovery, both about music and about others, and thus centered around the user *receiving* information. Here, music discovery responses ranged from broad statements (e.g., "discover new music" (various)) to discovery specifically within or outside of established tastes (e.g., "get exposed to new music within my general musical interest" (N29) versus "finding interesting new music, particularly in genres I'm less familiar with" (U36)). Learning and discovery about others tended to focus on either listening habits or musical preferences of family and friends. **Social** responses reflected purposes directed *outward* from users. Responses around *sharing* ranged from very general (e.g., "share my music" (various)) to sharing with specific others (e.g., "sharing music with friends" (U1), "share music with a significant other" (U45)), and to sharing based on others' preferences (e.g., "allocate all of the songs that we think each other would like to listen to" (U24)). Other Social responses mentioned *bonding* (e.g., "I use it to bond with friends, especially friends who live far away" (U12), "connect to another person through song" (U40)).

In **Utility** responses, CPs are described as a means of reducing *effort* and increasing *efficiency* (e.g., "I don't have to do as much work to create a playlist" (N15), "creator wants help creating the playlist in a shorter deadline" (U19)). **Orientation** refers to delivery of benefits to the *self* or *others* (e.g., "allows me to receive music recommendations" (U31) serves the self, while "a place to allocate all of the songs that we think each other would like to listen to" (U24) connotes benefits to others).

Counts and percentages of responses from each participant group in categories are summarized in Table 2. While a single response could be classified under multiple purposes, the reported percentages still suggest insights into which categories were emphasized by users versus non-users. For example, more CP users tended to report Practical and Social purposes, while a greater percentage of non-users reported Cognitive purposes, as well as Utility and Orientation connotations.

While responses from CP users did not contain significantly more words than those of non-users ($p = 0.25$), they tended to be more diverse. One manifestation of this is in the variety of responses. For example, non-users' statements relating to social events all involved parties, while users also mentioned holidays, road trips, workouts, and dorm events. Users also highlighted ways in which CPs serve other eventual goals (i.e., an intermediary purpose), for example a CP created in preparation for a concert or

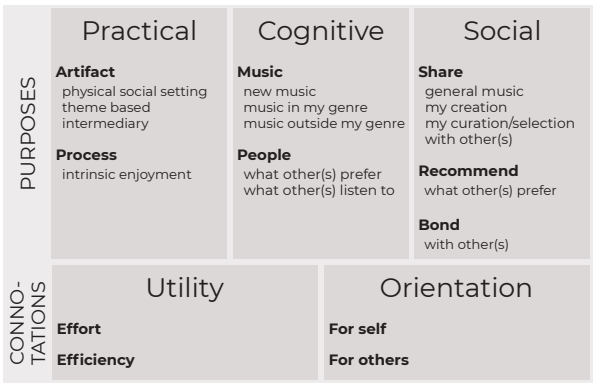


Figure 1. CP Framework: Purposes and Connotations.

[1] Non-user interest was assessed in a previous question. For Q1 we did not request responses from non-users ($N = 10$) who expressed no desire to participate in CPs for personal and/or logistical reasons.

	CP user	CP non-user	Total
Purposes			
Practical	**25 (66%)**	5 (29%)	30 (55%)
Cognitive	12 (32%)	**11 (65%)**	23 (42%)
Social	**25 (66%)**	6 (35%)	31 (56%)
Connotations			
Utility	6 (16%)	**4 (24%)**	10 (18%)
Orientation	23 (61%)	**14 (82%)**	37 (67%)

Table 2. Counts and percentages of user and non-user responses that reference CP purposes and connotations (Q1).

to centralize candidate repertoire for an a cappella group. CP users were also more descriptive and provided more specific use cases (e.g., "we have a playlist called 'Sharing Sundays' that we update weekly with a new song that we've been enjoying that week and then send the group a little message explaining why we chose that song" (U22)). Finally, the most nuanced responses spanning multiple purposes and connotations came from users:

- "A way to share songs and music tastes [Social]—both for the satisfaction of having others enjoy the same music I listen to [Orientation], and to find cool new music my friends share with me that I hadn't heard before [Cognitive]" (U7).
- "I use it to share music and to create larger playlists for team workouts or get-togethers with friends [Practical, Utility]. Oftentimes the playlist is for an upcoming event and whoever creates the playlist wants input from attending people in order to make the music maximally inclusive [Practical, Social]. Sometimes the playlist needs to be a few hours long and the creator wants help creating the playlist in a shorter deadline [Utility]" (U19).
- "Allows friends/ family to put songs on a playlist for everyone to enjoy and listen to [Practical, Orientation] and allows for others to share their music in a closed space with selective people [Social]. Also somewhat acts [as] a bonding experience and allows for people to bond over a mutual interest of music [Social]" (U30).

Furthermore, the frequency of updates could be inferred from the practical function the CP served: CPs for physical social settings and intermediary functions were usually created for one-off scenarios, whereas theme-based function connoted updates throughout the CP usage.

4.2 Outcomes (Q2)

Participant responses on actual (users) or speculative (non-users) CP outcomes were also considered for RQ1 and RQ2. Visualized results of medians and quartiles (Figure 2A) show that responses ranged from slight disagreement to slight agreement for decrease in time and effort to enjoy and manage music, whereas responses tended more toward agreement for "Diversify music library", "Increase ways of music discovery", and "Positively influence music taste". Median calculations across all participant responses show "Somewhat agree" as the dominant answer for most categories except decrease in time and effort to enjoy and manage music as well as "More open to new experiences", for which medians were "Neutral". As the plots suggest, responses between groups for "Decrease time and effort to manage music" (users $>$ non-users), "Increase ways of music discovery" (non-users $>$ users), and "Make listening to music more enjoyable" (users $>$ non-users) differed significantly when calculated independently, but were not significant after FDR correction.

4.3 Social Connection Through Music (Q3)

For RQ3, we found that users and non-users differed in their perception of music's social role (i.e., connecting with others), personally and in general (Figure 2B). Most participants disagreed that connections through music have declined, and agreed that music fosters connection. Users disagreed more strongly than non-users that personal connections through music have declined ($p = 0.03$). As marginally significant findings, users were more likely than non-users to report that music helps them to personally connect with others ($p = 0.05$); and non-users were more likely than users to report that in general, connecting with others through music has declined ($p = 0.08$).

4.4 Importance of Music Activities with Others (Q4)

Also related to addressing RQ3, participants rated the importance of musical activities to their social relationships (Figure 2C). With the exception of "Perform or create music", most responses fell between "Neither" (neutral) and "Very important". Inspection of the plot suggests distributional differences between users and non-users regarding discussing music, experiencing musical events, listening to recorded music, and sharing music with others. Quantitatively, differences between groups were marginally significant for "Listen to recorded music with others" ($p = 0.05$) and "Share music with others" ($p = 0.09$), with users reporting higher importance.

5. DISCUSSION

In this study, we analyzed survey responses from CP users and non-users. Analysis of free-text responses on speculative and actual CP purposes revealed three purposes (Practical, Cognitive, and Social) and two connotations (Utility and Orientation). Analyses of responses on CP usage outcomes, perspectives on social connection through music, and importance of music activities with others revealed similarities and differences between groups.

5.1 Similarities Between Users and Non-Users

Taking a holistic view across questions, we see similarities across participant groups. For example, both users and non-users were represented in each category of purposes and connotations identified from text responses to Q1 (Table 2); likewise, for Q2–Q4, overall patterns of responses were comparable across groups (Figure 2). These similarities could be attributed to non-users' awareness—of benefits and probable outcomes from CP usage—which

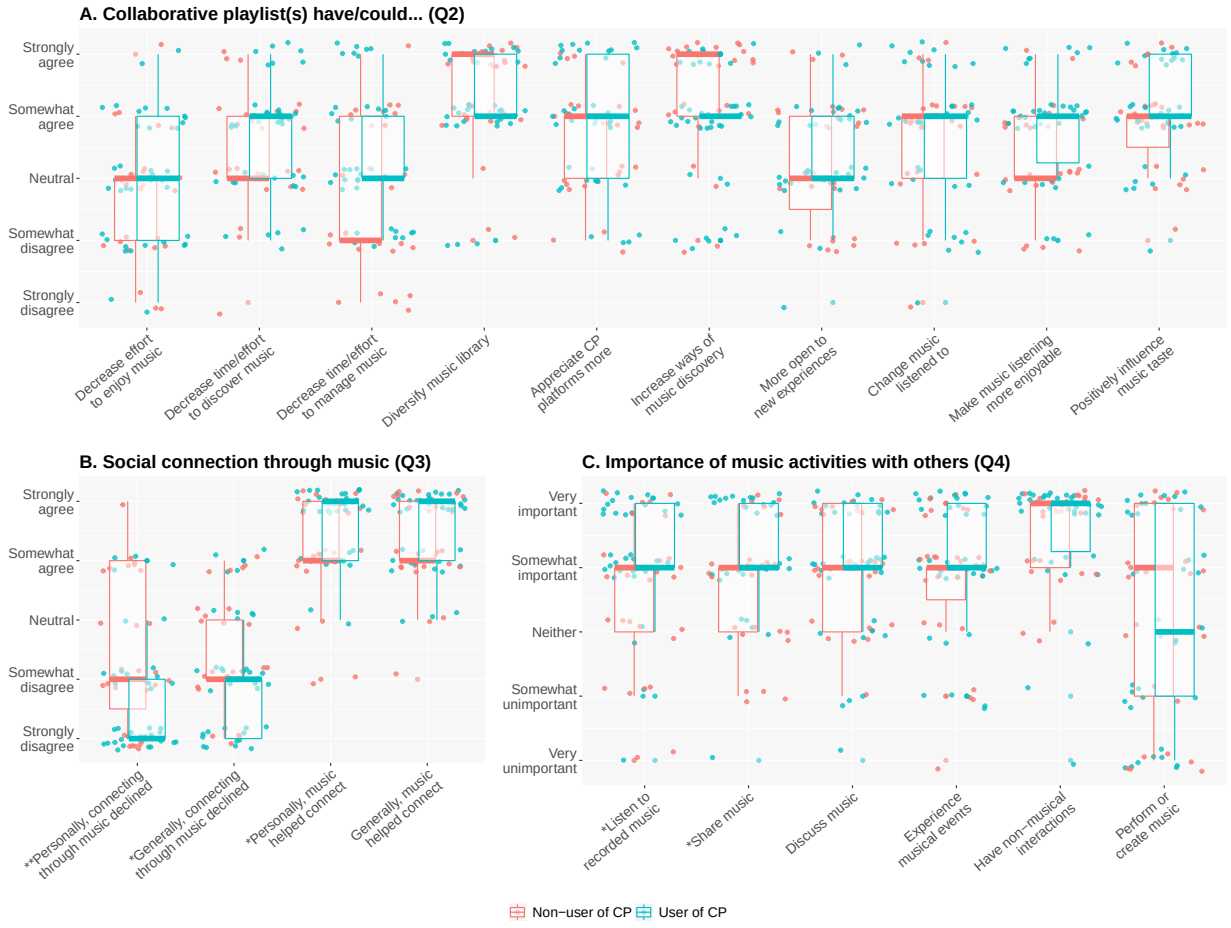


Figure 2. Boxplots of results for Q2–Q4: (A) Speculative and actual outcomes of CP usage (B) Social connection through music over the past 5 years (C) Importance of activities with others to social relationships (*$0.05 \leq p < 0.10$, **$p < 0.05$).

enabled them to be relatively accurate in their perceptions. While there were extreme responses outside the quartiles, most responses on music's social benefits were favorable. As observed in previous studies, we surmise that this was due in part to self-selection of survey participants [20] and recruitment through music-related channels [39].

5.2 Diverging Perspectives on Discovery

We also observed high-level differences between groups. Open-ended responses (Q1) show that a greater percentage of non-users' responses implicated Cognitive purposes (discovery, information seeking). Moreover, discovery was marginally more agreed-upon as a speculated purpose in Q2 by non-users than as an actual purpose by users. We propose several possible interpretations for these findings. One is that non-users might naturally translate discovery benefits offered in music personalization to the social playlist setting, whereas Practical and especially Social purposes are more specific to social music curation. Or, in actual CP usage, discovery is not as easy or actualized as speculated (e.g., if collaboration caters to shared tastes of a group [6, 9]), resulting in lower-than-expected outcomes for music discovery. Another possibility is that lower discovery outcomes for users stem from their CP purposes lying more in Practical and/or Social realms; hence users do not see a marked difference in their ways of discovery. Finally, users may perceive CP purposes or outcomes relative to one another; if Practical or Social purposes turn out to be most rewarding, they may serve as more dominant reasons for engaging in CPs and were thus reported more by users. Regardless of users' lower response to "Increase ways of music discovery", they still report that CPs have diversified and positively influenced their music taste. Therefore, while discovery may not be the prominent purpose for CPs, it is still an outcome.

5.3 Distinct Social Purposes

Compared to non-users, a higher percentage of CP users mentioned Social purposes (Q1). CP users also reported more personal connection through music (Q3) and higher quartiles for importance of music-related social activities (Q4). This might be due to reasons mentioned in §5.2, whereby non-users focus more on Cognitive purposes and users find Social outcomes of CPs more rewarding.

While we cannot attribute causality with certainty, we see that the Social purpose relates to valuing and experiencing connections with others, thereby distinguishing users from non-users. As anticipated, this is the biggest factor distinguishing CPs from personal playlists. This aspect has been reported in previous studies on social music curation, for example on bonding over shared music tastes [31, 34] or in creation and consumption of collabora-

tively curated playlists [8, 35]. Detailed comparisons with personal playlists are provided in §5.4.

Last but not least, the median "Somewhat agree" signal for "Appreciate CP platforms more" corroborates the statement made in prior work that "there is a strong social component to people's experience of and interaction with music, and music services that successfully incorporate such social features are well received" and lead to greater appreciation [29]. Perhaps CP platforms seeking to attract new users could highlight these attributes that were less reported by non-users.

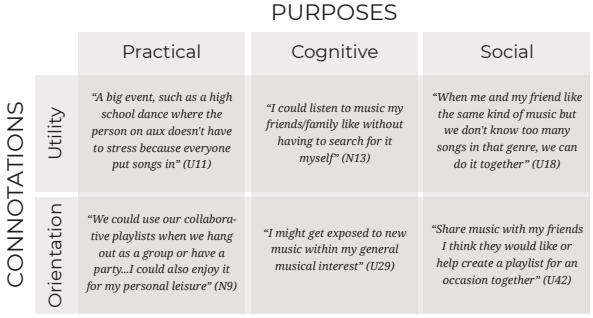


Figure 3. Collaborative Playlist Framework populated with example responses from participants.

5.4 CP Framework and Applications

Our purposes and connotations culminate in a Collaborative Playlist Framework (shown in Figure 3 with participant quotes). The framework co-articulates holistically one's reasons for engaging in CPs. The CP Framework can be applied to all responses in our study, and is also applicable to participant quotes from existing literature on social music. For example, "It would be very handy if you could ask the participants of your party to create the playlist to the party together" [31] expresses Practical purposes of CPs and Utility connotations of convenience.

We find the CP Framework to also be applicable to personal playlists, with Practical and Cognitive purposes and Orientation toward self being dominant. For example, a user from past literature described adding songs identified through Shazam to their Spotify playlist automatically [11], pointing to the discovery purpose. These songs then went into a "maybe playlist" from which the songs, depending on enjoyment, were moved into the "sometimes playlist", "officially [added] to my music collection", or deleted. These point to Practical purposes of personal playlists, also apparent in others' work [21].

We also observed similarities of Practical purposes between personal playlists and CPs. As discussed in §4 the similar contexts emerge from investigation on personal playlists, and in doing so reflect similar levels of detail as in the CP context [21]. Furthermore, we find that the update frequency implied from the playlist (artifact) type is consistent between personal and social playlists [1, 20, 21].

Just as one can have many reasons for engaging in an activity, CPs can be created with multiple purposes and connotations. These can evolve within and across the dimensions we have articulated, and can also facilitate one another. For example, suppose a group of friends decides to share the effort of creating a CP to play at a party [Utility, Practical]. The act of creating the CP provides an opportunity for social bonding [Social]. Collective consumption of the playlist at the party enables sharing and discovery [Social, Cognitive], ultimately bringing about further bonding [Social]. Facilitation can also occur within-category. While CPs may be created to fulfill a particular function for an event [Practical-artifact], the process itself of selecting music for a CP can be an intrinsically enjoyable activity [Practical-process], a phenomenon that has been reported in previous research [28, 34]. At times, sharing and suggesting music [Social-share/recommend] can in fact bring about the separate purpose of bonding [Social-bond]. We also already have hints of "with whom" participants (expect to) engage in CPs. Some create CPs with these personal connections in mind, while others might create CPs based on or in search of shared musical tastes [31]. Text responses indicate that such ensuing purposes, however, are not always sought by participants.

We acknowledge some shortcomings in the present work. A larger sample size could improve interpretability of results and also provide further insights into CP usage. In addition, by grouping responses based solely on CP usage, we may be overlooking valuable insights relating to other demographic factors; a larger sample size will help here as well. Finally, there are many other facets (e.g., ownership, group dynamics) through which this topic can be approached, and we are continuing our work through identification of CP usage patterns.

6. CONCLUSION AND FUTURE WORK

Our findings indicate that CPs have distinct purposes and outcomes (RQ1); non-users' speculation of CP usage outcomes do not differ greatly from actual user outcomes (RQ2); and differences in music perceptions, values, and habits exist between the groups (RQ3). These discoveries have direct implications for CPs, which are increasingly integrated into music consumption platforms. They are indicative of why users engage with CPs and what they gain from doing so. Consumption platforms may choose to heighten or make more conspicuous the features from which users derive the greatest benefits and be informed of other music-related activities to be integrated.

We continue to collect data and unpack the larger survey results to identify platform usage patterns of users and non-users. For CP users specifically, we analyze their survey responses and conduct semi-structured interviews to gain a nuanced understanding of their personal experiences and interactions with CPs. Our findings were predominantly based upon users in North America; we envision carrying out this study in other countries as well to examine cross-cultural perspectives. We will also be able to verify the validity of our CP Framework in other contexts and across time. Finally, informed by our findings, we aim to derive design implications for the kind of collaborative interfaces that users desire.

7. ACKNOWLEDGMENTS

We thank Jonathan Berger, Stanford's Music Engagement Research Initiative, and Stanford's Center for Design Research for helpful feedback and contributions.

8. REFERENCES

[1] C. Anbuhl. *Social and cultural practices around using the music streaming provider Spotify—A qualitative study exploring how German Millennials use Spotify*. PhD thesis, Malmö universitet/Kultur och samhälle, 2018.

[2] F. Axelsson and M. Östergren. SoundPryer: Joint music listening on the road. In *Adjunct Proceedings*, page 39, 2002.

[3] F. Bardhi and G. M. Eckhardt. Access-based consumption: The case of car sharing. *Journal of Consumer Research*, 39(4):881–898, 2012.

[4] J. S. Bauer, A. L. Jellenek, and J. A. Kientz. Reflektor: An exploration of collaborative music playlist creation for social context. In *Proceedings of the 2018 ACM Conference on Supporting Groupwork*, pages 27–38, 2018.

[5] S. Baumann, B. Jung, A. Bassoli, and M. Wisniowski. BluetunA: Let your neighbour know what music you like. In *CHI'07 Extended Abstracts on Human Factors in Computing Systems*, pages 1941–1946, 2007.

[6] F. Beierle, K. Grunert, S. Göndör, and A. Küpper. Privacy-aware social music playlist generation. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–7, 2016.

[7] B. Brown and A. Sellen. Sharing and listening to music. In *Consuming Music Together*, pages 37–56. Springer, 2006.

[8] B. Brown, A. J. Sellen, and E. Geelhoed. Music sharing as a computer supported collaborative application. In *Proceedings of the Seventh Conference on European Conference on Computer Supported Cooperative Work*, pages 179–198, 2001.

[9] D. L. Chao, J. Balthrop, and S. Forrest. Adaptive Radio: Achieving consensus using negative preferences. In *Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work*, pages 120–123, 2005.

[10] A. Crossen, J. Budzik, and K. J. Hammond. Flytrap: Intelligent group music recommendation. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, pages 184–185, 2002.

[11] S. J. Cunningham, D. Bainbridge, and A. Bainbridge. Exploring personal music collection behavior. In *International Conference on Asian Digital Libraries*, pages 295–306, 2017.

[12] S. J. Cunningham and D. M. Nichols. Exploring social music behaviour: An investigation of music selection at parties. In *ISMIR*, pages 747–752, 2009.

[13] S. J. Cunningham, D. M. Nichols, D. Bainbridge, and H. Ali. Social music in cars. In *ISMIR*, pages 457–462, 2014.

[14] S. J. Cunningham, N. Reeves, and M. Britland. An ethnographic study of music information seeking: Implications for the design of a music digital library. In *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 5–16, 2003.

[15] T. DeNora. *Music in everyday life*. Cambridge University Press, 2000.

[16] S. Ebare. Digital music and subculture: Sharing files, sharing styles. *First Monday*, 9(2), 2004.

[17] J. Frank. *Futurehit.DNA: How the digital revolution is changing Top 10 songs*. Futurehit, Inc., 2009.

[18] S. Frith. *Performing rites: On the value of popular music*. Harvard University Press, 1998.

[19] K. Gilmour. Collaborate on playlists with spotify's collaboration feature. `https://www.dummies.com/social-media/spotify/collaborate-on-playlists-with-spotifys-collaboration-feature/`, retrieved April 8, 2019.

[20] A. N. Hagen. The playlist experience: Personal playlists in music streaming services. *Popular Music and Society*, 38(5):625–645, 2015.

[21] A. N. Hagen and M. Lüders. Social streaming? Navigating music as personal and social. *Convergence*, 23(6):643–659, 2017.

[22] D. J. Hargreaves and A. C. North. The functions of music in everyday life: Redefining the social in music psychology. *Psychology of Music*, 27(1):71–83, 1999.

[23] J. Haupt. Last.fm: People-powered online radio. *Music Reference Services Quarterly*, 12(1-2):23–24, 2009.

[24] IFPI. Music consumer insight report. 2018. `https://www.ifpi.org/downloads/Music-Consumer-Insight-Report-2018.pdf`, retrieved April 8, 2019.

[25] D. S. Kirk, A. Durrant, G. Wood, T. W. Leong, and P. Wright. Understanding the sociality of experience in mobile music listening with Pocketsong. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*, pages 50–61, 2016.

[26] S. Koelsch. From social contact to social cohesion—the 7 Cs. *Music and Medicine*, 2013.

[27] A. E. Krause, A. C. North, and L. Y. Hewitt. Music-listening in everyday life: Devices and choice. *Psychology of Music*, 43(2):155–170, 2015.

[28] A. Laplante and J. S. Downie. The utilitarian and hedonic outcomes of music information-seeking in everyday life. *Library & Information Science Research*, 33(3):202–210, 2011.

[29] J. H. Lee, H. Cho, and Y. S. Kim. Users' music information needs and behaviors: Design implications for music information retrieval systems. *Journal of the Association for Information Science and Technology*, 67(6):1301–1330, 2016.

[30] A. Lehtiniemi and J. Ojala. Evaluating MoodPic—a concept for collaborative mood music playlist creation. In *2013 17th International Conference on Information Visualisation*, pages 86–95, 2013.

[31] A. Lehtiniemi, J. Ojala, and K. Väänänen. Socially augmented music discovery with collaborative playlists and mood pictures. *Interacting with Computers*, 29(3):416–437, 2017.

[32] E. Lenz, S. Diefenbach, M. Hassenzahl, and S. Lienhard. Mo. Shared music, shared moment. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, pages 736–741, 2012.

[33] T. W. Leong and P. C. Wright. Revisiting social practices surrounding music. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 951–960, 2013.

[34] K. T. Liu and R. A. Reimer. Social Playlist: Enabling touch points and enriching ongoing relationships through collaborative mobile music listening. In *Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 403–406, 2008.

[35] K. O'Hara, M. Lipson, M. Jansen, A. Unger, H. Jeffries, and P. Macer. Jukola: Democratic music choice in a public space. In *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, pages 145–154, 2004.

[36] S. Y. Park and B. Kaneshiro. An analysis of user behavior in co-curation of music through collaborative playlists. In *Extended Abstracts for the Late-Breaking Demo Session of ISMIR*, 2017.

[37] K. Poblocki. The Napster network community. *First Monday*, 6(11), 2001.

[38] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018.

[39] L. Spinelli, J. Lau, L. Pritchard, and J. H. Lee. Influences on the social practices surrounding commercial music services: A model for rich interactions. In *ISMIR*, 2018.

[40] M. Stewart, J. Tibau, D. Tatar, and S. Harrison. Co-designing for co-listening: Conceptualizing young people's social and music-listening practices. In *International Conference on Social Computing and Social Media*, pages 355–374, 2018.

[41] F. Vieira and N. Andrade. Evaluating conflict management mechanisms for online social jukeboxes. In *ISMIR*, pages 190–196, 2015.

[42] A. Voida, R. E. Grinter, N. Ducheneaut, W. K. Edwards, and M. W. Newman. Listening in: Practices surrounding iTunes music sharing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 191–200, 2005.

A HOLISTIC APPROACH TO POLYPHONIC MUSIC TRANSCRIPTION WITH NEURAL NETWORKS

Miguel A. Román
U.I. for Computing Research
University of Alicante, Spain
mroman@dlsi.ua.es

Antonio Pertusa
U.I. for Computing Research
University of Alicante, Spain
pertusa@dlsi.ua.es

Jorge Calvo-Zaragoza
U.I. for Computing Research
University of Alicante, Spain
jcalvo@dlsi.ua.es

ABSTRACT

We present a framework based on neural networks to extract music scores directly from polyphonic audio in an end-to-end fashion. Most previous Automatic Music Transcription (AMT) methods seek a piano-roll representation of the pitches, that can be further transformed into a score by incorporating tempo estimation, beat tracking, key estimation or rhythm quantization. Unlike these methods, our approach generates music notation directly from the input audio in a single stage. For this, we use a Convolutional Recurrent Neural Network (CRNN) with Connectionist Temporal Classification (CTC) loss function which does not require annotated alignments of audio frames with the score rhythmic information. We trained our model using as input Haydn, Mozart, and Beethoven string quartets and Bach chorales synthesized with different tempos and expressive performances. The output is a textual representation of four-voice music scores based on **kern format. Although the proposed approach is evaluated in a simplified scenario, results show that this model can learn to transcribe scores directly from audio signals, opening a promising avenue towards complete AMT.

1. INTRODUCTION

Automatic music transcription (AMT) aims to convert acoustic music signals into any sort of music notation. Most of the music we listen today is polyphonic, where simultaneous sound events produced by different audio sources (i.e., instruments) are combined in a single acoustic waveform. This aggregation process entails loss of information, making the transcription task very challenging even for trained musicians. Moreover, the different sound events are highly correlated in time and frequency due to the rhythmic and harmonic patterns usually found in music, which complicates sound separation even further as we cannot rely on the statistical independence of the source signals. Therefore, in order to produce a proper music score from an audio signal, multiple complex sub-tasks must be involved such as multi-pitch estimation, note onset/offset detection, source separation, as well as other musical context information retrieval tasks like metering and tonality estimation.

As pointed out by [2], there are many approaches to tackle AMT, yet most works focus on solving only one intermediate goal of the whole problem. Frame-level transcription, also known as multi-pitch estimation, aims to detect which fundamental frequencies are present at each time step of the input signal. Note-level transcription goes a step further by estimating the notes characterized by their pitch and clock-time duration (onset and offset times), producing a piano-roll representation of the music. Stream-level transcription extends the note-level approach by associating each note with its originating instrument based on its timbre. Lastly, the notation-level transcription is the final goal of AMT, aiming to produce a music score with enough information to interpret the original recording.

In this work, we denote the notation-level transcription as Audio-to-Score (A2S) task, where the audio signal is processed to be converted into a symbolic music score. Even with a perfect transcription, the output of any A2S system cannot faithfully represent the music that was originally played. It must be considered that musical audio signals are often expressive performances, rather than simple mechanical translations of notes read from a staff. A particular score can be performed by a musician in many different ways, and similarly there are several ways to represent the same musical excerpt with standard music notation (e.g., a dotted half note is "the same" as a half note tied to a quarter note). Music scores can only be seen as guides to aid musicians, highly correlating but never fully explaining musical experience. This makes A2S a rather ill-defined problem without unique solutions.

Despite the above, our work aims to demonstrate that the A2S task can be performed in a single step. To this end, we make use of a deep neural network that is trained in an end-to-end fashion to produce a sequence of musical symbols that describes a feasible polyphonic score out of the input audio. Our experiments are conducted using Haydn, Mozart, and Beethoven string quartets and Bach chorales synthesized with different tempos and expressive performances. [1] Although the analysis of the current performance requires a deeper reasoning regarding evaluation

[1] The source code and data are available at https://github.com/mangelroman/audio2score.

metrics, we provide some results that account for the good performance of the proposed model and allow us to be optimistic about this line of research.

1.1 Related Work

There are recent AMT approaches using deep neural networks for the multi-pitch detection task [8, 11, 12]. For this, Short-Term Fourier Transform (STFT), log-frequency STFT or Mel spectrograms are usually fed to Convolutional Neural Networks (CNN) to extract piano-roll representations as output. Other works focus on producing music scores from unquantized MIDI representation [4].

One of the few methods aiming to extract a complete score directly from audio is that of [14]. In this work, a multi-pitch detection method with note tracking is used to get a piano-roll representation that is further converted into a quantized MIDI file by using a rhythm quantization method [15]. Afterwards, a score typesetting software such as MuseScore can be used to get a MusicXML file from the MIDI output.

To the best of our knowledge, there are only two approaches that perform A2S in a single stage, directly converting the input audio into any music notation format. This has the advantage that a wrong detection in a given stage (such as the multi-pitch detection) is not propagated through the next processing stages, avoiding error cascading. The only works addressing notation-level AMT in an end-to-end manner are those of [3] and [17]. Both works follow a supervised learning approach with deep neural networks to solve the AMT task in one step. Although they bring promising results, the proposed models include several limitations (e.g. monophonic audio in [17] and fixed input length in [3]) that cannot be disregarded when addressing the notation-level AMT problem as a whole.

In [3], authors show how a Convolutional-Recurrent Neural Network architecture (CRNN) [19] can learn all the basic tasks involved in notation-level AMT, but it is only a very limited proof of concept that cannot address most of the possible scores. In the second of these works [17], the AMT problem was addressed as an Automatic Speech Recognition (ASR) problem. By using monophonic audio as input and a sequence of symbols (analogously to the written language characters) as output, several methods that were originally developed for ASR can be used. In particular, [17] adopted an architecture inspired in DeepSpeech2 [1], which learns to map audio frames to a sequence of characters without any alignment.

Training with unaligned data, i.e. without needing the input audio frames to be aligned with the music symbols, is a clear advantage as much more data can be gathered without going through the tedious task of manually annotating the location of the output symbols in their corresponding input audio frames. Nevertheless, monophonic audio transcription does not exhibit the essential challenge coming from simultaneous sound events. The present work goes one step beyond by showing that a similar formulation can also be reliable for polyphonic music.

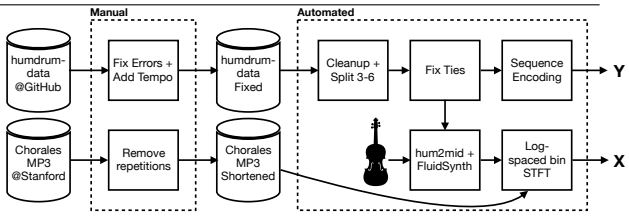


Figure 1: Data acquisition pipeline showing the manual and automated steps required to build the ground truth.

2. DATA

Our notation-level AMT approach, namely A2S, seeks to estimate which music score, modeled as a structure containing symbols from a fixed alphabet of music notation, would likely define that audio.

Let $\mathcal{X}$ be the domain of audio files and Σ the alphabet of music score symbols. The aim of our A2S is to compute a function that maps any audio file into a sequence of symbols, i.e., a function $f : \mathcal{X} \rightarrow \Sigma^*$.

2.1 Input Representation

The input representation of our model is the spectral information of the raw audio waveform over time, based on the STFT with log-spaced bins and log-scaled magnitude. In this type of spectrogram, frequency bins are aligned with equal-tempered music scales using 440Hz as the reference for A4 pitch. The sampling rate of the input audio files was 22,050Hz, and STFT was calculated with a Hamming window with size 92.88ms (2048 samples) and a hop of 23.22ms (512 samples). Only frequencies between pitches C2 and C7 were considered, extracting 48 bins per octave.

2.2 Output Representation

The output music notation of our model is a single sequence of symbols that can be used to render a multi-part western music score. These symbols represent both notes and rests with their corresponding duration, barlines, ties between notes, and fermatas. It is important to remark that in the context of the A2S task, notes are not the same as pitches. For example, pitch 349.23Hz can be represented as F4, E♯4 or G♭♭4 depending on the key signature.

We are not including clefs in our output representation, as they are only intended to aid in the score visualization and do not carry any musical information we can extract from the audio. For the sake of simplicity, we are also not including time signatures in the output sequences assuming they can be inferred from the predicted barlines for the type of scores in our training set. By the same rationale, key signatures are neither included assuming they can also be inferred from the predicted notes. Moreover, since most of our samples are made of fragments of much longer scores, they may not carry enough information to predict the correct key signature, therefore misleading the training process. Other music notation symbols such as slurs, grace notes, ornaments, and articulations marks are also left out of the scope of this work.

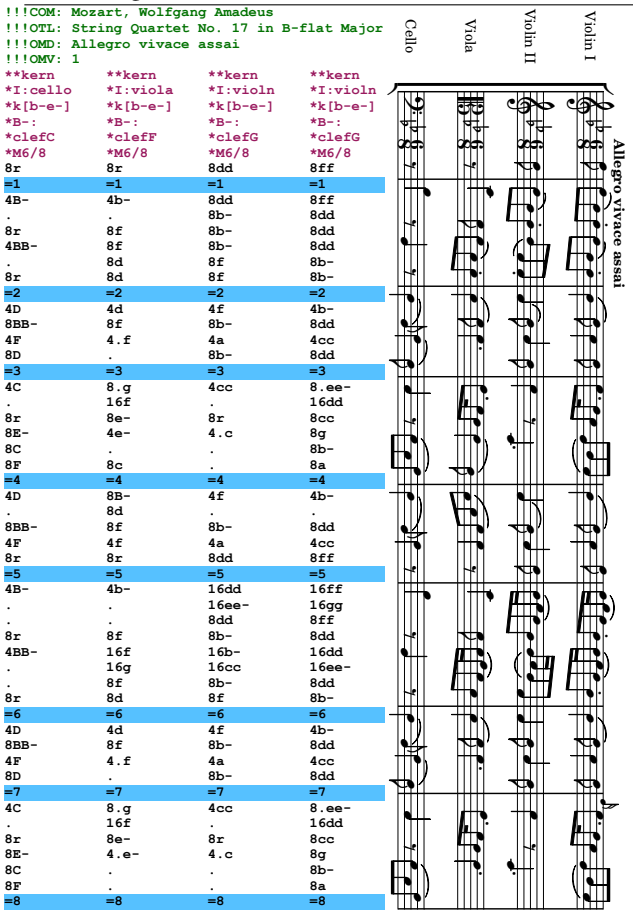


Figure 2: Example of one score in **kern format (left) representing the output of our model and the rendered western music score (right).

2.3 Data Preparation

As previously mentioned, the fact that we do not require data alignments is a clear advantage to easily build the ground truth needed to train our model. However, the majority of scores available in the public domain are usually in printed form, so we cannot automatically obtain the symbolic representation we need unless we make use of an Optical Music Recognition system. A lot of progress has been made in this area of study but unfortunately it is still insufficient to meet our precision needs, driving us to look for existing text based scores instead. After some analysis of the various types of music encoding formats within reach, we chose the humdrum toolkit [9] due to its versatility to represent polyphonic music.

The humdrum file format is a general-purpose human-readable 2D representation of music information intended to assist music researchers. The columns of the text file, separated by the tab character, represent the sources of information that produce music-related events. The lines of the text file represent the evolution of those events over time. The humdrum syntax defines the skeleton that contains other higher level schemes of music notation, like the **kern format our ground truth is based on. The **kern format is designed to encode the semantics of a western musical score, rather than the visual aspects of its printed

	Chorales	**Quartets**
Number of samples	352	34,512
Total duration	5.79h	20.25h
Max duration	120s	30s
Data Augmentation	No	Yes
Polyphony voices	4	4
Instruments	Pipe organ	Cello Viola Violin Flute
Pitch range	C2-A5	C2-E7
Shortest note	$1/16^{th}$	$1/64^{th}$
Irregular groups	None	Triplets
Tempo	♩ $\approx [60, 70]$	♩ $= [40, 200]$
Vocabulary Size	99	143
Train-test split %	80/20	70/30
Batch size	4	16

Table 1: Summary of the datasets' characteristics.

realization, matching nicely with the purpose of this work.

An example of a music excerpt encoded in **kern notation is shown in Figure 2 along with its associated sheet music excerpt. In this format, columns are called spines and they are associated with instruments, just like a pentagram in western sheet music. Spines may contain one single sound event or the combination of various sound events with the same canonical duration, namely a chord. Spines can also be split into two spines when two independent voices (excluding chords) occur for the same instrument. The newly created spine can be rejoined back to the original spine when the extra voice is no longer needed. This level of flexibility gives almost no restrictions to the kind of music it can support, making **kern a good candidate to endure future work.

We created two datasets out of the **kern files available in the humdrum-data repository [18]: the chorales dataset, containing 370 chorales of Bach, and the quartets dataset, containing most of the string quartets of Haydn, Mozart and Beethoven. In the chorales dataset we take each chorale as one training sample, and we use audio from expressive MIDI files synthesized with a high quality pipe organ soundfont [7]. As we did not synthesize the audio, we had to manually remove repetitions to ensure that samples are not unnecessarily long. In the quartets dataset we randomly split the scores in fragments of 3-6 measures each, and we synthesized the corresponding MIDI file obtained from the hum2mid tool, which converts **kern to MIDI using dynamic spines and articulation marks when available in the original **kern file. We removed grace notes and ornaments from the score as they cannot be properly synthesized. We also removed split spines and upper notes of all chords to ensure no more than 4 simultaneous voices were present at any given time. Samples with double dots, double sharps or double flats are out of the scope of this work and therefore discarded. On the training set only, we allow overlapping of fragments as a means of data augmentation technique. Table 1 summarizes the main characteristics of both datasets used in this work.

Figure 1 depicts the data acquisition pipeline we implemented to build our ground truth. The major inconvenience

Largo Assai	40	Allegro Moderato	120
Largo	50	Poco Allegro	124
Poco Largo	60	Allegro	130
Adagio	71	Molto Allegro	134
Poco Adagio	76	Allegro Assai	138
Andante	92	Vivace	150
Andantino	100	Allegro Vivace	160
Menuetto	112	Allegro Vivace Assai	170
Moderato	114	Poco Presto	180
Poco Allegretto	116	Presto	186
Allegretto	118	Presto Assai	200

Table 2: List of metronome markings chosen for classical music tempo annotations, given in number of quarter notes per minute.

was dealing with multiple errors present in the **kern files, such as invalid ties, wrong canonical duration of notes and rests, and missing metronome markings. While these errors do not prevent musical analysis of the scores, they become noisy labels that hinder our training process. For that reason, we had to revise all the scores manually to correct these errors and label the missing metronome markings, with the help of existing tempo annotations and the conversion shown in Table 2. Adding metronome markings ensured the synthesized audio perform at reasonable speeds according to the composer's intention. Additionally, a random scaling factor in the $\pm 6\%$ range was applied to each metronome marking to ensure tempo variability in all training samples.

The resulting **kern scores after the preprocessing stage were then encoded in a special symbolic notation intended to reduce the number of characters and ease the training process. Accordingly, each canonical duration for notes and rests including their dotted version were encoded with just one symbol of the vocabulary. Likewise, note pitches were also encoded with one symbol condensing name and octave. In our **kern dataset, barlines are always repeated for all spines, so only one barline is maintained in the output representation referring to all spines. The rest of characters are preserved in the output representation in the same way, i.e. tabs, new lines, tie symbols, fermatas and the "dot" character, which indicates that the previous note/rest still affects the current row.

3. METHOD

Once the input and output representations are defined, we can formulate the A2S task as retrieving the most likely sequence of score symbols $\hat{\mathbf{s}}$ given an audio file $\mathbf{x} \in \mathcal{X}$:

$$\hat{\mathbf{s}} = \arg\max_{\mathbf{s} \in \Sigma^*} P(\mathbf{s}|\mathbf{x}) \tag{1}$$

where Σ represents the set of characters necessary to encode the output as explained in the previous section (for instance, including "tab", "new-line" and "dot", among others). Additionally, Σ includes an *"empty"* symbol, denoted by ϵ, that is necessary to separate two or more instances of the same symbol that occur in consecutive frames.

Following successful approaches in other pattern recognition duties of similar formulation, we address this A2S with a holistic approach based on statistical models. Specifically, for learning the posterior probability provided in Eq. 1, we resort to Convolutional Recurrent Neural Networks (CRNN).

A CRNN is composed of one block of *convolutional* layers followed by another block of *recurrent* layers [19]. The convolutional block is in charge of learning how to extract relevant features from the input and the recurrent layers interpret these features in terms of sequences of musical symbols. The activations in the last convolutional layer can be seen as a sequence of feature vectors representing the input audio file, $\mathbf{x}$. Let W be the width (number of frames) of the input sequence $\mathbf{x}$. The length of the resulting features after the convolutional layer will be $L = \gamma W$, where $\gamma \leq 1$ is implicitly defined by the specific configuration of the convolutional block (which usually includes some type of down-sampling to reduce dimensionality).

The output activations of the convolutional block are then fed to the first layer of the recurrent block, and the activations of its last layer can be considered proper estimates of the posterior probabilities per frame:

$$P(\sigma \mid \mathbf{x}, j), \;\; 1 \leq l \leq L, \;\; \sigma \in \Sigma \tag{2}$$

3.1 Training

Convolutional neural networks can be trained through gradient descent using the well-known *Back Propagation* algorithm. RNN networks can be trained similarly by means of *Back Propagation Through Time* [21]. Therefore both the convolutional and recurrent blocks of a CRNN can be jointly trained by providing audio files annotated at the frame level.

In this work, however, we follow a holistic or "end-to-end" approach, which means that for each audio file we only provide its corresponding target transcript into score symbols, without any kind of explicit information about its segmentation into frames. A CRNN can be uniformly trained without this information by using the so-called Connectionist Temporal Classification (CTC) loss function [6]. The CTC training procedure is a form of Expectation-Maximization, similar to the backward-forward algorithm used for HMM training [16], that distributes the loss among all the frames to maximize Eq. 1 with respect to the ground-truth sequence.

3.2 Decoding

In order to solve Eq. 1, the most likely symbol is computed for each input feature vector of the recurrent block l, also referred as greedy decoding:

$$\hat{\sigma}_l \;=\; \arg\max_{\sigma \in \Sigma} P(\sigma \mid \mathbf{x}, l), \;\; 1 \leq l \leq L \tag{3}$$

Then, a pseudo-optimal sequence of musical symbols is obtained as $\hat{\mathbf{s}} \approx \mathcal{D}(\hat{\sigma})$, where $\hat{\sigma} = \hat{\sigma}_1 \dots \hat{\sigma}_L$ and $\mathcal{D} : \Sigma^\star \rightarrow \Sigma^\star$ is a function which first merges all the consecutive frames with equal symbol, and then deletes all "empty" symbols [6].

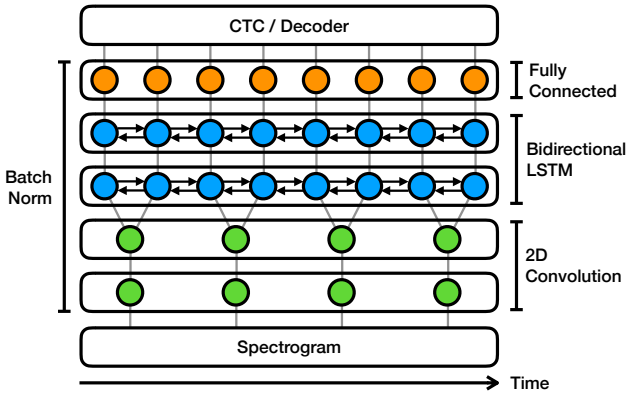


Figure 3: High-level architecture of the Convolutional-Recurrent Neural Network used in our experiments.

3.3 Architecture

The main building blocks of the CRNN considered for our experiments is illustrated in Figure 3.

The first two convolutional layers receive a 2D array containing the audio spectrogram described in section 2.1 and apply 16 filters of 3×3 with a stride of 2 in the frequency axis. We use filter striding to reduce the input dimensionality without the need of pooling layers.

For the Quartets dataset, output frames from the convolutional block are split in half, effectively doubling the number of frames feeding the next recurrent block. This is necessary to comply with the CTC loss function precondition for which the number of input frames must be greater or equal to the number of output symbols. Considering the high number of symbols per second in our sequence-based representation of a polyphonic score of the Quartets dataset, we apply this frame doubling technique at a lesser computational cost than increasing the density of the input spectogram.

The next two recurrent layers are based on Bidirectional Long Short-Term Memory (LSTM) cells, with 1024 hidden units each. The fully connected layer at the end of the recurrent block converts the output per-frame predictions to the size of the output representation vocabulary.

With the purpose of reducing overfitting, Batch Normalization layers [10] are added between any other layer excluding the input and output layers, as well as Dropout layers [20] added after all the convolutional layers and after the last recurrent layer, with a drop probability of 0.1 for the Quartets dataset and 0.2 for the Chorales dataset. A higher drop probability is required for the Chorales dataset since less data is available for training, which increases the risk of overfitting.

4. EXPERIMENTS

To the best of our knowledge, there are few specific evaluation metrics to measure the performance of a notation-level AMT method. In [14] an evaluation metric for note-level AMT is discussed, but it still cannot be directly applied to our task (e.g. we do not have note onsets and offsets). [17] adapts this metric to the A2S task by defining note duration errors instead of onsets/offsets errors. However, we believe this metric is still insufficient to properly evaluate a notation-level AMT method since, for instance, it does not take into account barlines and their effect in subsequent predictions of note durations and ties. The MV2H metric (Multi-pitch detection, Voice separation, Metrical alignment, note Value detection, and Harmonic analysis) was introduced in [13]. This metric is closer to our needs, although its source code uses timing information in seconds that is not provided by our method. We leave it as an open point for future work to establish a proper notation-level AMT metric.

In order to validate our method accuracy during training, we adopt the evaluation metrics from the ASR task as in [17], namely Word Error Rate (WER) and Character Error Rate (CER). They are defined as the number of elementary editing operations (insertion, deletion, or substitution) needed to convert the predicted sequences into the ground-truth sequences, at the word and character level respectively. Even though WER and CER are not specific to AMT, they provide a good indication of how close our score is to the ground-truth score.

In the context of our A2S task, we define words as any group of characters representing notes (including ties), rests and barlines in the output score. The "tab" and "new line" characters act as word separators, and only contribute to the CER calculation.

4.1 Training process

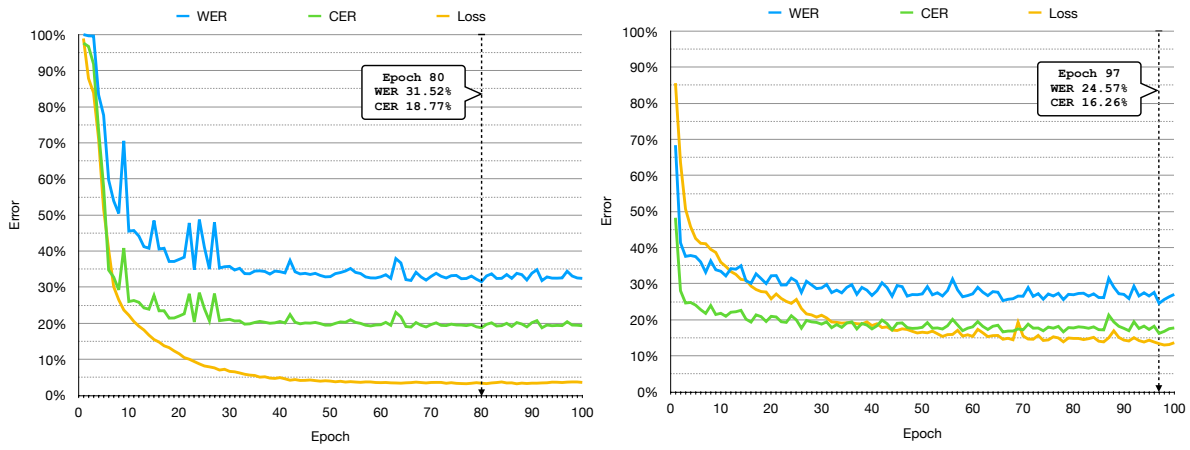
The models were trained for 100 epochs using mini-batch Stochastic Gradient Descent (SGD) optimizer, with Nesterov momentum of 0.9. Our learning rate scheduling consists of 2 cycles of 50 epochs each, starting at 0.0003 and annealing by 1.1 at every epoch. After each epoch, the WER and CER are calculated for the validation set, and the model with the lowest WER is appointed as the best model for testing purposes.

The Chorales dataset, whose samples are full-length chorales, is trained with a batch size of 4. The Quartets dataset, whose samples are small excerpts extracted from the full-length quartets, is trained with a batch size of 16. Figure 4 shows the evolution of the CTC loss, WER and CER at training time on both datasets. Each figure also highlights the epoch where the best model was obtained.

4.2 Results

The best model obtained after the training process is then evaluated against the test set for both the Chorales and Quartets datasets, giving a WER of 30.96% and CER of 18.10% for Chorales, and WER of 21.02% and CER of 13.53% for Quartets.

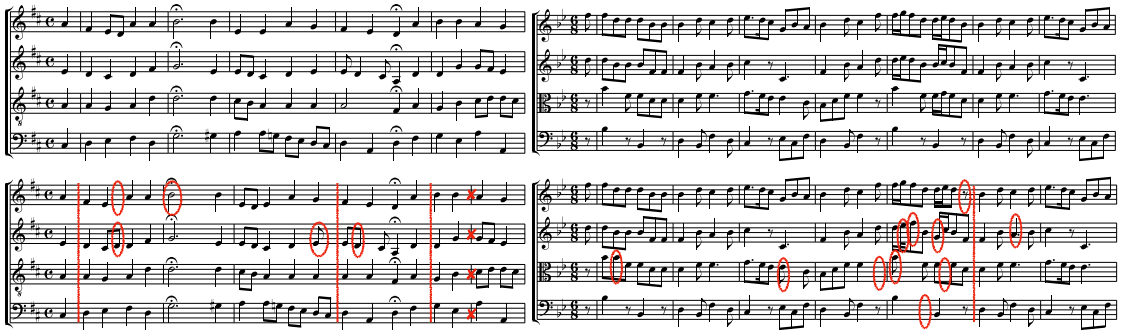
After analyzing all test predictions, we observe that the model occasionally generates sequences that do not comply with the **kern format. Nevertheless, we believe these formatting errors can be solved by providing more samples to the training set or imposing syntax constraints. As shown in Figure 5, most of the errors arise from wrongly



(a) Chorales dataset.

(b) Quartets dataset.

Figure 4: Evolution of loss, validation WER and CER during 100 epochs of training with a) Chorales dataset and b) Quartets dataset. Chorales WER is 30.96% and CER is 18.10%. Quartets WER is 18.10% and CER is 13.53%.



(a) Chorales dataset example

(b) Quartets dataset example

Figure 5: Excerpt of original (top row) and predicted scores (bottom row) from a test sample in a) Chorales dataset, b) Quartets dataset. The differences between original and prediction are highlight in red.

estimated note durations and barlines. Exchanging notes between voices is another common mistake our model makes, specially when voice pitches are too close or even when two voices cross their melodic lines.

The model struggles at predicting ties and triplets, which requires further analysis to determine whether it is related to barline errors, to the output representation format based on **kern, or to the lack of enough samples in the training set (i.e., ties and triplets are very infrequent in our training data compared to other symbols).

5. CONCLUSIONS

In this work, we focus on the A2S task, a hardly explored formulation consisting of extracting a full score from an audio file. Note that A2S resembles what a human would expect to get if it intends to visualize the input audio as a music score (e.g., MusicXML), unlike what most authors consider AMT where the output sequence format is intended to be further processed by a computer (e.g., MIDI).

The proposed methodology which performs the A2S task has the following advantages over other AMT methods: 1) Frame-level alignment of the ground truth is not needed; 2) The end-to-end approach avoids propagating errors from one stage to the other; 3) The output of our model is based on **kern format and can be straightforwardly translated to a valid music score.

We are aware this simplified scenario used for evaluation does not include real audio and some score symbols, but we argue the results provide the basis to open a new path of research towards notation-level AMT.

One of the main limitations of the proposed approach is the maximum-length of input sequences due to memory constraints. For example, this prevents an end-to-end training with complete songs, only allowing fragments of 2 minutes at a maximum on a typical training infrastructure. For this, we plan in a future work to explore other architectures such as the Transformer XL [5], a sequence-to-sequence model that can deal with much longer sequences. Other future works include defining an evaluation metric for A2S and building a dataset from real audio to validate the approach with actual music performances.

6. ACKNOWLEDGMENT

This work was funded by the Spanish Ministerio de Economia, Industria y Competitividad through HISPAMUS project (TIN2017-86576-R).

7. REFERENCES

[1] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Hannun, Billy Jun, Patrick LeGresley, Libby Lin, and Zhenyao Zhu. Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. *Computer Research Repository*, abs/1512.02595, 2015.

[2] Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. Automatic Music Transcription: An overview. *IEEE Signal Procesing Magazine*, 36(1):20–30, 2019.

[3] Ralf Gunter Correa Carvalho and Paris Smaragdis. Towards End-to-end Polyphonic Music Transcription: Transforming Music Audio Directly to a Score. In *IEEE Workshop for Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2017.

[4] Andrea Cogliati, David Temperley, and Zhiyao Duan. Transcribing Human Piano Performances into Music Notation. In *Proc. of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York, USA*, 2016.

[5] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. *Computer Research Repository*, abs/1901.02860, 2019.

[6] Alex Graves, Santiago Fernández, Faustino Gómez, and Jürgen Schmidhuber. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proc. of the 23rd International Conference on Machine Learning*, International Conference on Machine Learning, pages 369–376. ACM, 2006.

[7] Margaret Greentree. Chorale Harmonizations by Bach. `http://sporadic.stanford.edu/Chorales/`.

[8] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. In *Proc. of the 19th International Society for Music Information Retrieval Conference*, pages 50–57, 2018.

[9] David Huron. The Humdrum Toolkit: Software for Music Research. `http://www.humdrum.org`.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proc. of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July*, pages 448–456, 2015.

[11] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. In *Proc. of the 17th International Society for Music Information Retrieval Conference*, pages 475–481, 2016.

[12] Rainer Kelz and Gerhard Widmer. An Experimental Analysis of the Entanglement Problem in Neural-Network-based Music Transcription Systems. In *Audio Engineering Society Conference: AES International Conference on Semantic Audio*, Jun 2017.

[13] Andrew McLeod and Mark Steedman. Evaluating automatic polyphonic music transcription. In *Proc. of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France*, pages 42–49, 2018.

[14] Eita Nakamura, Emmanouil Benetos, Kazuyoshi Yoshii, and Simon Dixon. Towards Complete Polyphonic Music Transcription: Integrating Multi-Pitch Detection and Rhythm Quantization. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2018.

[15] Eita Nakamura, Kazuyoshi Yoshii, and Shigeki Sagayama. Rhythm transcription of polyphonic piano music based on merged-output HMM for multiple voices. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 25(4):794–806, 2017.

[16] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice hall, 1993.

[17] Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. An End-to-End Framework for Audio-to-Score Music Transcription on Monophonic Excerpts. In *Proc. of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France*, 2018.

[18] Craig S. Sapp. humdrum-data. `https://github.com/humdrum-tools/humdrum-data.git`.

[19] Baoguang Shi, Xiang Bai, and Cong Yao. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:2298–2304, 2017.

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[21] Ronald J. Williams and David Zipser. Gradient-based Learning Algorithms for Recurrent Networks and Their Computational Complexity. In Yves Chauvin and David E. Rumelhart, editors, *Backpropagation: theory, architecture and applications*, pages 433–486. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1995.

GENERALIZED METRICS FOR SINGLE-F0 ESTIMATION EVALUATION

Rachel Bittner*, Juan J. Bosch*
Spotify, USA

*Equal contribution

ABSTRACT

Single-f_0 estimation methods, including pitch trackers and melody estimators, have historically been evaluated using a set of common metrics which score estimates frame-wise in terms of pitch and *voicing* accuracy. "Voicing" refers to whether or not a pitch is active, and has historically been regarded as a *binary* value. However, this has limitations because it is often ambiguous whether a pitch is present or absent, making a binary choice difficult for humans and algorithms alike. For example, when a source fades out or reverberates, the exact point where the pitch is no longer present is unclear. Many single-f_0 estimation algorithms select a threshold for when a pitch is active or not, and different choices of threshold drastically affect the results of standard metrics. In this paper, we present a refinement on the existing single-f_0 metrics, by allowing the estimated voicing to be represented as a continuous likelihood, and introducing a weighting on frame level pitch accuracy, which considers the energy of the source producing the f_0 relative to the energy of the rest of the signal. We compare these metrics experimentally with the previous metrics using a number of algorithms and datasets and discuss the fundamental differences. We show that, compared to the previous metrics, our proposed metrics allow threshold-independent algorithm comparisons.

1. INTRODUCTION

Single-f_0 estimation algorithms, including pitch trackers and melody or bass extraction algorithms, predict fundamental frequency (f_0) over time for an audio file. However, there can be time intervals where there is no (target) f_0 value present, for example during silent regions. To account for this, single-f_0 estimation methods additionally estimate the *voicing* over time - i.e. when a given frame contains an active pitch or not. Choosing when the estimated voicing should be active/voiced (1) or inactive/unvoiced (0) often involves choosing a threshold on a confidence value. Single-f_0 estimation algorithms are evaluated by comparing the accuracy of the estimated f_0 and voicing sequence against a reference f_0 and voicing sequence. The choice of threshold to estimate voicing has a critical effect on the resulting metrics; the threshold is often treated as a hyperparameter and is chosen on a validation set. Any confidence information used to determine voicing is discarded and not considered in the evaluation metrics.

The perceptual salience of a pitch is affected by a number of factors, including the volume, the frequency content, the duration and the presence of interference from other sources [12, 18, 21, 27]. In some cases, the brain can perceive a pitch even when the f_0 is not physically present, for example when one short time segment in the middle of a longer pitch sequence is set to be silent [9]. The effect of these factors can be different for each listener, making the task of "objectively" determining if a pitch is present or not a difficult one. Additionally, in polyphonic mixtures, a pitch can be masked by other sources. In the current metrics, algorithms are equally penalized for mistakes on salient and non-salient f_0 values.

We propose a generalization of the existing metrics which (1) allows an algorithm to report voicing as a continuous value (between 0 and 1), and (2) allows frames to be weighted by a reward, which more heavily penalizes mistakes in frames where the energy of the source producing the f_0 is high compared to the rest. These changes remove the need for making a strict decision on whether or not a pitch is present, allowing threshold-independent algorithm comparisons, and allow an optional weighting to be added to reflect frame importance. We also show that when the provided voicing is binary, the proposed metrics are equivalent to the existing metrics. The proposed metrics are to be seen as complementary to the classic ones, which remain useful for measuring performance for applications where a binary threshold is needed in practice. However, binary thresholds are not needed for a number of applications, including pitch informed source separation or melodic similarity.

Additionally, generalized metrics would also help mitigate non-uniformity in decisions made when annotating datasets (e.g. inclusion of delays and reverb as part of the annotation or not), by rewarding correct pitch estimations proportionally to the energy of the pitched signal to be detected. Furthermore, they provide information about the confidence of the estimators, which is useful for many applications.

For reproducibility, the code used for this in this paper is available online [1].

[1] `github.com/juanjobosch/continuousf0eval`

2. VOICING DETERMINATION

Historically, single-f_0 estimation methods need to determine whether a given frame contains a pitch or not. To perform this binary decision, algorithms have commonly used a (static or dynamic) threshold on e.g. energy, salience or pitch likelihood [1, 3, 13–15, 23]. For instance, melody extraction algorithms may exploit pitch contour salience distributions and use heuristics [24], or a threshold on melody contour probabilities produced by a discriminative model [4, 7]. Durrieu et al. [14] first perform source separation on the melodic source, and subsequently estimate the energy of the separated signal frame by frame; frames with energy above a threshold are determined to be voiced. The threshold is empirically selected such that 99.95% of the leading instrument energy is contained in voiced frames. Fuentes et al. [15] also use an energy threshold (of -12dB) on a low-pass filtered separated melody signal. The ideal threshold typically depends on the difference in intensity between melody and accompaniment.

Some methods bypass the use of an explicit threshold and deal with voicing estimation using a classifier, for instance by adding an "unvoiced" class to the set of possible pitch outputs [2]. Other approaches model singing voice detection separately from pitch estimation, and even try to exploit information from neighboring frames (e.g. with LSTMs) for making a decision on the presence of melody on a given frame [17, 23]. Finally, some of the state of the art algorithms provide a measure of confidence on their estimations. However, traditional evaluation metrics do not consider this information.

3. CLASSIC EVALUATION METRICS

Pitch estimation methods have commonly been evaluated using metrics derived from information retrieval, commonly focused on pitch-related accuracy and seldom consider voicing [11, 16]. Melody extraction algorithms are evaluated using similar metrics to pitch estimation, but voicing also takes an important role.

Symbol	Description
n	sample index $\in \{0, \ldots, N-1\}$
f_n	reference frequency (Hz) at sample n
v_n	reference voicing $\in \{0, 1\}$ at sample n
r_n	pitch estimation reward $\in [0, 1]$ at sample n
$\hat{f}_n$	estimate frequency (Hz) at sample n
$\hat{v}_n$	estimate voicing $\in [0, 1]$ at sample n

Table 1. Definition of symbols.

For melody estimation in particular, several metrics are commonly used in the literature [22, 26]. Raw Pitch Accuracy (RPA) and Raw Chroma Accuracy (RCA) measure pitch-related estimation quality. Let the reference and estimate f_0 and voicing sequences be defined as in Table 1. RPA measures the percentage of melody frames in the reference for which the estimated pitch is considered correct (usually within half a semitone of the reference). RCA also measures pitch accuracy, but both estimated and reference pitches are mapped into one octave, forgiving octave mistakes.

$$\mathtt{RPA} = \frac{\sum_{n=0}^{N-1} v_n \mathcal{T}_{\hat{f}_n, f_n}}{\sum_{n=0}^{N-1} v_n}, \; \mathtt{RCA} = \frac{\sum_{n=0}^{N-1} v_n \mathcal{O}_{\hat{f}_n, f_n}}{\sum_{n=0}^{N-1} v_n} \tag{1}$$

where the "correct pitch" indicator function is defined as:

$$\mathcal{T}_{\hat{f}_n, f_n} = \begin{cases} 1 & |d_s(\hat{f}_n, f_n)| \leq 0.5 \\ 0 & |d_s(\hat{f}_n, f_n)| > 0.5 \end{cases} \tag{2}$$

and the difference d_s between two frequency values in semitones is defined as:

$$d_s(\hat{f}_n, f_n) = 12 \log_2 \left(\frac{\hat{f}_n}{f_n} \right) \tag{3}$$

Similarly, the "correct chroma" indicator function is defined as:

$$\mathcal{O}_{\hat{f}_n, f_n} = \begin{cases} 1 & |d_o(\hat{f}_n, f_n)| \leq 0.5 \\ 0 & |d_o(\hat{f}_n, f_n)| > 0.5 \end{cases} \tag{4}$$

and the single-octave pitch difference d_o is defined as:

$$d_o(\hat{f}_n, f_n) = d_s(\hat{f}_n, f_n) - 12 \left\lfloor \frac{d_s(\hat{f}_n, f_n)}{12} + 0.5 \right\rfloor \tag{5}$$

Voicing estimation is evaluated with Voicing Recall rate (VR) and Voicing False Alarm rate (VFA). VR measures the percentage of frames labeled as voiced in the reference which are also estimated as voiced by the algorithm. On the other hand, VFA measures the percentage of frames labeled as un-pitched in the reference that are mistakenly estimated as melody frames by the algorithm.

$$\mathtt{VR} = \frac{\sum_{n=0}^{N-1} \hat{v}_n v_n}{\sum_{n=0}^{N-1} v_n}, \; \mathtt{VFA} = \frac{\sum_{n=0}^{N-1} \hat{v}_n (1 - v_n)}{\sum_{n=0}^{N-1} (1 - v_n)} \tag{6}$$

Finally, Overall Accuracy (OA) is used as a single aggregate measure to evaluate algorithms, as it accounts for both pitch and voicing estimation accuracy. In particular, OA measures the percentage of frames that were correctly labeled in terms of both pitch and voicing.

$$\mathtt{OA} = \frac{1}{N} \sum_{n=0}^{N-1} v_n \hat{v}_n \mathcal{T}_{\hat{f}_n, f_n} + (1 - v_n)(1 - \hat{v}_n) \tag{7}$$

In order to allow each of the metrics to give insights about different aspects of methods, the traditional evaluation methodology allows algorithms to report "negative" pitch values, which are only considered for the computation of RPA and RCA. This was an attempt to evaluate voicing estimation performance separately from pitch estimation performance, and therefore make the result of RPA

Metric	**Classic** ($r_n = v_n$, binary $\hat{v}_n$)	**Generalized** (continuous r_n and $\hat{v}_n$)
RPA	average pitch accuracy in voiced frames	weighted average pitch accuracy in voiced frames
RCA	average chroma accuracy in voiced frames	weighted average chroma accuracy in voiced frames
VR	fraction of voiced frames estimated as voiced	average voicing likelihood in voiced frames
VFA	fraction of unvoiced frames estimated as voiced	average voicing likelihood in unvoiced frames
OA	fraction of frames with correct voicing and pitch	weighted average correctness of each frame

Table 2. Description of the meaning of the metrics in the classic and generalized cases. Note that two possible cases are not described: binary $\hat{v}_n$ with a continuous reward r_n, and continuous $\hat{v}_n$ with $r_n = v_n$.

and RCA independent of the voicing estimation. However, many algorithms do not actually report negative pitches, and furthermore due to the inner functioning of some methods (e.g. Melodia [24]), increasing the number of reported pitches (either positive or negative) not only has an effect on voicing estimation accuracy but also on pitch accuracy.

Other metrics have been proposed to give further insights, such as the continuity of the correctly estimated pitches (either in pitch or chroma), which is relevant for tasks such as automatic transcription, source separation or visualization [8]. Metrics related to user satisfaction have also been studied in the context of melody extraction: different kind of errors have a different impact in the quality perceived when users listen to synthesized melodies that have been extracted automatically [20]. However, most single-f_0 estimation literature does not consider the influence of the energy of the signal under study (or its relation to the accompaniment) in the evaluation. Some exceptions [6, 25] present an evaluation of pitch salience functions, which are commonly correlated to the energy of the signals. Bosch et al. [8] also study the influence of the predominance of the melody over the accompaniment for different algorithms in the context of symphonic music, and monophonic pitch estimators have been evaluated in the presence of different noise levels [16, 28]. However, in the classic single-f_0 estimation metrics, all frames contribute equally to the results, even though in many cases the presence or absence of a (melody) pitch may be unclear for both humans and algorithms.

4. GENERALIZED METRICS

This section presents a generalization of the traditional metrics, in order to deal with the previously introduced limitations: voicing estimates must be binary, and all frames receive equal importance. The proposed metrics (1) allow algorithms to report voicing $\hat{v}_n$ as a *continuous* rather than a binary quantity, representing the likelihood that the frame is voiced, and (2) optionally weight the pitch accuracy in voiced frames using a reward $r_n \in [0, 1]$, allowing mistakes in less important frames to count less than mistakes in important frames.

In the following metrics, we require that (1) $r_n = 0$ if and only if $v_n = 0$, (2) $v_n = 0$ if $f_n = 0$ and (3) $\hat{v}_n = 0$ if $\hat{f}_n = 0$. Note that we may have $\hat{f}_n \neq 0$ and $\hat{v}_n = 0$, allowing the metrics to score pitch accuracy when voicing mistakes are made.

Equation 8 presents the proposed generalization of RPA and RCA, which aggregate the pitch/chroma accuracy proportional to the reward r_n. This makes the generalized metrics more forgiving on unimportant frames, and more demanding on important frames in comparison to previous metrics.

$$\mathtt{RPA} = \frac{\sum_{n=0}^{N-1} r_n \mathcal{T}_{\hat{f}_n, f_n}}{\sum_{n=0}^{N-1} r_n}, \quad \mathtt{RCA} = \frac{\sum_{n=0}^{N-1} r_n \mathcal{O}_{\hat{f}_n, f_n}}{\sum_{n=0}^{N-1} r_n} \tag{8}$$

Generalized versions of VR and VFA remain the same as in Equation 6, however $\hat{v}_n$ need not be binary. In both cases, VR is simply the average of $\hat{v}_n$ in voiced frames ($v_n = 1$), and similarly VFA is the average of $\hat{v}_n$ in unvoiced frames ($v_n = 0$).

Finally, we propose a generalized version of OA in Equation 9, which scores voiced frames proportionally to $\hat{v}_n$, weighted by r_n, and scores unvoiced frames proportionally to $1 - \hat{v}_n$. Let $V = \sum_{n=0}^{N-1} v_n$, the number of voiced frames in the reference annotations. The generalized OA becomes:

$$\mathtt{OA} = \frac{V \frac{\sum_{n=0}^{N-1} r_n \hat{v}_n T_{\hat{f}_n, f_n}}{\sum_{n=0}^{N-1} r_n} + (N - V) \frac{\sum_{n=0}^{N-1} (1 - v_n)(1 - \hat{v}_n)}{\sum_{n=0}^{N-1} (1 - v_n)}}{N} \tag{9}$$

In this generalized OA, voicing "mistakes" are penalized according to the confidence of the estimator, which is softer than in the binary case where mistakes are "all or nothing". When $r_n = v_n$ (equal reward in all voiced frames) and $\hat{v}_n$ is binary, each of the generalized metrics is equivalent to the metrics defined in the binary case. For RPA, RCA, VR and VFA the equivalence is straightforward. For OA, substituting r_n by v_n, plugging in the given equation for V, and simplifying the resulting quantity shows equivalence. Table 2 gives summaries of the metrics in the classic and generalized cases.

4.1 Behavior on Artificial Examples

Figure 1 shows the behavior of the proposed metrics for a few simple examples. For instance, the different results obtained in plots (a), (b), (e) and (f) show that if the pitches are correct, VR and OA get the best results with highly confident estimations. Plot (a) also shows that errors in the

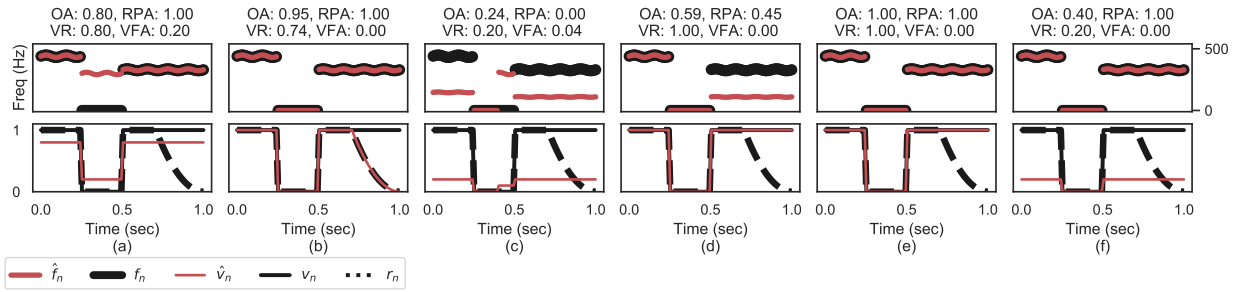


Figure 1. Artificial examples of different combinations of f_n, r_n, $\hat{f}_n$, and $\hat{v}_n$, and the behavior of the generalized metrics.

voicing estimation are less penalized if the estimate confidence is low. In this example, VFA is relatively low (0.2), while if the algorithm had used a very low threshold to determine a binary voicing value $\hat{v}_n$, VFA would be equal to 1 (the worst possible score), since all unvoiced frames would have been estimated as voiced. Plot (d) shows the effect of having perfect voicing estimation but incorrect pitch estimation – the estimator is penalized in OA and RPA. Plot (c) shows a completely wrong pitch estimation, which gets a small score for voicing recall, due to the low estimated confidence. In the same plot, VFA is very low due to the correctly identified unvoiced frames, and the errors between 0.4 and 0.5 s are not heavily penalized due to the low reported confidence.

5. COMPUTING PITCH ESTIMATION REWARDS

In order to create the pitch estimation reward (r_n) for single f_0 datasets, we propose the computation of Root-Mean-Square (RMS) energy in frames over time. The first step is to compute the RMS of the source producing the f_0 (RMS_{f_0}) and the RMS of the mixture (RMS_m) in frames. The second step deals with a frame-wise normalization, in order to obtain a reward signal: $r_n = (\text{RMS}_{f_{0_n}})/(\max(\text{RMS}_{f_0})+\text{RMS}_{m_n})$, where n corresponds to the index of the frame. In frames where there is no pitch annotation ($f_n = 0$), we set $r_n = 0$, as illustrated in Figure 2.

A mismatch between the energy of the melodic source and the voicing derived from pitch annotations (if the f_0 is non zero) could happen due to several factors. One of them is the fact that there may be energy due to a melodic instrument but actually no pitch, for instance in transient percussive sounds, or with unpitched vocal sounds (e.g., many of the consonants). Another possible factor is that the procedure followed during the annotation did not consider echos or reverberation, while they might be clearly present in the signal.

5.1 Isolated Sources

For datasets where isolated sources are available, we can simply compute the frame-wise RMS of the signals over time. For instance, in a melody extraction dataset, we would use the RMS of the source playing the melody in each frame to derive RMS_{f_0}. Note that this is compatible with multiple melody definitions, even allowing different

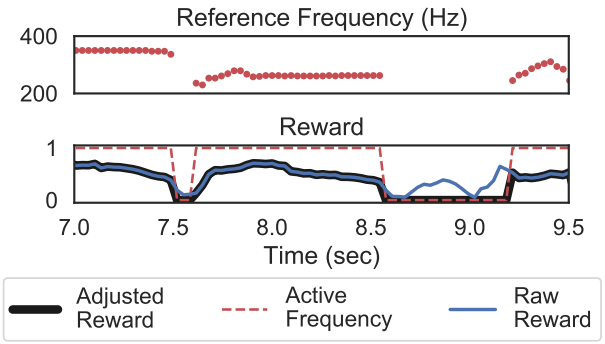


Figure 2. The reference frequency and the estimated reward values. Frames where no reference frequency is provided may have non-zero reward estimates (in blue) - in these cases the reward is set to 0 (in black).

instruments to play the melody sequentially in a given music excerpt [5,8]. RMS_m is computed from the instruments which are not playing the melody pitch in each frame, and the reward computed following the methodology from section 5. A simpler case corresponds to monophonic pitch estimation datasets, where RMS_m is zero, so the raw reward in each frame is equal to the RMS_{f_0}, normalized by its maximum RMS value in the example.

5.2 Sources in Polyphonic Mixtures

When isolated sources are not available, it is more difficult to compute RMS_{f_0} and RMS_m. For these cases, we propose the use of pitch-informed source separation [8, 14] in order to obtain an estimate of the energy of the source and accompaniment. We test the effectiveness of this approach using the iKala dataset by comparing the difference in the reference reward when computing it using isolated vocals, and using the results of pitch informed source separation on the mixture signal.

Figure 3 shows the results of "Melody-A" (see Section 6, for all metrics, with the confidence computed both using source separation and in the ideal case (having access to the isolated sources). As we can observe, VR and VFA have the same values, and OA, RPA, and RCA present some small differences but which are statistically significant, according to a paired t-test ($\alpha = 0.05$). However, in Figure 3 (right) we see that for each metric, the distribution of differences in comparison to "Melody-B" is very

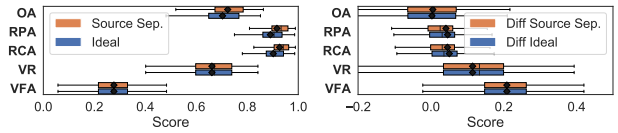


Figure 3. (Left) Metrics for "Melody-A" on iKala for confidence computed using both source separation and in the ideal case. (Right) The difference in score between "Melody-A" and "Melody-B" per track for both confidence measures on iKala.

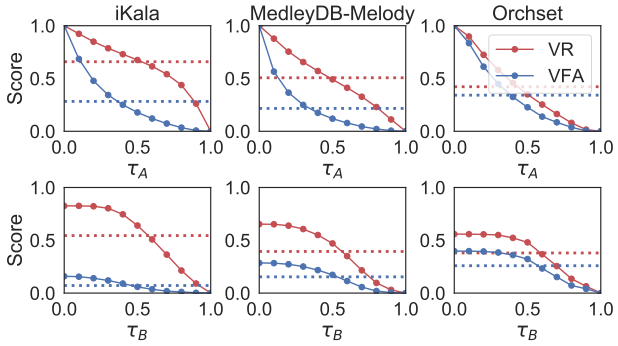


Figure 4. Classic voicing metrics (VR - red, VFA - blue) as a function of threshold. Dashed horizontal lines show the value of the generalized metrics computed with continuous $\hat{v}_n$. (Top) Threshold τ_A for "Melody-A" (Bottom) Threshold τ_B for "Melody-B".

similar when using confidence computed on both the ideal case and with source separation. This suggests that using source separation as a proxy to get the confidence measure would not have an impact on the ranking of algorithms, and therefore the methodology proposed would be useful to compare different algorithms. We leave the improvement of pitch-informed source separation for obtaining a better reward, i.e. more similar to the values obtained if the isolated sources were available, as future work.

6. PROPOSED METRICS ON REAL DATA

In order to show the behavior of the metrics with real data and algorithms, we create variants of four established algorithms: two monophonic pitch estimators "Pitch-A" (based on CREPE [16]) and "Pitch-B" (based on pYIN [19]) and two melody extraction algorithms "Melody-A" (based on Deep Salience [3]) and "Melody-B" (based on Melodia [24]). Note that the main objective is not to actually evaluate/compare these algorithms, but show the behaviour and give further insights about the proposed metrics. Therefore the arbitrarily taken decisions about the estimators such as the normalization, or using default parameters, should not be regarded as important.

In order to test our metrics, we need each algorithm to produce a continuous voicing estimate $\hat{v}_n$. "Pitch-A" and "Melody-A" predict confidence values as part of the algorithm, which we use directly as $\hat{v}_n$. "Pitch-B" and "Melody-B" do not directly predict confidence values, but determine which frames are voiced and unvoiced using thresholds on signals computed internally. We derive a value of $\hat{v}_n$ for these algorithms using normalized versions of these signals (the maximum probability of the pitch candidates for "Pitch-B", and the contour confidence measure for "Melody-B"). Note that not all algorithms currently provide a confidence value as an output, but all of them determine voicing at some level, and the steps used to make this decision can typically be used to create a measure of voicing confidence.

We use three melody extraction datasets in our experiments: iKala [10], MedleyDB [5] and Orchset [8]. iKala comprises 252 30-second excerpts sampled from 206 songs. MedleyDB contains 108 melody annotated files, which are mostly full-length songs between 3 and 5 minutes long, and cover a variety of instrumentation and genres. For our experiments, we use the melody 2 definition: the f_0 curve of the predominant melodic line drawn from multiple sources. Finally, Orchset contains 64 short audio excerpts (between 10 and 30 s.) of symphonic music. For pitch tracking, we use a dataset derived from MedleyDB, with 103 tracks of solo, monophonic instruments. In the two datasets for which we have isolated sources readily available, MedleyDB-Pitch and iKala, we compute the reference reward using the method described in Section 5 using a hop size of 256 and a window size of 4096 for a sample rate of 44100 Hz.

6.1 Voicing Estimation Metrics

We first examine the difference in the generalized versus the classic metrics for VR and VFA. In the classic metrics, the choice of voicing threshold has a major effect on VR and VFA. Figure 4 shows the classic metrics as a function of the voicing threshold as dots for VR (red) and VFA (blue) for "Melody-A" and "Melody-B" on the three melody datasets. The dashed horizontal lines show the value of the generalized metric, which is computed independently of the threshold. We see that the value of the generalized metrics is close to the average value of the metrics for all thresholds.

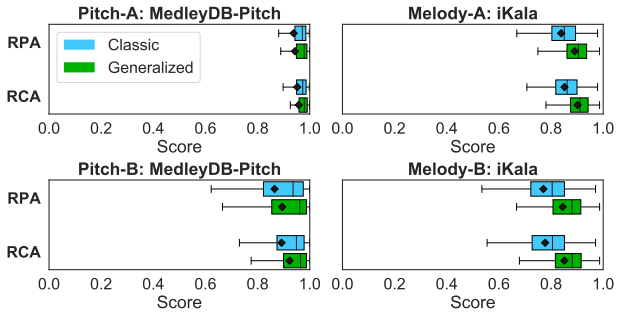


Figure 5. Generalized vs Classic RPA and RCA for four algorithms. r_n is used for the reference datasets. Boxplots show statistics across tracks for each metric.

6.2 Pitch Accuracy Metrics

Figure 5 shows the generalized and classic RPA and RCA on iKala (for "Melody-A" and "Melody-B") and MedleyDB-Pitch (for "Pitch-A" and "Pitch-B"). We can see that the values of the generalized metrics are higher in all cases, which confirms that algorithms commonly make more errors when the reference reward is lower (more difficult cases). The difference between the classic and generalized metrics is larger on "Melody-B" (+0.08 on average for RPA) than "Melody-A" (+0.05 on average for RPA), which suggests that "Melody-A" is less prone to pitch estimation errors when the melody is less predominant.

6.3 Overall Accuracy

Finally, we compare the generalized metrics with the classic metrics for OA. It is most often used as a single measure to compare the performance of two algorithms, but in the classic metric, the choice of each algorithm's voicing threshold can change the relative ranking of OA. In Figure 6, the middle and right columns show the classic OA as a function of threshold for two algorithms, and the left column shows the relative ranking of the classic OA for each combination of thresholds; when a cell is red, the algorithm in the middle gets a higher value for this metric, and vise versa when a cell is blue. We see that for all datasets, a pair of threshold values can be chosen which rank one algorithm higher than the other. This makes the comparison of two algorithms in terms of the classic OA highly dependent on the choice of threshold.

We see that when algorithms are ranked based on the generalized OA, the algorithm which is ranked higher is always the algorithm which is also more often ranked higher for the classic OA (i.e. the dominant color in Figure 6, left). The ordering of the generalized OA is also consistent with the highest possible value of the classic OA (the "star" marker in Figure 6). This suggests that the generalized OA provides a threshold independent way to fairly rank algorithms. Comparing the generalized metrics with and without r_n (Figure 6, horizontal dashed and solid lines), we see that overall the behavior of OA is similar, but harsher when r_n is not used.

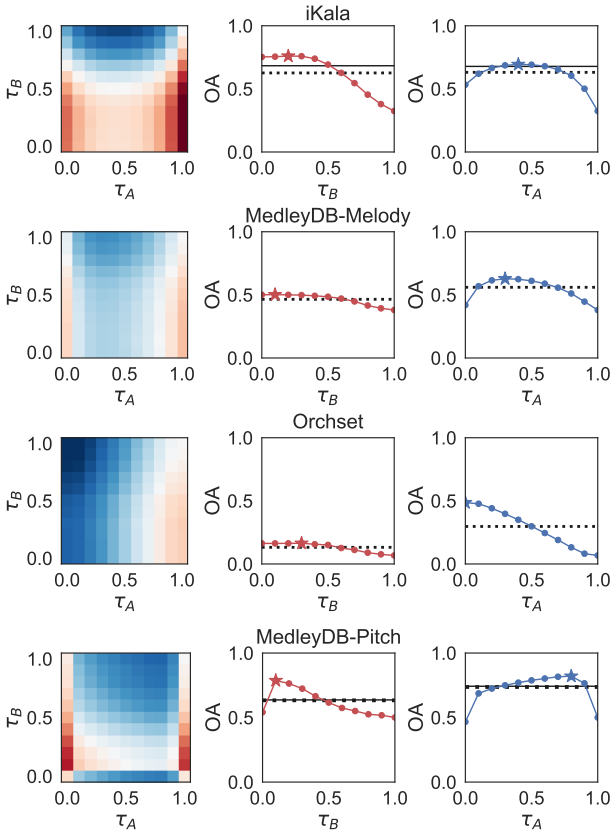


Figure 6. (Left column) difference in overall accuracy between two algorithms. Red: algorithm B gets a higher OA, Blue: algorithm A gets a higher OA, White: They are the same. (Middle and Right Columns) OA as a function of threshold for "Melody-B" (middle) and "Melody-A" (right) for rows 1-3, and for "Pitch-B" (middle) and "Pitch-A" (right) in row 4. Dashed lines show the generalized OA computed with $r_n = v_n$ and continuous $\hat{v}_n$, solid lines show the generalized OA computed with continuous r_n and continuous $\hat{v}_n$. Solid lines are missing for two datasets because the isolated melody sources are not available so we cannot accurately compute continuous r_n.

7. CONCLUSIONS

This paper presents a generalization of traditional single-f_0 estimation metrics, which allows estimators to provide a continuous voicing estimate and introduces a weighting on pitch accuracy. We perform an experimental comparison of the proposed metrics using both monophonic pitch estimators and melody extraction algorithms and show that the generalized metrics provide a threshold-independent way of comparing algorithms. Additionally, we propose a methodology for the annotation of the reference reward r_n based on the energy of the isolated sources and also propose a promising variant for the case when only polyphonic mixtures are available, based on pitch-informed source separation. One of the limitations of the proposed method for estimating the reference reward is that it does not explicitly consider aspects related to pitch perception, which we leave for future work. Finally, the proposed evaluation framework could also be extended to multiple pitch estimation metrics. The concept of confidence and reward, in this case, would be related to each of the individual pitches present, and the methodology would still hold.

While this paper focuses on the generalization of the classic metrics, we also foresee the creation of new metrics, including the adaptation of metrics from the Information Retrieval literature (such as the ROC-AUC score). The current work only considers voicing confidence for estimators and a kind of "pitch confidence" for references, however pitch confidence for estimators and voicing confidence for references could also be incorporated. Future work could also experiment with metrics based on different types of rewards r_n, such as metrics that examine pitch accuracy in difficult frames.

8. REFERENCES

[1] V. Arora and L. Behera. On-line melody extraction from polyphonic audio using harmonic cluster tracking. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(3):520–530, March 2013.

[2] D. Basaran, S. Essid, and G. Peeters. Main melody extraction with source-filter nmf and crnn. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[3] R. Bittner, B. McFee, J. Salamon, P. Li, and J.P. Bello. Deep salience representations for f0 estimation in polyphonic music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, October 2017.

[4] R. Bittner, J. Salamon, S. Essid, and J. Bello. Melody extraction by contour classification. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 500–506, Oct. 2015.

[5] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. Bello. Medleydb: a multitrack dataset for annotation-intensive mir research. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 155–160, Oct. 2014.

[6] J. Bosch. *From heuristics-based to data-driven audio melody extraction*. PhD thesis, Universitat Pompeu Fabra, June 2017.

[7] J. Bosch, R. M. Bittner, J. Salamon, and E. Gómez. A comparison of melody extraction methods based on source-filter modelling. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 571–577, Aug. 2016.

[8] J. Bosch, R. Marxer, and E. Gómez. Evaluation and combination of pitch estimation methods for melody extraction in symphonic classical music. *Journal of New Music Research*, 45(2):101–117, 2016.

[9] A. Bregman. *Auditory scene analysis: The perceptual organization of sound*. MIT press, 1994.

[10] T. Chan, T. Yeh, Z. Fan, H. Chen, L. Su, Y. Yang, and R. Jang. Vocal activity informed singing voice separation with the ikala dataset. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 718–722. IEEE, 2015.

[11] H. Cheveigné, A.and Kawahara. Comparative evaluation of f0 estimation algorithms. In *Seventh European Conference on Speech Communication and Technology*, 2001.

[12] JM. Doughty and WR. Garner. Pitch characteristics of short tones. ii. pitch as a function of tonal duration. *Journal of Experimental Psychology*, 38(4):478, 1948.

[13] K. Dressler. Towards Computational Auditory Scene Analysis: Melody Extraction from Polyphonic Music. In *Proceedings of the International Symposium on Computer Music Multidisciplinary Research (CMMR)*, pages 319–334, 2012.

[14] J. Durrieu, G. Richard, B. David, and C. Févotte. Source/filter model for unsupervised main melody extraction from polyphonic audio signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):564–575, 2010.

[15] B. Fuentes, A. Liutkus, R. Badeau, and G. Richard. Probabilistic model for main melody extraction using constant-Q transform. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5357–5360. IEEE, 2012.

[16] J. Kim, J. Salamon, P. Li, and J.P. Bello. Crepe: A convolutional representation for pitch estimation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 161–165. IEEE, 2018.

[17] S. Kum and J. Nam. Joint detection and classification of singing voice melody using convolutional recurrent neural networks. *Applied Sciences*, 9(7):1324, 2019.

[18] JCR Licklider. Influence of phase coherence upon the pitch of complex, periodic sounds. *The Journal of the Acoustical Society of America*, 27(5):996–996, 1955.

[19] M. Mauch and S. Dixon. Pyin: A fundamental frequency estimator using probabilistic threshold distributions. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 659–663, May 2014.

[20] B. Nieto. Addressing user satisfaction in melody extraction. Master's thesis, Universitat Pompeu Fabra, 2014.

[21] C.J. Plack, A.J. Oxenham, and R. Fay. *Pitch: neural coding and perception*, volume 24. Springer Science & Business Media, 2006.

[22] G. Poliner, D. Ellis, A. Ehmann, E. Gómez, S. Streich, and B. Ong. Melody transcription from music audio: Approaches and evaluation. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(4):1247–1256, 2007.

[23] F. Rigaud and M. Radenen. Singing voice melody transcription using deep neural networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 737–743, 2016.

[24] J. Salamon and E. Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio. Speech. Lang. Processing*, 20(6):1759–1770, 2012.

[25] J. Salamon, E. Gómez, and J. Bonada. Sinusoid extraction and salience function design for predominant melody estimation. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 73–80, 2011.

[26] J. Salamon, E. Gómez, D. Ellis, and G. Richard. Melody extraction from polyphonic music signals: Approaches, applications, and challenges. *IEEE Signal Processing Magazine*, 31(2):118–134, 2014.

[27] P. Singh. Perceptual organization of complex-tone sequences: A tradeoff between pitch and timbre? *The Journal of the Acoustical Society of America*, 82(3):886–899, 1987.

[28] P. Verma and RW. Schafer. Frequency estimation from waveforms using multi-layered neural networks. In *Proceedings of Interspeech*, pages 2165–2169, 2016.

LEARNING DISENTANGLED REPRESENTATIONS OF TIMBRE AND PITCH FOR MUSICAL INSTRUMENT SOUNDS USING GAUSSIAN MIXTURE VARIATIONAL AUTOENCODERS

Yin-Jyun Luo[1,2] **Kat Agres**[2,3] **Dorien Herremans**[1,2]
[1] Singapore University of Technology and Design
[2] Institute of High Performance Computing, A*STAR, Singapore
[3] Yong Siew Toh Conservatory of Music, National University of Singapore
yinjyun_luo@mymail.sutd.edu.sg,kat_agres@ihpc.astar.edu.sg,dorien_herremans@sutd.edu.sg

ABSTRACT

In this paper, we learn disentangled representations of timbre and pitch for musical instrument sounds. We adapt a framework based on variational autoencoders with Gaussian mixture latent distributions. Specifically, we use two separate encoders to learn distinct latent spaces for timbre and pitch, which form Gaussian mixture components representing instrument identity and pitch, respectively. For reconstruction, latent variables of timbre and pitch are sampled from corresponding mixture components, and are concatenated as the input to a decoder. We show the model's efficacy using latent space visualization, and a quantitative analysis indicates the discriminability of these spaces, even with a limited number of instrument labels for training. The model allows for controllable synthesis of selected instrument sounds by sampling from the latent spaces. To evaluate this, we trained instrument and pitch classifiers using original labeled data. These classifiers achieve high F-scores when tested on our synthesized sounds, which verifies the model's performance of controllable realistic timbre/pitch synthesis. Our model also enables timbre transfer between multiple instruments, with a single encoder-decoder architecture, which is evaluated by measuring the shift in the posterior of instrument classification. Our in-depth evaluation confirms the model's ability to successfully disentangle timbre and pitch. [1]

1. INTRODUCTION

A disentangled feature representation is defined as having disjoint subsets of feature dimensions that are only sensitive to changes in corresponding factors of variation from observed data [2, 27, 32]. Deep generative models [13, 19, 25, 33] have been exploited to learn disentangled representations in different domains. In the visual domain, studies are focused on learning independent representations for data generative factors such as identity and azimuth [5, 14, 26]. In natural language generation, efforts have been made to generate texts with controlled sentiment [10, 18, 36]. Also in the speech domain, we have witnessed successful attempts in controllable speech synthesis by disentangling factors such as speaker identity, speed of speech, emotion, and noise level [15, 17, 35]. There has been relatively little research on learning disentangled representations for music. In this paper, we disentangle the pitch and timbre of musical instrument sound recordings.

Pitch and timbre are essential properties of musical sounds. Given that one pitch can be played with different instruments, we assume they can be separated. From the perspective of music analysis, disentangled representations of pitch and timbre can be regarded as timbre- and pitch-invariant features which could be exploited for downstream tasks [29,30]. From the synthesis point of view, disentangled representations enable the generation of musical notes with identical pitches (timbres) and different timbres (pitches). Recently, Hung *et al.* presented the first attempt to learn disentangled representations of pitch and timbre for synthesized music by using frame-level instrument and pitch labels based on encoder-decoder networks [21]. Even though the authors managed to change instrumentation to some extent without affecting pitch structure, the approach was restrictive, as it worked with MIDI-synthesized audio and relied on clean frame-level labels, which are scarce to find. Disentangled representations allow for several applications, including music style transfer. Brunner *et al.* proposed a model based on variational autoencoders (VAEs) [25] to generate music with controllable attributes [4]. While genre was factorized by an auxiliary classifier, other musical properties were entangled. Besides the aforementioned models based on MIDI, research on audio has focused on translating between different domains of instrumentation [3,7,20,28]. None of them, however, has addressed learning disentangled latent variables of both pitch and timbre.

This research distinguishes itself from others by disentangling instrument sounds into distinct sets of latent variables (i.e., pitch and timbre), with a framework based on Gaussian Mixture VAEs (GMVAEs). We model the gener-

[1] Example audio files and code at `http://bit.ly/2Dbyt9j`

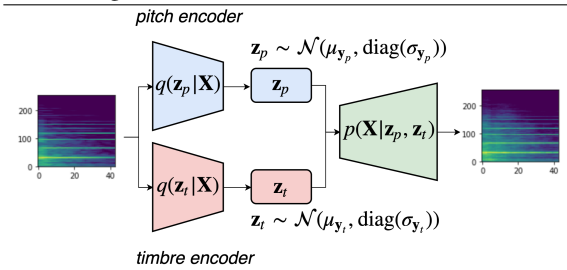


Figure 1. The proposed framework includes separate encoders for pitch and timbre, and a shared decoder.

ative process of an isolated musical note by independently sampling pitch and timbre (instrument) categorical variables. Note that the two factors are actually dependent in a sense that range of pitch is instrument-dependent, however, we verify the model's capability to disentangle them under this simplified assumption of independence. Conditioned on these categorical variables, Gaussian-distributed latent variables are then sampled that characterize variation in the sampled pitch and instrument, respectively. Finally, the data are generated conditioned on the two latent variables. We favor the proposed framework over vanilla VAEs [8, 9] for its more flexible latent distribution compared to a standard Gaussian. In addition, it allows for unsupervised or semi-supervised clustering, which can learn interpretable mixture components and corresponding Gaussian parameters. More importantly, such a framework facilitates the applications in this research: controllable synthesis of instrument sounds, and many-to-many transfer of instrument timbres. Our proposed framework differs from previous studies on timbre transfer, in that we achieve transfer between *multiple instruments* without training a domain-specific decoder for each instrument (e.g. [28]), and we infer both the pitch and timbre latent variable without requiring categorical conditions of source pitch and instrument as in [3]. We evaluate our model by visualizing both the latent space and the synthesized spectrograms, and explore the classification F-scores of classifiers trained in an end-to-end fashion. The results confirm the model's ability to learn disentangled pitch and timbre representations. The rest of the paper is organized as follows: in Section 2, we discuss the proposed framework, and Section 3 describes the dataset and experimental setup. Experiments and results are reported in Section 4. We conclude our work and provide future directions in Section 5.

2. PROPOSED FRAMEWORK

In this section, we briefly describe VAEs and GMVAEs, and elaborate on the proposed framework and architecture.

2.1 Gaussian Mixture Variational Autoencoders

VAEs [25] are unsupervised generative models that combine latent variable models and deep learning [12]. We denote the observed data and the latent variables respectively by $\mathbf{X}$ and $\mathbf{z}$. A graphical model, corresponding to $\mathbf{z} \rightarrow \mathbf{X}$, is trained by maximizing the lower bound of the log marginal likelihood $p(\mathbf{X})$. The intractable posterior $p(\mathbf{z}|\mathbf{X})$ is approximated by introducing a variational distribution $q(\mathbf{z}|\mathbf{X})$ parameterized with neural networks. In regular VAEs, a common choice for the prior distribution $p(\mathbf{z})$ is an isotropic Gaussian, which encourages each dimension of the latent variables to capture an independent factor of variation from the data, and results in a disentangled representation [14]. Such a unimodal prior, however, does not allow for multi-modal representations. GMVAEs [6,22,24] extend the prior to a mixture of Gaussians, and assume the observed data are generated by first determining the mode from which it was generated, which corresponds to learning a graphical model $\mathbf{y} \rightarrow \mathbf{z} \rightarrow \mathbf{X}$. This introduces a categorical variable $\mathbf{y}$, and $q(\mathbf{y}|\mathbf{X})$, which infers the classes of data. This enables semi-supervised learning [24] and unsupervised clustering [6, 22] in deep generative models. In the speech domain, Hsu *et al.* used two mixture distributions to separately model the supervised speaker and unsupervised utterance attributes, which allowed for extra flexibility in conditional speech generation [17]. We build upon this idea to learn separate latent distributions to represent the pitch and timbre of musical instrument sounds. More importantly, to facilitate downstream creative applications such as controllable synthesis and instrument timbre transfer in music, we propose to model supervised pitch representations and semi-supervised timbre representations, with labels of pitch and instrument identity. As such, the mixture components in latent space of pitch and timbre can be clearly interpreted as the classes, i.e., pitch and instrument identity.

2.2 Model Formulation

The latent variables of pitch and timbre for an isolated musical note $\mathbf{X}$ are denoted as $\mathbf{z}_p$ (*pitch code*) and $\mathbf{z}_t$ (*timbre code*), respectively. To represent Gaussian mixture latent distributions, two categorical variables are introduced: an M-way categorical variable $\mathbf{y}_p$ for pitch, where M is the number of recorded pitches in the dataset, and a K-way categorical variable $\mathbf{y}_t$ for timbre, where K is the number of instrument classes. We consider $\mathbf{y}_p$ to be observed (fully supervised), which assumes the availability of pitch labels during training, and is reasonable as we model isolated instrument sounds in this research. For $\mathbf{y}_t$, we investigate both unsupervised and semi-supervised learning, i.e., using varying numbers of instrument labels for training. It is shown in Section 4 that our model can efficiently leverage the limited number of labels. Without loss of generality, we denote $\mathbf{y}_t$ as unobserved (unsupervised) as in [17]. The joint probability of $\mathbf{X}$, $\mathbf{y}_t$, $\mathbf{z}_t$ and $\mathbf{z}_p$ is written as:

$$\begin{aligned} p(\mathbf{X}, \mathbf{y}_t, \mathbf{z}_t, \mathbf{z}_p|\mathbf{y}_p) = \\ p(\mathbf{X}|\mathbf{z}_p, \mathbf{z}_t)p(\mathbf{z}_p|\mathbf{y}_p)p(\mathbf{z}_t|\mathbf{y}_t)p(\mathbf{y}_t), \end{aligned} \tag{1}$$

where $p(\mathbf{y}_t)$ is uniform-distributed, i.e., we do not assume to know the instrument distribution in the dataset. Both the conditional distributions $p(\mathbf{z}_p|\mathbf{y}_p) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{y}_p}, \mathrm{diag}(\boldsymbol{\sigma}_{\mathbf{y}_p}))$ and $p(\mathbf{z}_t|\mathbf{y}_t) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{y}_t}, \mathrm{diag}(\boldsymbol{\sigma}_{\mathbf{y}_t}))$ are assumed to be diagonal-covariance Gaussians with learnable means and constant variances. This amounts to both the marginal prior $p(\mathbf{z}_p)$ and $p(\mathbf{z}_t)$ being Gaussian mixture models

(GMMs) with diagonal covariances. Ideally, each mixture component in the former (*pitch space*) uniquely represents the pitch of $\mathbf{X}$, while that in the latter (*timbre space*) is interpreted as the instrument identity. As we will see in Section 4.1, however, moderate supervision is essential to learn a timbre space that groups instruments perfectly. For creative applications such as the synthesis and timbre transfer of instrument sounds, the proposed model has numerous merits: 1) the learnt representations are not restricted to be unimodal, which offers a more discriminative timbre space than regular VAEs (Section 4.1 and 4.2); 2) direct and intuitive sampling from pitch and timbre space allows for consistent and controllable synthesis of instrument sounds, attributed to the fact that Gaussian parameters of each interpretable mixture component are readily available after training (Section 4.3); and 3) simple arithmetic manipulations between means of mixture components facilitate many-to-many transfer between instrument timbres (Section 4.4). For the training objective, we closely follow the derivation in [17] and train the model by maximizing the evidence lower bound (ELBO) as follows:

$$\begin{aligned}\mathcal{L}(p,q;\mathbf{X},\mathbf{y}_p) = &\mathbb{E}_{q(\mathbf{z}_p|\mathbf{X})q(\mathbf{z}_t|\mathbf{X})}[\log p(\mathbf{X}|\mathbf{z}_p,\mathbf{z}_t)] \\ &- D_{KL}(q(\mathbf{z}_p|\mathbf{X})||p(\mathbf{z}_p|\mathbf{y}_p)) \\ &- \mathbb{E}_{q(\mathbf{y}_t|\mathbf{X})}[D_{KL}(q(\mathbf{z}_t|\mathbf{X})|p(\mathbf{z}_t|\mathbf{y}_t)] \\ &- D_{KL}(q(\mathbf{y}_t|\mathbf{X})||p(\mathbf{y}_t)),\end{aligned} \tag{2}$$

where $p(\mathbf{X}|\mathbf{z}_p,\mathbf{z}_t)$, $q(\mathbf{z}_p|\mathbf{X})$, and $q(\mathbf{z}_t|\mathbf{X})$ are parameterized with neural networks, referred to as the *decoder*, *pitch encoder* [2] , and *timbre encoder*, respectively. Instead of using another neural network, we approximate $q(\mathbf{y}_t|\mathbf{X})$ by $\mathbb{E}_{q(\mathbf{z}_t|\mathbf{X})}[p(\mathbf{y}_t|\mathbf{z}_t)]$. Readers interested in detailed derivation are referred to Appendix A in [17].

2.3 Architecture

Our model is composed of a shared decoder and separate encoders for pitch and timbre, as illustrated in Figure 1. Specifically, we reshape the T-by-F spectrogram to have number of channels $C = F$, each of which is a T-by-1 vector, where T and F refer to time and frequency. Each encoder contains two one-dimensional convolutional layers, each with 512 filters of shape 3×1, and a fully connected layer with 512 units. A Gaussian parametric layer follows and outputs two $L-$dimensional vectors which represent mean and log variance. $\mathbf{z}_p$ and $\mathbf{z}_t$ are sampled from the Gaussian layer with the reparameterization trick [25], which enables stochastic gradient descent, and are then concatenated for the decoder to reconstruct the input. The architecture of the decoder is symmetric to the encoder. Batch normalization followed by the activation function `relu` are used for every layer except for the Gaussian and the output layer. We use the activation function `tanh` for the output layer as we normalize the data within $[-1, 1]$.

[2] A common alternative is conditioning the model with categorical pitch labels such that one does not have to train a pitch encoder [3, 7]. It, however, requires the pitch of the inputs to be known a priori to performing tasks such as timbre transfer [3], and also prohibits the model from extracting pitch features for downstream tasks. By training this extra encoder, we also demonstrate how one can extend the model to possibly learn multiple interpretable latent variables.

3. EXPERIMENTAL SETUP

In this section, we describe the experimental setup, including details of the dataset, input representations, and model configurations.

3.1 Dataset

Inspired by Esling *et al.* [8], we use a subset of Studio-On-Line (SOL) [1], a database of instrument note recordings. [3] The dataset contains 12 instruments, i.e, piano (`Pno`, 246), violin (`Vn`, 138), cello (`Vc`, 147), English horn (`Ehn`, 128), French horn (`Fhn`, 214), tenor trombone (`Trtb`, 63), trumpet (`Trop`, 194), saxophone (`Sax`, 99), bassoon (`Bn`, 251), clarinet (`Clr`, 180), flute (`Fl`, 118) and oboe (`Ob`, 107). There are 1,885 samples in total. All recordings are resampled to 22,050Hz, and only the first 500ms segment ($T = 43$) of each recording is considered. We extract Mel-spectrograms with 256 filterbanks ($F = 256$), derived from the power magnitude spectrum of the short-time Fourier transform (STFT). To compute STFT, we use a Hann window with window size of 92ms and hop size of 11ms. As a result, the input representation is a 43-by-256 Mel-spectrogram. The dataset is split into a training (90%) and validation set (10%), each containing the same distribution of instruments. The magnitude of the Mel-spectrogram is scaled logarithmically, and the minimum and maximum values in the training set are used for normalizing the magnitude within $[-1, 1]$ in a corpus-wide fashion to preserve differences in dynamics.

3.2 Hyperparameters

In order to train both the GMMs in pitch and timbre space, we initialize the means of mixture components using Xavier initialization [11]. We set constant standard deviations, rather than trainable ones, for pitch and timbre space. For pitch space, $\boldsymbol{\sigma}_{\mathbf{y}_p} = \mathbf{e}^{-\mathbf{2}}$ for all mixture components, which is relatively small, as each mixture component represents a pitch, and we do not expect a large variance over recordings that play the same pitch. For timbre space, we let $\boldsymbol{\sigma}_{\mathbf{y}_t} = \mathbf{e}^{\mathbf{0}}$ for all mixture components, which captures the timbre variation of each mixture component, i.e., instrument identity. The dimensionality of the latent space is $L = 16$, and the numbers of mixture components are $M = 82$ and $K = 12$, equivalent to the numbers of classes of pitch and instrument, respectively. For all experiments, a batch size of 128 is used, model parameters are initialized with Xavier initialization and are trained using the Adam optimizer [23] with a learning rate of 10^{-4}.

In addition to the proposed model (M_{GMVAE}), we consider a baseline (M_{VAE}) that substitutes the timbre space with an isotropic Gaussian as in regular VAEs. Training such a model amounts to optimizing Eqn (2) with the last two terms replaced with $D_{KL}(q(\mathbf{z}_t|\mathbf{X})||p(\mathbf{z}_t))$, where $p(\mathbf{z}_t) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. The experimental results in Section 4.1 and Section 4.2 show that $\mathrm{M_{GMVAE}}$ learns a more discriminative and disentangled timbre space than $\mathrm{M_{VAE}}$.

[3] Access to the dataset was requested from [8].

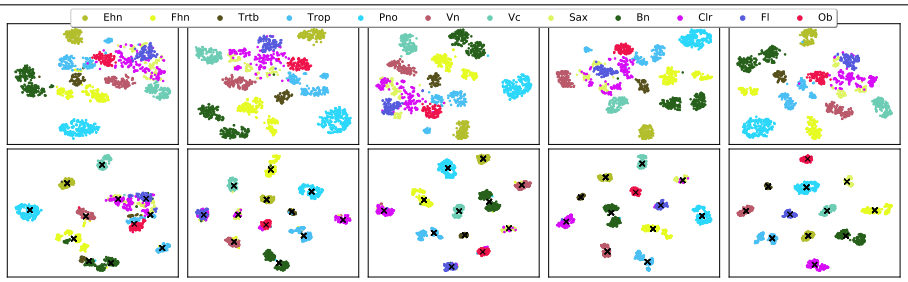


Figure 2. Timbre space visualization of $\mathrm{M}_{\mathrm{VAEs}}$ (top) and $\mathrm{M}_{\mathrm{GMVAEs}}$ (bottom). From left to right: models trained with 0, 25, 50, 75, or 100% of instrument labels, respectively.

	Instrument Classification					Pitch Classification				
N (%)	CNN	$\mathrm{M}_{\mathrm{VAE}}$		$\mathrm{M}_{\mathrm{GMVAE}}$		CNN	$\mathrm{M}_{\mathrm{VAE}}$		$\mathrm{M}_{\mathrm{GMVAE}}$	
		$\mathbf{z}_t$	$\mathbf{z}_p$	$\mathbf{z}_t$	$\mathbf{z}_p$		$\mathbf{z}_t$	$\mathbf{z}_p$	$\mathbf{z}_t$	$\mathbf{z}_p$
0	-	0.960	0.163	0.937	0.175	-	0.112	0.966	0.146	0.960
25	0.920	0.960	0.192	0.971	0.180	-	0.169	0.966	0.084	0.977
50	0.983	0.971	0.169	0.988	0.186	-	0.158	0.977	0.079	0.977
75	1.000	0.971	0.169	1.000	0.163	-	0.079	0.971	0.045	0.977
100	1.000	0.937	0.158	1.000	0.197	0.983	0.039	0.983	0.028	0.966

Table 1. The F-scores of instrument and pitch prediction by linear classifiers and CNNs. N (%) refers to the percentage of instrument labels used to train the models. Columns $\mathbf{z}_t$ and $\mathbf{z}_p$, respectively, refer to the F-scores obtained using the learned timbre and pitch code to train the down-stream linear classifier.

3.3 Semi-Supervised Learning

We exploit a moderate number of instrument labels to learn a timbre space in which the clusters clearly represent instrument identity. Similar to Kingma *et al.* [24], in the semi-supervised training for $\mathrm{M}_{\mathrm{GMVAE}}$, we *guide* the inference of instrument labels $q(\mathbf{y}_t|\mathbf{X})$ by leveraging limited amounts of supervision. This is done by adding an additional loss term which measures the cross entropy between the inferred and true instrument labels. Because we do not infer $\mathbf{y}_t$ in $\mathrm{M}_{\mathrm{VAE}}$, we use $\mathbf{z}_t$ to train an auxiliary classifier to predict $\mathbf{y}_t$. It has two 128-unit fully-connected layers, and is jointly optimized with $\mathrm{M}_{\mathrm{VAE}}$. We consider varying numbers of instrument labels N = 0 (unsupervised), 25, 50, 75, and 100% (fully supervised) of the total number. We randomly sample and let the label distribution match the distribution of instruments.

4. EXPERIMENTS AND RESULTS

The experiments and the results are presented in this section. We first visualize the timbre space, and quantitatively evaluate the disentangled representations. We then demonstrate the applications of controllable synthesis and many-to-many timbre transfer. Finally, we identify the particular latent dimension that is sensitive to the distribution of the spectral centroid, which allows for finer timbre controls.

4.1 Visualization

Figure 2 visualizes the timbre space using t-distributed stochastic neighbor embedding (t-SNE) [34], a technique that projects vectors from high- to low-dimensional space. We first observe that $\mathrm{M}_{\mathrm{GMVAE}}$ learns a Gaussian-mixture distributed timbre space, with means of mixture components marked as crosses in the figure. Second, attributed to the pitch encoder which addresses pitch variations, both $\mathrm{M}_{\mathrm{VAE}}$ and $\mathrm{M}_{\mathrm{GMVAE}}$ are able to form clusters of instrument identity even without being trained with instrument labels (the leftmost column). We observe that the wind family (e.g., saxophone, clarinet and flute) forms an ambiguous cluster. Such an ambiguity remains in the $\mathrm{M}_{\mathrm{VAE}}$ even with increased N, while it is less present in the $\mathrm{M}_{\mathrm{GMVAE}}$ latent space, due to the multi-modal prior distribution. As we will confirm in Section 4.2, $\mathrm{M}_{\mathrm{GMVAE}}$ outperforms $\mathrm{M}_{\mathrm{VAE}}$ in learning a more discriminative and disentangled timbre space. Note that in $\mathrm{M}_{\mathrm{GMVAE}}$, $p(\mathbf{y}_t)$ is assumed to be uniformly distributed over 12 classes of instruments, i.e., mixture components are equally weighted. As a result, instruments with larger within-class variances (e.g., bassoon and trumpet) are assigned to more than one cluster when $N = 0$. In future work we aim to improve the performance of the unsupervised clustering of instruments.

4.2 Pitch and Instrument Disentanglement

A disentangled pitch (timbre) representation should be discriminative for pitch (instrument identity), and at the same time non-informative of instrument identity (pitch). Therefore, we evaluate $\mathbf{z}_p$ and $\mathbf{z}_t$ by means of classification. We train linear classifiers to map $\mathbf{z}_p$ and $\mathbf{z}_t$ to predict both pitch and instrument labels with one fully connected layer. For comparison, we train an end-to-end convolutional neural network (CNN), whose architecture is the same as the encoder and is a strong baseline, to map the original input Mel-spectrograms to either pitch or instrument labels.

Table 1 shows the results. The CNN achieves high F-scores on both instrument and pitch classification; note that N is the supervisory percentage of the total number of *instrument* labels, and we always use all pitch labels to train the models, which is reasonable as we model isolated notes

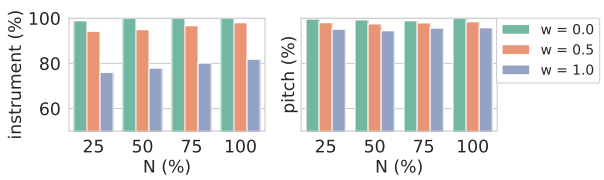


Figure 3. The F-scores for predicting instrument (left) and pitch (right) labels from the synthesized spectrograms.

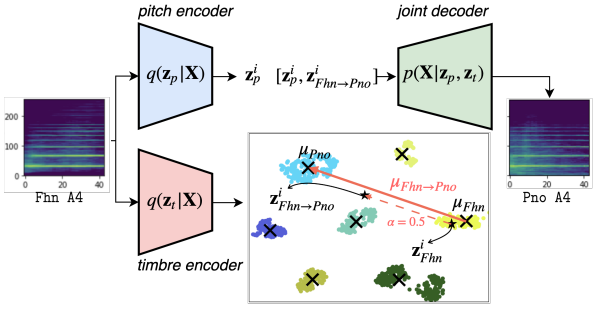


Figure 4. Many-to-many timbre transfer. The ith sample of the `Fhn` is transferred to the `Pno`, with vector arithmetic in the (partially shown) timbre space.

in this work. In instrument classification, using $\mathbf{z}_t$ as the feature representations outperforms $\mathbf{z}_p$ by a large margin, as expected. Specifically, in both models, the $\mathbf{z}_t$ learned with unsupervised learning ($N = 0$) is already discriminative enough to predict instruments with linear classifiers. While the F-score of M_{GMVAE} improves with increased N, that of M_{VAE} does not. Moreover, the linear classifier trained with $\mathbf{z}_t$ outperforms the CNN when $N < 75$. The timbre space of M_{GMVAE} displays the most discriminative power among the models. We attribute the F-scores of instrument classification attained by $\mathbf{z}_p$ to the fact that the piano covers all possible pitches in the dataset, while other instruments account for a smaller pitch range. As a result, $\mathbf{z}_p$ of notes that were only recorded by piano are correctly classified. Future work can be done to decorrelate particular pitches and instruments by data augmentation and adversarial training as in [16]. In pitch classification, $\mathbf{z}_p$ outperforms $\mathbf{z}_t$ as expected, and both models achieve comparable results. More importantly, M_{GMVAE} performs better than M_{VAE} in terms of disentanglement, as $\mathbf{z}_t$ results in lower F-scores when predicting pitch with increased N.

4.3 Controllable Synthesis of Instrument Sounds

As shown in Figure 2, M_{GMVAE} learns a timbre space $p(\mathbf{z}_t)$, whose mixture components are clearly interpreted as instrument identity when trained with moderate supervision. Meanwhile, mixture components in $p(\mathbf{z}_p)$ represent pitch. As Gaussian parameters are readily available after training, we can achieve controllable sound synthesis by sampling $p(\mathbf{z}|\mathbf{y})$. To synthesize the target pitch y_m and instrument y_k, we first sample $\mathbf{z}_p \sim \mathcal{N}(\boldsymbol{\mu}_{y_m}, w \cdot \text{diag}(\boldsymbol{\sigma}_{y_m}))$ and $\mathbf{z}_t \sim \mathcal{N}(\boldsymbol{\mu}_{y_k}, w \cdot \text{diag}(\boldsymbol{\sigma}_{y_k}))$, where the multiplier $w \in \{0, 0.5, 1.0\}$ serves to examine the effect of sampling latent variables that deviate from the modes. The decoder then synthesizes the Mel-spectrogram by consuming $[\mathbf{z}_t, \mathbf{z}_p]$. For evaluation, the CNNs (trained on the original dataset) are used to test whether the synthesized spectrograms are still recognized as belonging to the desired instrument and pitch. High F-scores therefore indicate high controllability of the model in sound synthesis. We use the sound samples in the validation set as the targets to synthesize, and repeat the sampling 30 times for each target.

The F-scores for pitch and instrument classification are reported in Figure 3. We first note that increasing w degrades classification performance. This is expected, as a sample which is synthesized using a latent variable far from its corresponding mean of mixture component deviates more from the intended instrument or pitch distribution. Moreover, the fact that the CNN was trained on the original samples while tested on the synthesized ones also contributes to the inferior performance. Second, increasing N improves instrument classification performance. Finally, the high F-scores across all N's when $w \in \{0, 0.5\}$ indicate accurate and consistent synthesis of instrument sounds with intended pitches and instruments, even with a timbre space trained using a limited number of instrument labels. This implies that M_{GMVAE} efficiently exploits the instrument labels, and learns a discriminative mixture distribution of timbre, which is consistent with the visualization in Figure 2 (bottom row, $N \geq 25$). We do not explore the timbre space resulting from unsupervised learning ($N = 0$) in this experiment, as the instrument identity of each mixture component is not directly available. We can, however, infer the instrument identity of each mixture component by sampling and synthesis, and expect reasonably good performance for controllable synthesis if the clustering of instruments shown in the bottom left of Figure 2 is improved. This will be explored in future work.

4.4 Many-to-Many Transfer of Timbre

In this experiment, we demonstrate many-to-many transfer of instrument timbre. In Mor *et al.*, a domain-specific decoder was trained for each target [28]. To achieve timbre transfer with a single encoder-decoder architecture, Bitton *et al.* proposed to use a conditional layer [31] which takes both instrument and pitch labels as inputs [3]. On the other hand, our model infers $\mathbf{z}_t$ and $\mathbf{z}_p$, and only uses a single joint decoder. As illustrated in Figure 4, timbre transfer is achieved by decoding $[\mathbf{z}_{transfer}, \mathbf{z}_p]$, i.e., transferring timbre while keeping pitch unchanged, where $\mathbf{z}_{transfer} = \mathbf{z}_{source} + \alpha\boldsymbol{\mu}_{source\to target}$, $\boldsymbol{\mu}_{source\to target} = \boldsymbol{\mu}_{target} - \boldsymbol{\mu}_{source}$, and $\alpha \in [0, 1]$. Once again, we rely on the trained CNNs in Table 1 for evaluation. More specifically, we examine the posterior shift in instrument prediction of the CNN, before and after transferring from source to target instruments with $\alpha = \{0, 0.25, 0.5, 0.75, 1.0\}$. For simplicity, the most frequent instruments (i.e., French horn, piano, cello, and bassoon) of the four families are selected as the representatives, and we perform timbre transfer using the samples in the validation set as the source. For example, consider `Fhn` as the source and `Pno` as target, as shown in Figure 4. We modify the timbre code as $\mathbf{z}^i_{Fhn\to Pno} = \mathbf{z}^i_{Fhn} + \alpha\boldsymbol{\mu}_{Fhn\to Pno}$, where $\mathbf{z}^i_{Fhn}$ is the timbre code of the ith `Fhn` sample, and $i = \{1, 2, \ldots, \text{N}_{\text{Fhn}}\}$.

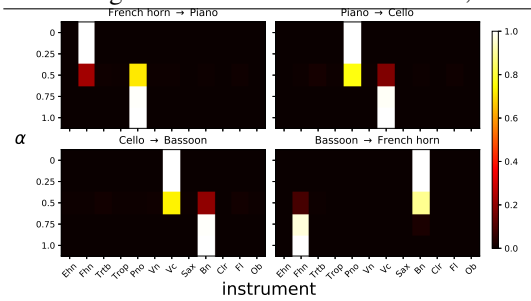


Figure 5. The averaged posterior (color) shift in instrument prediction of the CNN, caused by timbre transfer.

We decode as described earlier and report the averaged posterior (over N_{Fhn}) of instrument prediction of the CNN.

For simplicity, in Figure 5, we report the results of the source-target pairs $Fhn \rightarrow Pno$, $Pno \rightarrow Vc$, $Vc \rightarrow Bn$ and $Bn \rightarrow Fhn$. Each subfigure refers to a source-target pair, and represents the averaged posterior shift of instrument classification of the CNN, with varying α. For all pairs, the biggest posterior shift (hence the prediction change) happens when $\alpha = 0.5$. This also applies to the rest of the possible instrument pairs not shown in the figure. Meanwhile, by using pitch classification, we examine if the pitches are the same before and after timbre transfer, and we use the original pitch labels as ground-truths. We find that, except in the case where the source is piano, all source-target pairs attain a perfect F-score in terms of pitch. This confirms the ability of the model to successfully perform many-to-many timbre transfer. A special case arises when piano is the source. The F-scores before transfer, after transfer to French horn, to cello, and to bassoon, are 0.958, 0.750, 0.791, and 0.791, respectively. As described earlier in Section 4.2, lower F-scores can be attributed to the fact that the range of piano is much larger than that of the target instruments, or the classifier fails to predict the synthesized samples that have unseen combinations of pitch and instrument. The other possible reason is the model falls short of generalization. Nevertheless, this only happens in some cases when the source is piano; as demonstrated in Figure 6, the model is able to transfer `Pno G6` to cello (the first row), which is an example of generalizing to an out-of-range pitch for the target instrument. In the first and third row, the high-frequency components appear with increased α, and the energy distributes over the segment without decay. The model, however, falls short in

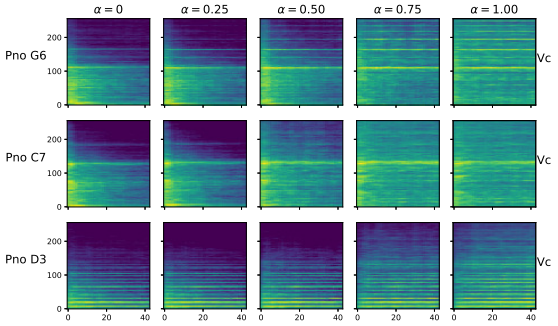


Figure 6. Examples of timbre transfer $Pno \rightarrow Vc$. The top two rows are tones outside of the cello range.

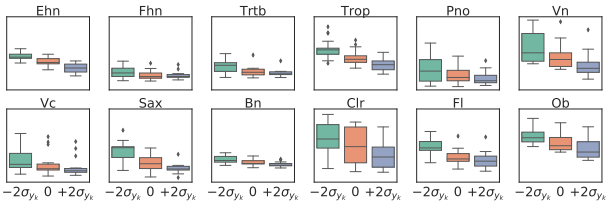


Figure 7. Spectral centroid values in response to z_t^{13}.

generalizing to the higher pitch, i.e., `Pno C7` (the second row), where the energy remains focused at the onset, and high-frequency components are smeared. In the future, we could improve the model generalizability by performing data augmentation and adversarial training as in [16].

4.5 Spectral Centroid Disentanglement

A diagonal-covariance Gaussian prior encourages the model to learn disentangled latent dimensions [14]. This applies to all mixture components in our model. In particular, we identify a latent dimension that correlates with the spectral centroid. we modify the 13th dimension of $\mathbf{z}_t$, z_t^{13}, of each sound sample in the validation set by $\pm 2\sigma_{y_k}$, where $\sigma_{y_k} = e^0$ for all instruments, and then synthesize the spectrograms, for which we then calculate the spectral centroid. Figure 7 shows the distributions of the spectral centroid before and after the modifications. The two-tailed t-test indicates significant differences ($p < 0.05$) between $-2\sigma_{y_k}$ and $+2\sigma_{y_k}$ for all instruments. As demonstrated in Figure 8, we observe that increased z_t^{13} reduces the energy of high-frequency components and results in lower spectral centroid values. In future research, we will further investigate disentangling specific acoustic features for finer control of sound synthesis beyond pitch and instrument.

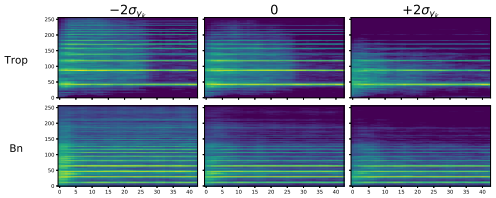


Figure 8. Latent dimension traverse of z_t^{13}.

5. CONCLUSIONS AND FUTURE WORK

We have proposed a framework based on GMVAEs to learn disentangled timbre and pitch representations for musical instrument sounds, which is verified by our experimental setup. We demonstrate its applicability in controllable sound synthesis and many-to-many timbre transfer. In future work, we plan to conduct listening tests for a more comprehensive evaluation of the applications, and further disentangle both low- (e.g., acoustic features) and high-level (e.g., playing techniques) sound attributes, enabling finer control of synthesized timbres. By using supervised and unsupervised learning in a deep generative model, the framework can be easily adapted to learn interpretable mixtures such as singer identity, music style, emotion, etc., which facilitates music representation learning and creative applications.

6. ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their constructive reviews. This work is supported by a Singapore International Graduate Award (SINGA) provided by the Agency for Science, Technology and Research (A*STAR), under reference number SING-2018-01-1270.

7. REFERENCES

[1] G. Ballet, R. Borghesi, P. Hoffmann, and F. Levy. Studio online 3.0: An internet "killer application" for remote acess to ircam sounds and processing tools. *Journee Informatique Musicale*, 1999.

[2] Y. Bengio. Deep learning of representations: Looking forward. In *International Conference on Statistical Language and Speech Processing*, pages 1–37. Springer, 2013.

[3] A. Bitton, P. Esling, and A. Chemla-Romeu-Santos. Modulated variational auto-encoders for many-to-many musical timbre transfer. *arXiv preprint arXiv:1810.00222*, 2018.

[4] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer. Midi-vae: Modeling dynamics and instrumentation of music with applications to style transfer. In *Proc. of the International Society for Music Information Retrieval Conference*, pages 23–27, 2018.

[5] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[6] N. Dilokthanakul, P. A. M. Mediano, M. Garnelo, M. C.-H. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.

[7] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan. Neural audio synthesis of musical notes with wavenet autoencoders. In *Proc. of the International Conference on Machine Learning*, pages 1068–1077, 2017.

[8] P. Esling and A. Bitton. Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces. In *Proc. of the International Society for Music Information Retrieval Conference*, 2018.

[9] P. Esling, A. ChemlaRomeu-Santos, and A. Bitton. Generative timbre spaces with variational audio synthesis. In *Proc. of the International Conference on Digital Audio Effects*, 2018.

[10] Z. Fu, X. Tan, N. Peng, D. Zhao, and R. Yan. Style transfer in text: Exploration and evaluation. In *AAAI*, 2017.

[11] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AAAI*, pages 249–256, 2010.

[12] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 2014.

[14] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, M. Shakir, , and A. Lerchner. Beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

[15] W.-N. Hsu, Y. Zhang, and J. Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in Neural Information Processing Systems*, pages 1878–1889, 2017.

[16] W.-N. Hsu, Y. Zhang, R. J. Weiss, Y.-A. Chung, Y. Wang, Y. Wu, and J. Glass. Disentangling correlated speaker and noise for speech synthesis via data augmentation and adversarial factorization. In *Advances in Neural Information Processing Systems*, 2018.

[17] W.-N. Hsu, Y. Zhang, R. J. Weiss, H. Zen, Y. Wu, Y. Wang, Y. Cao, Y. Jia, Z. Chen, J. Shen, P. Nguyen, and R. Pang. Hierarchical generative modeling for controllable speech synthesis. In *International Conference on Learning Representations*, 2019.

[18] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P Xing. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596, 2017.

[19] Z. Hu, Z. Yang, R. Salakhutdinov, and E. P. Xing. On unifying deep generative models. *arXiv preprint arXiv:1706.00550*, 2017.

[20] S. Huang, Q. Li, C. Anil, X. Bao, S. Oore, and R. B. Grosse. Timbretron: A wavenet (cyclegan (cqt (audio))) pipeline for musical timbre transfer. *arXiv preprint arXiv:1811.09620*, 2018.

[21] Y.-N. Hung, Y.-A. Chen, and Y.-H. Yang. Learning disentangled representations for timber and pitch in music audio. *arXiv preprint arXiv:1811.03271*, 2018.

[22] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In *International Joint Conference on Artificial Intelligence*, 2017.

[23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

[25] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2014.

[26] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2530–2538, 2015.

[27] A. Kumar, P. Sattigeri, and A. Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. *International Conference on Learning Representations*, 2018.

[28] N. Mor, L. Wolf, A. Polyak, and Y. Taigman. A universal music translation network. *arXiv preprint arXiv:1805.07848*, 2018.

[29] M. Muller, D. P. W. Ellis, A. Klapuri, and G. Richard. Signal processing for music analysis. *IEEE Journal of Selected Topics in Signal Processing*, 5(6):1088–1110, 2011.

[30] M. Muller and S. Ewert. Towards timbre-invariant audio features for harmony-based music. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):649–662, 2010.

[31] E. Perez, F. Strub, H. D. Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

[32] K. Ridgeway. A survey of inductive biases for factorial representation-learning. *arXiv preprint arXiv:1612.05299*, 2016.

[33] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[34] L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[35] Y. Wang, D. Stanton, Y. Zhang, RJ Skerry-Ryan, E. Battenberg, J. Shor, Y. Xiao, F. Ren, Y. Jia, and R. A. Saurous. Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis. *arXiv preprint arXiv:1803.09017*, 2018.

[36] H. Zhou, M. Huang, T. Zhang, X. Zhi, and B. Liu. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *AAAI*, 2017.

THE ISMIR EXPLORER – A VISUAL INTERFACE FOR EXPLORING 20 YEARS OF ISMIR PUBLICATIONS

Thomas Low[1] **Christian Hentschel**[2] **Sayantan Polley**[1] **Anustup Das**[1]
Harald Sack[3] **Andreas Nürnberger**[1] **Sebastian Stober**[1]
[1] Faculty of Computer Science, Otto von Guericke University Magdeburg, Germany
[2] Hasso Plattner Institute for IT Systems Engineering, University of Potsdam, Germany
[3] FIZ Karlsruhe – Leibniz Institute for Information Infrastructure, Karlsruhe, Germany
stober@ovgu.de

ABSTRACT

Ever since the first International Symposium on Music Information Retrieval in 2000, the proceedings have been made publicly available to interested researchers. After 20 years of annual conferences and workshops, this number has grown to an impressive amount of almost 2,000 papers. When restricted to linear search and retrieval in a document collection of this size, it becomes inherently hard to identify topics, related work and trends in scientific research. Therefore, this paper presents and evaluates a map-based user interface for exploring 20 years of ISMIR publications. The interface visualizes k-nearest neighbor subsets of semantically similar papers. Users may jump from one neighborhood to the next by selecting another paper from the current subset. Through animated transitions between local k-nn maps, the interface creates the impression of panning a large global map. Evaluation results of a small user study suggest that users are able to discover interesting links between papers. Due to its generic approach, the interface is easily applicable to other document collections as well. The search interface and its source code are made publicly available. [1]

1. INTRODUCTION

Music Information Retrieval (MIR) has been a steadily growing field of research as documented by the official statistics from the International Society of Music Information Retrieval (ISMIR). [2] For the 20th anniversary of the annual ISMIR conference, the number of published papers at this event is expected to exceed 2,000. This work aims to make this great resource more accessible for researchers who would like to familiarize themselves with the field of MIR.

The classical search interface based on keyword search and result lists is well suited for users who know what they are looking for. However, in other scenarios, this might not be the ideal search approach and the amount and diversity of papers and topics can be overwhelming. Exploratory information retrieval systems support users with vague or evolving information needs by providing special user interfaces and interaction models. The ISMIR Cloud Browser [3] [4], last updated in 2013, allows browsing through links of related terms. Our new web-based search interface, the ISMIR Paper Explorer, takes a different – more visual – approach to exploration using similarity-based two-dimensional maps. It can be accessed at https://ismir-explorer.ai.ovgu.de.

Section 2 covers related and prior research that led to or inspired our approach. Details on the underlying dataset and preprocessing are provided in Section 3. We then describe the user interface in Section 4 and present the results from a usability study of our first prototype in Section 5. Furthermore, we also provide insights gained from analyzing the cumulative ISMIR proceedings to detect trends like shifts in topics and methods, and demonstrate whether and how these findings show in the visual search interface (Section 6). Finally, Section 7 offers conclusions.

2. RELATED WORK

In contrast to keyword-based search or recommendation, the goal of exploration is not only to present relevant or related items, but to allow the user to learn about the information space and its properties. This is often desirable when dealing with large document collections and when an overview over a collection is preferred over a pinpoint search.

A common problem when designing interfaces for exploratory search is the typically very high-dimensional feature space used to describe the contents of the respective documents. Considering the task of exploring text documents such as scientific papers, the document contents is commonly expressed as a (weighted) vector of term frequencies (tf-idf). Depending on the selected vocabulary, these often gather up to thousands of dimensions, which are hard to be displayed on a common computer display. Thus, map-based visualizations of inter-document similarities requires projection of these features into a two-dimensional display space. The neighborhood relations in high-dimensional space, however, should be

[1] https://doi.org/10.6084/m9.figshare.8342594
[2] https://www.ismir.net/stats.html
[3] http://dc.ofai.at/browser

preserved throughout the projection so that similarity can be easily perceived by the user.

In order to reduce the dimensionality of the feature spaces, different dimension reduction techniques have been proposed (a comprehensive survey is given in [2]). Popular approaches for dimensionality reduction are Self-Organizing Maps (SOMs) [6], Principle Component Analysis (PCA) [5] and Multidimensional Scaling (MDS) techniques [8].

Inevitably, any reduction of dimensionality will also reduce the information conveyed. None of the aforementioned methods is capable of fully preserving the structure of the collection (unless the collection already has a 2-D structure). A projection will cause errors in a sense that documents, whose feature vectors are very close in high-dimensional space might be projected at large distances from each other and documents that are very dissimilar are placed next to each other respectively. Most dimension reduction techniques try to minimize this error (e.g. stress optimization in MDS), however, a complete resolution is impossible due to the nature of the error. In [13] the authors reviewed and compared different dimensionality reduction algorithms for the visualization of music collections. Based on a user study, MDS was favored as best layout algorithm when the collection undergoes changes due to newly added items.

Although applied in the context of image retrieval, Rubner et al. [11] were among the first to propose multidimensional scaling for iterated search in large document collections. The authors suggest to compute a *local* MDS on the top ten best matching images for a given query. Follow-up queries are emitted by using a representative image of a specific area in the MDS as the next query. A *global* MDS is suggested by the authors to provide a broad overview of the entire collection. While this global approach gave good results for rather small sets of images (the authors tested with a 2-D map of 500 images) it can be easily seen that very large collections (such as the collection of all ISMIR papers) cannot be reasonably presented at a time. Moreover, the number of projection errors will increase with increasing number of documents to be displayed.

A different approach to local MDS is given in [14]. The authors discuss the problems of dimension reduction and introduce the measures of *trustworthiness* (closest neighbors of a document in display space are also neighbors in original space) and *continuity* (all proximities in original space are visualized in display space). The proposed method to local multidimensional scaling is different from the one suggested by [11] in that it allows to parameterize and adjust the compromise between trustworthiness and continuity using an adapted cost function. The MDS is computed on varying neighborhood sizes and trustworthiness as well as continuity are reported to decrease with increasing neighborhood size. In a subsequent work the authors present an unsupervised and a supervised approach to optimize both measures [15].

As discussed, naturally, none of these approaches is capable of actually solving the problems of dimension reduction. In this paper, we therefore follow the idea of reducing the projection error itself rather than trying to attenuate the errors that occur from dimension reduction. Similar to [11] we compute a local MDS on a small subset of the entire collection. Our approach followed in this paper takes these ideas one step further by computing a significantly larger virtual map using MDS of which only a small subset is actually displayed. By selecting one of the displayed documents as follow-up query document, the user can navigate through the collection. Thus, we increase the overlap of any two consecutive maps and thereby increase continuity throughout the navigation process. Furthermore, as suggested in [13] we use Procrustes analysis [3] to better align newly generated maps with their respective predecessors, which leads to a reduced confusion of users when navigating with aligned maps.

3. DATASET PREPARATION

We collected all accessible papers from the years 2000 to 2018 [4] by scraping the respective conference websites, the DBLP index [5] and the recently established Zenodo repository. [6] From a total of 1822 URLs, 28 were broken. We were able to retrieve 16 of these papers through a manual web search. 12 papers are still missing. Meta-data such as paper title and authors, publication year and abstract was retrieved from DBLP. In order to obtain paper citation counts, Google Scholar [7] was queried.

All papers were pre-processed in order to extract the plain text from the originating PDFs (where possible). For several older papers, we had to apply OCR techniques because they only contained the text as image or used obscure font encodings – possibly to hamper copying text from the paper. We experimented with several text extraction tools and eventually settled for *pdftotext* as provided by the Xpdf PDF viewer and toolkit [8] which consistently provided good extraction results. 3 PDFs turned out to be corrupted, leaving 1807 paper to be indexed.

As the meta-data is missing abstracts for most of the papers, we automatically extracted abstracts from the texts using simple heuristics. Furthermore, a preview image of the first page of each paper was extracted using *pdftopng* provided by Xpdf. For imports of future proceedings, scripts have been created to automate the process.

4. PROPOSED USER-INTERFACE FOR VISUAL BERRYPICKING

In classic information retrieval, *berrypicking* refers to the process of finding relevant information through a series of modified search queries [1]. With each query the user inspects a new set of search results and gains a better understanding of the underlying document space as well as his or her information need. Due to this incremental learning process the user is able to formulate new search terms and continue searching.

The proposed user interface adapts this pattern from a list-based presentation of search results to a map-based visualization. Instead of sorting results by relevance with respect to search terms, they are arranged in a similarity preserving two-dimensional projection. Additionally, the burden of having to formulate search terms is substituted by a query-by-example

[4] We aim to update the explorer until the 2019'th ISMIR conference in order to provide access to all 20 years of ISMIR publications.

[5] https://dblp.uni-trier.de/db/conf/ismir

[6] https://zenodo.org/communities/ismir

[7] https://scholar.google.com/

[8] https://www.xpdfreader.com

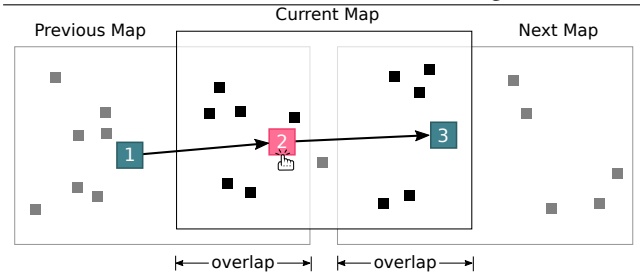


Figure 1. Similarity-based projection of nearest neighbors (squares) using three seed items (colored squares 1,2,3). Common neighbors (black squares) overlap between consecutive maps and are used for alignment when navigating from one item to the next.

search strategy. Such a user interface is hypothesized to be more in line with the goal of learning about the document space.

4.1 General Approach

Instead of trying to visualize the whole proceedings collection in a single similarity-based two-dimensional map, we visualize only the set of k-nearest neighbors for a given seed document in a small map [9]. This circumvents the challenge of the degrading projection quality (c.f. Section 2) and the increasing computational costs for on-the-fly map generation. By selecting a new seed document amongst the k-nearest neighbors, the user is able to hop from one neighborhood map to another. Consecutive maps are aligned as illustrated in Figure 1 to create a consistent transition that is, ideally, perceived as panning a large (global) map. Users are able to transfer knowledge about the content and the relevance of documents accumulated during the exploration process from one visualization to the next. This allows the user to navigate step-by-step through the whole collection.

4.2 Methods

We apply Multi-dimensional Scaling (MDS) [8] for dimensionality reduction. By limiting the number of documents used to compute the projection, we reduce the impact of projection errors and thus visualizations become more reasonable. Transitions between consecutive maps are animated. In order to make these transitions as consistent as possible, we align consecutive maps on their common neighbors. We use Procrustes analysis [3] to reduce the sum of the squared distances between the two sets of images that remain visible during the transition.

As preparation within the Python-based backend, all extracted texts are indexed using the Whoosh library. [9] We also pre-compute pairwise document similarities (stored in a distance matrix) using the tf-idf vector space model [12] considering the 1,000 most frequent terms after applying a Porter Stemmer [10]. This pre-computation allows to quickly retrieve k-nearest neighbors and the corresponding distance matrix for the MDS projection.

The front-end is implemented in HTML5 based on the jQuery [10] javascript library. Cookies are used to store pinned and highlighted papers as well as the user interface settings (cf. Section 4.3). They also allow to track individual sessions for evaluation purposes.

4.3 User Interface

The user interface as shown in Figure 2 consists of three major components: a search bar on top that provides a keyword search interface and access to the settings for the visualization, the map visualization on the left and a details pane that displays the meta-data and the abstract or a preview image for the paper that is currently in focus.

A search can be started by clicking on one of the initially displayed papers or by typing query terms into the search bar. This will generate a map of the k most relevant ISMIR papers for the query. The matching query terms used to populate the initial view are highlighted in the presented information on the map as well as in the details pane. The parameter k is initially set to 30 but can be altered by the user. By clicking on any paper, it will gain focus (indicated by a blue border) and the map will change to display the k most similar ISMIR papers using this paper as new query. A paper can also gain temporary focus (without map update) by hovering the mouse pointer over it. As alternative to the default map visualization, the papers can be arranged in a grid layout that preserves similarities and, in addition, avoids overlaps, see Figure 6 (right).

On the map, each paper is visualized as a rectangle with the title and the last names of the first and last author inside. The fill color saturation visualizes the citation count whereas the border color highlights the current paper in focus (in strong blue) as well as the recent history of focused papers (in light blue). On hovering over a paper, functions for pinning and highlighting become available which are visualized by a landmark pin and a star respectively as explained in Figure 3. Both become permanently displayed in yellow if selected. Pinning keeps a paper on the map, even if it no longer belongs to the k-nearest neighbors of the paper in focus. This can be especially helpful when identifying related work and using the map to visualize similarity relations between the found papers. Highlighted papers are easy to spot because of the star and the additional yellow border color. This function can, for instance, be used to help orientation on the map. Furthermore, papers can be bookmarked using the bookmarking function built into the web browser. The browser's back and forward buttons allow to step through the search history.

The details pane on the right displays the meta-data of the paper in focus comprising the title, the full list of authors, the publication year, and the citation count at the top. Below are two tabs containing the abstract and a preview of the first page of the PDF respectively. Clicking on the title opens the PDF. There is further a link to the DBLP BibTeX record. The content of the details pane can be temporarily changed by hovering over a paper on the map. This allows to inspect paper details without having to focus the paper which would imply a map update.

A first user interface prototype with the functions described above was evaluated in a usability study as described in Section 5. Afterwards, several additional functions were introduced to support a historical analysis of the cumulative ISMIR proceedings for the 20th anniversary. To visualize the year of publication for papers displayed on the map, the fill color hue

[9] https://bitbucket.org/mchaput/whoosh/
[10] https://jquery.com

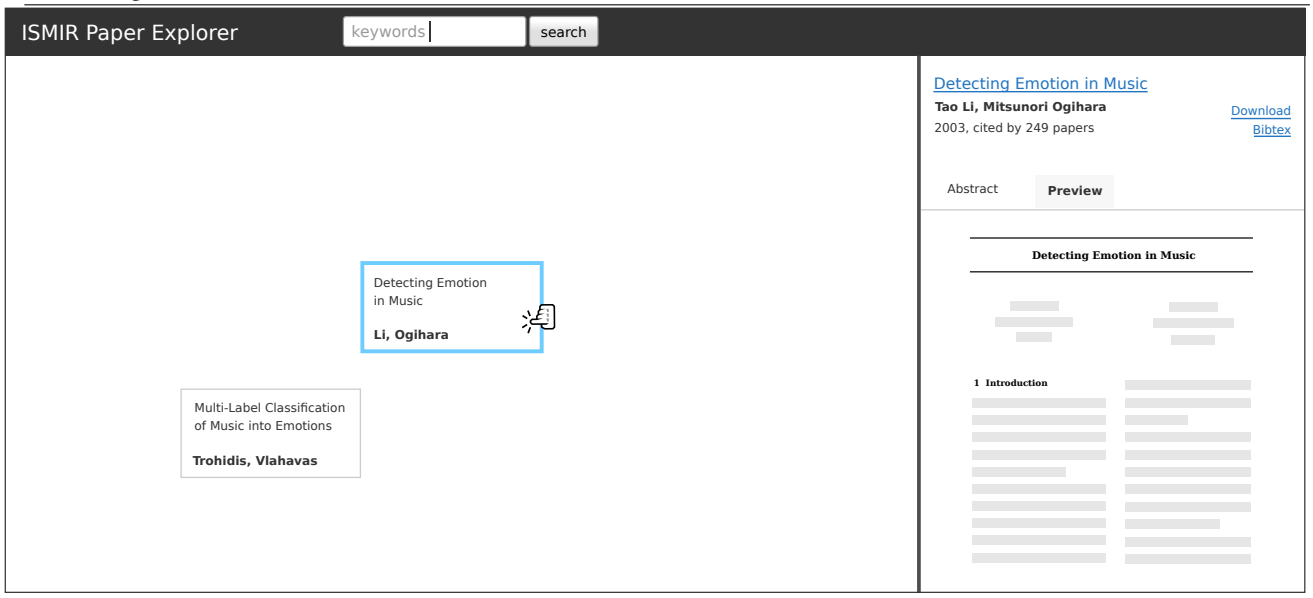


Figure 2. User interface schematic.

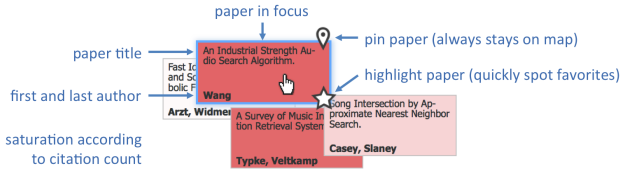


Figure 3. Explanation of the visual elements used in the map (accessible through a help button in the search bar at the top of the user interface).

was adapted. To this end, a continuous color map was divided into 20 steps representing the years. This is compatible with the original coloring to indicate the citation count as this only relies on the color saturation. Furthermore, instead of starting with a random paper in focus and its k-nearest neighbors, we now start with the k most cited papers which naturally cover a wide range of topics addressed in MIR. This makes for a nice starting point to explore the different sub-fields of MIR.

5. EVALUATION

Based on the prototypical implementation of the exploration interface described above, we conducted a small anonymous user study with the aim to evaluate user satisfaction with the presented design. Study participants were asked to compare the explorer interface with available traditional linear paper browsing interfaces for the ISMIR proceedings. Specifically, we asked about experiences using DBLP [5], the Cumulative ISMIR Proceedings browser [11] as well as the ISMIR Cloud Browser [3] [4]. Furthermore, participants were also free to mention tools apart from these. We were interested to see whether the prototype helped to identify related literature and discover additional related topics, which were the fundamental goals when designing the interface. We also wanted to know whether the prototype helped to efficiently find what the user was looking for.

Furthermore, we asked the participants about their overall satisfaction with regard to the proposed interface. Here, we were mainly interested to see whether a fundamentally different approach to paper exploration is still considered as *easy to use*. A key aspect of the interface is the exploitation of the 2-D map to spatially arrange similar papers. We therefore wanted to know whether the spatial grouping helped to *infer relevance* for unknown papers and whether the user were able to effectively *refine the topic focus* during navigation. Finally, as mentioned above, applying the MDS to a restricted neighborhood may result in some confusion when changing the views. We were interested to understand whether the arrangement of papers during navigation was still perceived as *plausible*.

Finally, the users were able to enter additional comments and proposals for improvements in free text.

5.1 Evaluation Results

We received a total of 20 valid responses. Figure 4 presents the aggregated results of all responses. Out of the 20 participants, 13 users were experienced with some of the alternative search tools for finding relevant ISMIR papers. 2 participants mentioned Google or Google Scholar as alternative tools for browsing, which are both linear search and retrieval tools. For users, who were not experienced in any other alternative method for ISMIR paper exploration, we rated their answers regarding the comparison with the proposed protoype as neutral.

In general, the prototype was rated very positively by the study participants. We were pleased to see that a majority of the participants (65%) agreed that the proposed approach helped them to identify literature related to their topic of interest. An overwhelming majority of 90% of the participants furthermore agreed or strongly agreed to the claim that the interface was easy to use, which surprised us due to the fundamentally different browsing experience. Furthermore, according to the results of the study, the suggested spatial alignment helped 70% of the participants to infer the relevance of unknown papers based on the grouping of the neighborhood. As could have been expected, a small number of participants (20%) were not satisfied with the arrangement of the papers

[11] http://www.ismir.net/proceedings/index.php

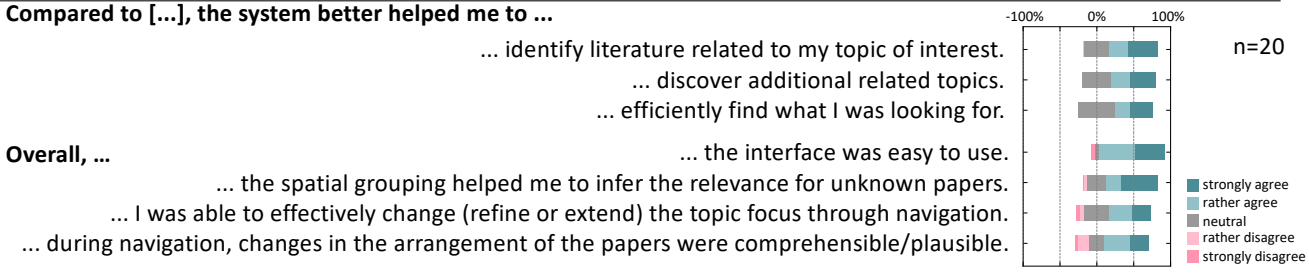


Figure 4. Evaluation results presented in diverging stacked bar charts showing the percentage of participants rating the proposed interface according to the respective statements.

during navigation as it was not considered comprehensible.

A number of participants asked us to publicly release the source code for future development. A few other comments were made with suggestions to further improvements of the usability. One participant mentioned that it is currently not possible to retrieve a list of bookmarked papers, which is a valuable feature for a future release. A few users mentioned that a persistently visible tutorial or legend would help a novice to better use the interface. Finally, an issue was raised about the overlapping surrogates in the non-grid view, which make it sometimes hard to reach them with the mouse. While we encourage the use of the grid view in these cases, we are aware that the grid layout also breaks the spatial alignment of the papers.

6. EXPLORATORY SEARCH

To identify interesting trends in the dataset, we first followed a conventional approach. We binned the publications into 5-year spans and performed a basic bi-gram analysis on the paper abstracts. Table 1 shows the most frequent bi-grams for each bin. Next, we applied a simple heuristic proposed in [7] to detect the most popular methods by assuming that the name of the method appears after phrases like "based on" or "using."

Having identified the most prominent methods, we counted all papers per year mentioning them. For this, we also considered variations like acronyms or slightly different ways of spelling. A plot for the most popular methods is shown in Figure 5 (top). The rise of deep learning since about 2012 cannot be overlooked here. The middle plot in Figure 5 shows the development for select MIR topics like query-by-humming (QBH) which was particularly popular in the early 2000s, optical music recognition (OMR) which has been of constant interest, the development of graphical user interfaces (GUIs) which peaked in 2004 and 2008 mainly driven by approaches based on self-organizing maps (SOMs), and music recommendation which gained massive popularity around the turn of the decade and has leveled off slightly since then. Finally, Figure 5 (bottom) shows the popularity of selected music features over time. Here, the increasing use of chroma features and the slow development of MIR research addressing emotions stand out. At the same time, melody had a very high initial popularity (likely due to a focus on symbolic music representations and QBH in early years) that it lost to some extend over time but may gain back thanks to recently sparked interest in generative models for music. This trend could also explain the emergence of "symbolic music" as a top-5 bi-gram in most recent paper abstracts.

Next, we wanted to see, whether we can make similar

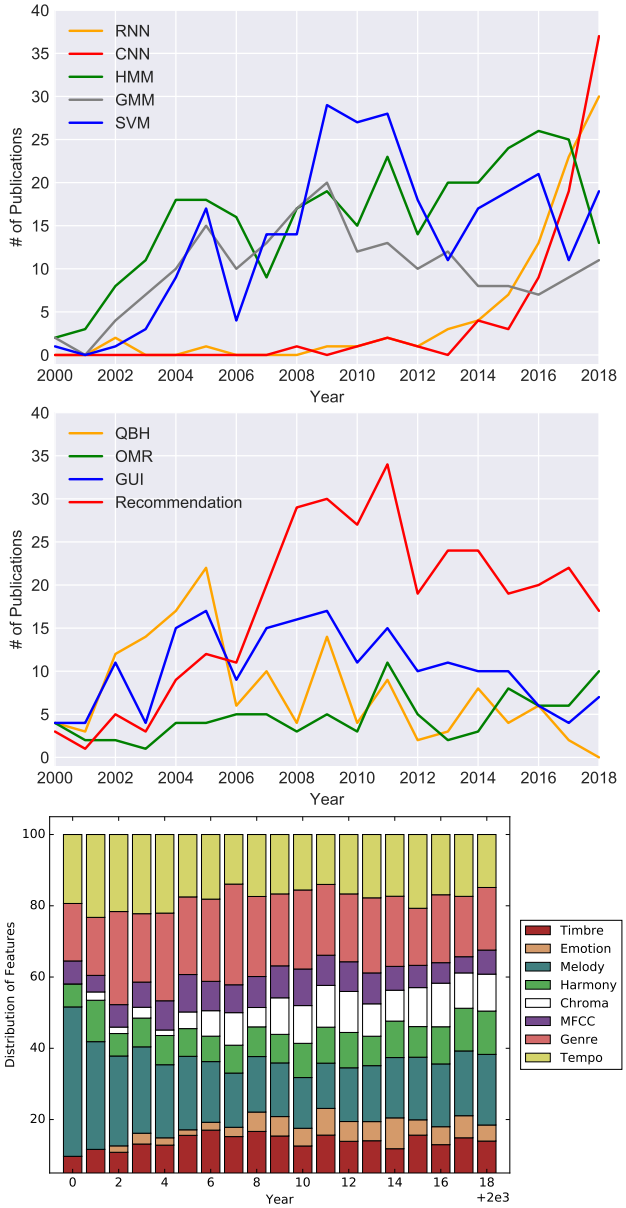


Figure 5. Examples for development over time. Top: popular machine learning techniques. Middle: selected MIR topics. Bottom: selected aspects/features of music.

observations with our exploratory search interface. Here, the same data is used in the back but only a small part of it is shown at a time to not overwhelm the user. Figure 6 (left) visualizes the top-30 papers for the query "query by humming." Based on the dominating orange and green hues, most of

Table 1. Top-5 bi-grams for each 5-year bin. Numbers correspond to the frequency of each bi-gram within paper abstracts.

2000–2004	2005–2009	2010–2014	2015–2018
content based 30	polyphonic music 30	matrix factorization 27	neural network 52
polyphonic music 26	Gaussian mixture 15	music structure 18	recurrent neural 33
similarity measures 13	music recommendation 10	interactively browse 15	convolutional neural 32
pattern matching 10	collaborative filtering 9	emotion recognition 15	time frequency 22
acoustic features 6	rhythmic information 7	automatic transcription 11	symbolic music 16

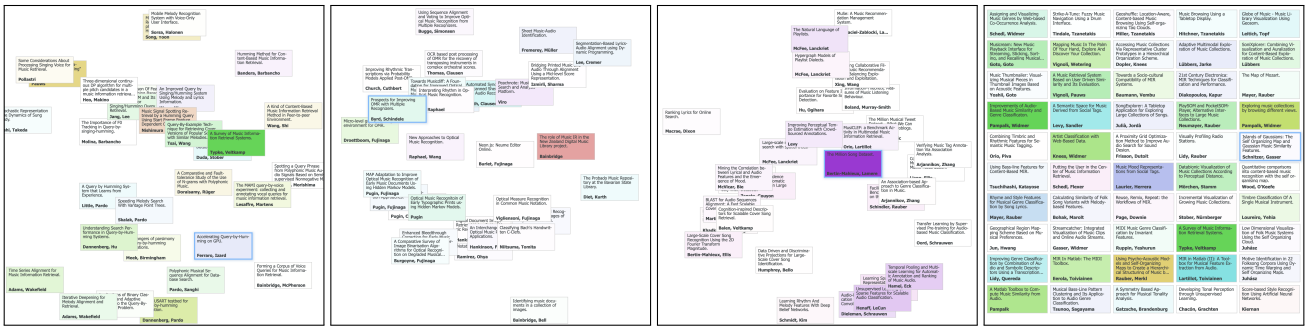


Figure 6. Map visualizations for selected initial search queries – from left to right: "query by humming", "OMR", "million song dataset", "SOM". All except the rightmost sub-figure show the top-30 results. The rightmost sub-figure shows the top-50 results for "SOM" in grid mode. The color saturation is proportional to the citation counts and the hue corresponds to the year starting with red in 2000 and continuing in equal steps through the spectrum.

the papers come from the early-to-mid 2000s with the by far most-cited paper being a survey of MIR systems from 2005. This matches the respective plot in Figure 5 (middle). The advantage in the view is that clusters can be identified like the aspects of singing in the top left, alignment in the lower left or system performance at the bottom.

For the topic OMR (second map from the left in Figure 6), there are some highly-cited papers around 2010 but there is also plenty of coverage in recent years which is in line with Figure 5 (middle). Recognizable clusters comprise, for instance, alignment (top right), applications in libraries (middle right) and old data (bottom).

The third map from the left in Figure 6 shows papers mentioning the Million Song Dataset. As can be seen easily, these are all fairly recent papers with the original publication standing out clearly due to its high citation count. This visualization nicely illustrates the different ways this dataset has been used in research since its publication. There are, for instance, clusters on cover songs (lower left), training deep neural networks (lower right), and on playlists and recommendation (top).

Finally, the rightmost map in Figure 6 shows the top-50 papers for the query "SOM." They are displayed in grid mode to avoid clutter which starts to become an issue at this setting of k for typical screen resolutions. Nevertheless, the 50 papers can still be displayed comfortably in grid mode. This visualization covers almost all relevant papers for this topic. Here, the most-cited papers can be attributed to the period between 2005 and 2010 which was a time of high popularity for this technique being used in a large variety of graphical user interfaces. This also correlates with the two peaks around this time for GUI in Figure 5 (middle).

These are just a few examples about what can be discovered by exploring the cumulative ISMIR proceedings using our proposed user interface.

7. CONCLUSION AND OUTLOOK

In this paper, we presented and evaluated a novel map-based user interface to explore the past 20 years of ISMIR publications. The proposed approach uses an MDS dimensionality reduction on a locally-restricted neighborhood of a query paper in order to project a high-dimensional feature space into 2-D while at the same time keeping projection distortions at a minimum. Consecutive maps are aligned in order to create a consistent impression of panning a global map.

Based on the interface, the user is able to find out a number of interesting topics in the ISMIR collection that would be hard to identify using a linear search interface. As an example, the topic "query by humming" almost disappeared in the early 2010's after having peaked in the early 2000's. While this fact would be hard to extract manually from a linear search interface, it is revealed at a single glance using the proposed prototype.

Besides updating the data to include recent ISMIR publication, one possibility for extension is to consider MIR-related papers not published at ISMIR. Respective candidates could be discovered by analyzing the reference sections. This would also allow an additional co-reference analysis which could enrich the interface further. We will also provide support to integrate the ISMIR Explorer into the official ISMIR website.

8. ACKNOWLEDGMENTS

We would like to thank all participants of the user study and the reviewers for their constructive feedback.

9. REFERENCES

[1] Marcia J. Bates. The design of browsing and berrypicking techniques for the online search interface. *Online Information Review*, 13(5):407–424, 1989.

[2] Imola K. Fodor. A survey of dimension reduction techniques. Technical Report UCRL-ID-148494, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, June 2002.

[3] John C. Gower. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51, 1975.

[4] Maarten Grachten, Markus Schedl, Tim Pohle, and Gerhard Widmer. The ISMIR cloud: A decade of ISMIR conferences at your fingertips. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR'09*, pages 63–68, 2009.

[5] I. T. Jolliffe. *Principal component analysis*. Springer Verlag, 1986.

[6] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[7] Tomoki Kondo, Hidetsugu Nanba, Toshiyuki Takezawa, and Manabu Okumura. Technical Trend Analysis by Analyzing Research Papers Titles. In Zygmunt Vetulani, editor, *Human Language Technology. Challenges for Computer Science and Linguistics*, number 6562 in Lecture Notes in Computer Science, pages 512–521. Springer Berlin Heidelberg, 2011.

[8] Joseph B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

[9] Thomas Low, Chistian Hentschel, Sebastian Stober, Harald Sack, and Andreas Nürnberger. Visual berrypicking in large image collections. In *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational (NordiCHI'14)*, pages 1043–1046, 2014.

[10] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[11] Yossi Rubner, Leonidas J. Guibas, and Carlo Tomasi. The earth mover's distance, multi-dimensional scaling, and color-based image retrieval. *Proceedings of the ARPA Image Understanding Workshop*, pages 661–668, 1997.

[12] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[13] Sebastian Stober, Thomas Low, Tatiana Gossen, and Andreas Nürnberger. Incremental visualization of growing music collections. In *14th Intl. Conf. on Music Information Retrieval*, ISMIR '13, pages 433–438, 2013.

[14] Jarkko Venna and Samuel Kaski. Local multidimensional scaling. *Neural Networks*, 19:889–899, 2006.

[15] Jarkko Venna, Jaakko Peltonen, Kristian Nybo, Helena Aidos, and Samuel Kaski. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *Journal of Machine Learning Research*, 11:451–490, 2010.

PATTERN CLUSTERING IN MONOPHONIC MUSIC BY LEARNING A NON-LINEAR EMBEDDING FROM HUMAN ANNOTATIONS

Timothy de Reuse, Ichiro Fujinaga
Centre for Interdisciplinary Research in Music Media and Technology, McGill University
{timothy.dereuse, ichiro.fujinaga}@mcgill.ca

ABSTRACT

Musical pattern discovery algorithms find instances of repetition in symbolic music, allowing for some user-specifiable amount of variation between identified repetitions; however, they can yield an intractably large number of discovered patterns when allowing for even small amounts of variation. This is commonly addressed by defining some heuristic notion of pattern significance, and returning only the most significant patterns. This paper develops a method of pattern discovery that models human judgement of what constitutes a significant pattern by incorporating annotations of repeated patterns, avoiding the need to design heuristics.

We take pattern discovery as a clustering task, where the input is a set of passages of monophonic music, represented as vectors of extracted features, and the output clusters correspond to discovered patterns. The human annotations are used to train a neural network to learn a low-dimensional embedding of the feature space that maps passages of music close together when they are occurrences of the same ground-truth pattern. The results of this approach match up with the annotations significantly better than the results of an approach using clustering without subspace learning. We provide examples of the types of patterns that this method tends to discover and discuss its feasibility and practicality as a tool for extracting useful information about repetitive structure in music.

1. INTRODUCTION

To discover patterns in a piece of music is, loosely speaking, to find passages that are similar to other passages, and to cluster these passages into inter-related groups. This is done with the goal of producing motivic analyses [34], automated composition systems [6], or as a single step to provide information for some larger Music Information Retrieval application [17]. Humans can perceive repetition between two musical passages even when the passages vary somewhat, so it is important to incorporate some allowable margin of variation between members of a single pattern; unfortunately, this causes a combinatorial explosion, returning many more patterns than are useful for any application. Inexact repetition simply occurs in music too often by chance; as [29] put it, "Most repetitions in music are not interesting." For this reason, discovery of patterns cannot be reduced to the problem of finding passages of music within a single piece that are similar to one another. Limiting the number of patterns found by a pattern discovery algorithm in a systematic way means explicitly or implicitly defining some measure by which one pattern can be judged as "more significant" than another; we refer to these as *pattern significance* measures.

Previous work by Collins et al. [10] suggests that analysts judge pattern significance consistently, based on quantifiable features of the musical surface. By analyzing ground-truth significant patterns, we can simultaneously learn a sense of what metric of melodic similarity the annotators used (by comparing occurrences of a single pattern) and a sense of the metric of pattern significance used (by comparing significant patterns to insignificant patterns), and use these findings to inform a pattern discovery method. We use these insights to model pattern discovery as a cluster analysis problem, where the input data points are a set containing all possible passages of music from the piece under investigation, and the output clusters correspond directly to discovered patterns. There are parallels between common issues in cluster analysis and pattern discovery that make this a sensible choice of technique: it is rarely known in advance how many clusters (or patterns) are present in a dataset (or piece), and validity measures for particular clusterings (or sets of discovered patterns) are the subject of ongoing investigation [15]. The goal here is that our clustering approach learns some criteria of what makes the particular patterns in the ground truth significant, and uses that criteria to return a limited number of patterns. Ideally, the model should be able to rediscover all the ground truth patterns from the songs that contain them, and not too many more.

The terms used to refer to sets of repeating musical passages vary widely throughout the literature. We will define a *pattern* as any set of musical excerpts, and refer to these excerpts as the pattern's *occurrences*. Each pattern is labelled as either *significant* or *trivial*. We ascribe no external meaning to a pattern's significance; if an analyst deems a pattern "significant," we assume that to mean nothing more and nothing less than "this analyst would consider this pattern to be of particular interest in this piece."

2. RELATED WORK

Early musical pattern discovery algorithms were string-based, dealing primarily with monophonic inputs [3, 12, 16, 22, 33, 34]. Geometric pattern discovery approaches were developed to deal with polyphonic music, most often representing music as sets of (onset time, MIDI pitch) or (onset time, morphetic pitch) pairs [5, 9, 21, 29, 36].

There are many proposed methods of reducing the number of patterns that are discovered. An early heuristic was based on the concept that patterns are more significant when they are unexpected, and so should be given more weight if their occurrences are statistically unlikely given the distribution of pitches and duration values elsewhere in the piece [11]. Other heuristics take inspiration from information theory, reasoning that a pattern is significant if it can be said to explain a great deal of the redundancy of the piece containing it [**?**, 1, 23, 26–28]. Research by Lartillot has attempted to emulate the cognitive processes that cause a listener to associate present material with past material stored in memory [18–20]. Velarde et al. use the Haar wavelet transform to analyze symbolic music for purposes of pattern discovery by considering the pitch of a monophonic melody as a signal; implicit in this approach is the assumption that this transform allows access to a higher-level musical property that is more relevant to perception of melodic similarity than raw pitch data [39, 40]. Clustering-based approaches are relatively uncommon, often using the clustering aspect for some form of visualization [2,4,17]. Directly comparing the performance of these algorithms is not straightforward. Evaluation against a set of human annotations is standard for the MIREX task in this area [8], but the ground truth used for evaluation there is sparse and taken from several different areas, making it difficult to extrapolate the accuracy of a single method to the method's quality as a whole.

The largest study that investigated human annotations of pattern importance was performed by Collins et al. [10], who asked 20 musically trained subjects to classify patterns in one of Chopin's Mazurkas based on how likely they would be to discuss each pattern if asked to write an essay analysing the whole piece. Principal findings from this study were that a small number of features could be used to explain 70% of the variation between the patterns' importance ratings. The results from this study do not constitute a pattern discovery method in and of themselves, but they speak to the possibility that human significance judgements might be consistent enough to inform a pattern discovery method.

3. APPROACH

This section describes the setup necessary to define our proposed approach. Three main steps are necessary: assembly and feature extraction on the data set, defining a training method that learns an embedding of our feature space, and clustering on the embedded data set.

3.1 Dataset Assembly

The dataset under investigation is the Meertens Tune Collection Annotated Corpus (MTC-ANN) [38], which comprises 93 patterns among 360 monophonic Dutch folk songs in 26 tune families. Each pattern has, on average, 17.8 occurrences, and the average length of an occurrence is 4.14 notes. The small size of the occurrences in this corpus is worth concern; out of the 1657 total occurrences identified across all patterns, 433 of them are three notes long, and 323 of them contain only two notes. It is not obvious that such short snippets of music permit the extraction of useful information about pattern significance. Additionally, these patterns were found as part of a larger annotation process which emphasized finding features of songs that are useful in separating the songs into tune families [37, 41]; our approach uses the assumption that the same implicit significance measure was used for all annotations, but the annotators may have changed their criteria depending on the tune family under consideration. Still, the high number of occurrences per pattern is beneficial for our clustering method, since it is in general easier to detect a denser cluster than a sparser one. Since MTC-ANN contains only segment-like occurrences (i.e., occurrences that contain every note within a single time interval) we will deal only with this type of occurrence. Section 5 discusses how more general subsets of notes might be used instead.

We must find some trivial patterns to compare with this set of significant patterns. We operate under the assumption that any pattern not specified as significant in MTC-ANN is implicitly judged to be trivial. The SIARCT-C (Structure Induction Algorithm for R superdiagonals with Compactness Trawling and Categorization) algorithm, described in [9] and distributed in the PattDisc software toolkit [7], will be used to generate a set of negative examples. In MTC-ANN, patterns are discovered between songs that lie in the same tune families; to match this, and to avoid dealing with the gargantuan number of patterns that would be found looking in 360 songs at once, we find our trivial patterns strictly within the bounds of each tune family. In this process, SIARCT-C will likely find some of the motifs that were identified in MTC-ANN. We remove any pattern from our collection of trivial patterns if it matches one of the significant patterns too closely, where a match is registered if at least half of the occurrences are identical.

We stated previously that pattern discovery could be defined as a clustering problem by considering the input to be "all possible passages of music" extracted from a given input. The occurrences of these discovered trivial patterns will serve as a stand-in for this set of all possible passages to avoid having to work with it directly. This should not cause any loss of generality, since the huge number of patterns returned by SIARCT-C does not significantly help us narrow down the search space. This run of SIARCT-C can be interpreted as a pre-processing step on the set of all possible musical passages from the dataset, where passages are removed if they do not repeat often enough to have a chance at being part of a significant pattern.

3.2 Feature Extraction

Instead of representing each occurrence of our patterns as an ordered sequence of notes and durations, we represent them as feature vectors, to allow the inclusion of information about the context of each occurrence; for example, we quantify how much its pitch range differs from that of the song containing it. Most of the features we extract are defined in the documentation of jSymbolic2, a software tool for extraction of features from symbolic music files [24, 25], but others were devised in previous work specifically for the purpose of extracting useful information from very short passages of music [13]. They fall into five broad categories, here listed with the number of features each contains:

- **Pitch-Related:** Features relating solely to the ordered pitch values of each note (n = 21).
- **Rhythm-Related:** Features relating solely to the ordered duration values and metrical positions of each note (n = 10).
- **Contour-Related:** Features using both pitch and duration values to describe how the sequence changes over time (n = 6).
- **Histograms:** Multi-valued features indicating the raw number of notes with a particular pitch class or duration (n = 31).
- **Context-Related:** Features comparing properties of the occurrence to properties of the song containing it (n = 38).

All together, this yields a feature set of size 106. A detailed list of features and their definitions is provided in the supplementary material to this paper.

3.3 Learning an Embedding

We now have a dataset containing 11,471 short passages of music, each represented as a 106-dimensional feature vector. 1,657 of these are categorized into one of 93 significant patterns, where all members of a single pattern are considered similar to one another and dissimilar to members of any other pattern. 9,814 of our data points are from trivial patterns, and we consider each of these to be dissimilar to every other data point.

The neural network used to learn the embedding is a fully-connected feed-forward network with two hidden layers, each containing 100 nodes, using dropout and batch normalization, and an output layer of size five. The training process for this network takes pairs of data points as input. We use three types of pairs, in equal measure: pairs where both points are members of the same ground-truth pattern (labelled as similar), where both points are members of different ground-truth patterns (labelled as dissimilar), and where one point is from a ground-truth pattern and one is from a trivial pattern (labelled as dissimilar). Both data points are fed separately through the hidden layers, transforming them both into lower-dimensional vectors of length 5, and then their difference is taken; the L1 norm of this difference is the output of the network. Training implements a hinge loss that encourages the output to be near zero if the two input data points are labelled as similar, and encourages the output to be above some margin value (here set to 1) if the points are labelled as dissimilar. Training is halted when the loss on a validation set does not decrease for 1,000 epochs.

The effect of this process is to train the network to learn an embedding of the 106-dimensional feature space into a 5-dimensional space where occurrences of significant patterns are clustered together, and all clusters are placed far away from one another.

3.4 Clustering

We use the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm to cluster our transformed data points [14]. DBSCAN labels sparser areas of the dataset as containing unlabelled noise, which is ideal for this application, as we expect that most data points will represent musical content not part of significant patterns.

DBSCAN takes two parameters; a minimum cluster size, which we set to 3 based on the size of the smallest patterns in our ground truth, and a value ϵ which, roughly, characterizes how close together points must be near the centers of the discovered clusters. Choosing an optimal ϵ is not straightforward; too high and the data will be partitioned into a few large clusters, but too low and most points in the dataset will be taken as noise. The designers of the algorithm recommend the use of a *k-dist graph* to estimate an optimal value. The k-dist graph of a dataset is formed by finding the k-th nearest neighbor of each data point, calculating the distance between each point and its identified neighbor, and sorting these distances in descending order. On a dataset well-suited to clustering, this sequence of distances should form a curve with a single sharp bend; to one side of this bend, where the distances are high, most points are noise points not part of any cluster, and on the other side of the bend, most points are close to their k-nearest neighbor and are likely members of well-defined clusters. The recommended value for ϵ is the value of the graph at this point of bending. Since most points in our dataset might be considered noise, we will evaluate our method over a range of values for ϵ that are further down the curve than this bending point, which will force clusters to be more tightly packed.

An image of the k-dist graph for one of the experiments run in Section 4 is included in the supplementary material to this paper.

4. EVALUATION

We use cross-validation to evaluate the method's performance. The data is split into training / validation / testing sets based on tune family, to ensure that we train the clustering method on the patterns contained in a particular set of songs and then test that method by discovering patterns in a totally separate set of songs. Each set includes all pat-

	Num. Clusters Ratio	Median Cluster Size	All Points		Significant Points	
			Homogeneity	Completeness	Homogeneity	Completeness
Embedding						
ϵ_5	**2.14 ± 0.39**	**6.50 ± 0.40**	0.25	0.12	0.37	0.62
ϵ_{10}	4.76 ± 1.02	5.90 ± 0.21	0.44	0.13	0.63	**0.67**
ϵ_{15}	6.68 ± 1.37	6.00 ± 0.28	0.53	0.13	**0.69**	0.66
ϵ_{20}	8.66 ± 1.67	5.20 ± 0.18	0.54	0.12	**0.69**	0.63
ϵ_{25}	10.15 ± 2.02	5.10 ± 0.08	**0.56**	0.12	0.61	0.64
PCA						
ϵ_5	2.70 ± 0.76	4.50 ± 0.20	0.15	0.13	0.21	0.53
ϵ_{10}	4.97 ± 1.23	4.60 ± 0.21	0.28	0.13	0.37	0.58
ϵ_{15}	6.94 ± 1.39	4.40 ± 0.22	0.37	**0.14**	0.47	0.60
ϵ_{20}	8.42 ± 1.70	4.40 ± 0.21	0.45	**0.14**	0.55	0.61
ϵ_{25}	10.96 ± 2.33	4.40 ± 0.22	0.52	0.12	0.63	0.62

Table 1: Results of the experiments described in Section 4. Plus-minus signs (±) indicate the standard error of a statistic over the five test sets. Standard errors for the homogeneity and completeness statistics are consistently $\ll 0.05$ and are omitted for readability.

terns, significant and trivial, that lie in its designated tune families.

The training and validation sets must be assembled into pairs before the neural network can take them as input. We generate three sets of pairs corresponding to the categories defined in section 3.3. We take all possible unique pairs of the first category that our training set permits—that is, all possible pairs involving two distinct occurrences of the same significant pattern—and reduce the next two categories to the same size as the first through random sampling without replacement. The total number of data point pairs generated via this process varies depending on which tune families are selected from MTC-ANN for validation and training, since every tune family has a different number of identified patterns, but in practice this number lies in the range from $20,000$ to $40,000$.

Once the network has been trained on this set of pairs, we use it to reduce the test set into vectors that lie in the learned subspace. We run DBSCAN with five different values of ϵ, which are estimated by building a k-dist graph on the test set with k=3. For all test sets, this graph has a very sharp bend near the 5th percentile. Denoting the value at the nth percentile of the k-dist graph as ϵ_n, we test DBSCAN with values of $\epsilon_5, \epsilon_{10}, \epsilon_{15}, \epsilon_{20}$, and ϵ_{25}. Each of these clusterings can finally be compared directly with the patterns in the test set.

We contrast this method with one that uses Principal Component Analysis (PCA) to reduce the dimensionality of the dataset instead of a learned embedding. The test sets are processed with PCA and the five components of highest magnitude are retained. DBSCAN is used to cluster the result, with the same procedure for estimating ϵ as before, using a k-dist graph built on the PCA-reduced data.

Testing proceeds with 5-fold cross-validation. This does not evenly divide the number of tune families (26), but each tune family has a different number of patterns, and each pattern has a different number of occurrences, so no truly equitable division is possible anyway. The neural network is implemented using PyTorch 1.0 [30], while the implementation of DBSCAN is from scikit-learn [31]. Code for running these experiments is available at `https://github.com/timothydereuse/musical-pattern-clustering`.

4.1 Results

In Table 1, the *Num. Clusters Ratio* column compares the number of clusters obtained in each experiment to the number of ground-truth patterns in each test set. The *Median Cluster Size* represents the median number of occurrences within each pattern, averaged over all test sets. This somewhat awkward metric is necessary because using a straight average would skew too high to accurately represent the data; on every test set, both the PCA and embedding approaches tend to return one or two large patterns with hundreds of "noisy" occurrences, a phenomenon further discussed in Section 4.2.

Traditional metrics of classification accuracy such as precision and recall are not applicable in a clustering task where the number of classes is itself being predicted. The metrics used here are *homogeneity* and *completeness*, two complementary measures of clustering validity which compare a clustering to a ground-truth [35]. The homogeneity of a clustering measures the degree to which clusters in the output are comprised of a single class, assigning a score of 1 if every pattern in the output contains data points from only a single ground-truth pattern, even if some patterns are split apart. Similarly, the completeness measures the degree to which classes in the input are mapped to a single cluster of the output, assigning a score of 1 if every pattern in the input stays unbroken when mapped to patterns in the output, even if some patterns get merged together. We evaluate these two metrics against the ground truth in two different ways: first, considering all points in the test set, and then considering only the points

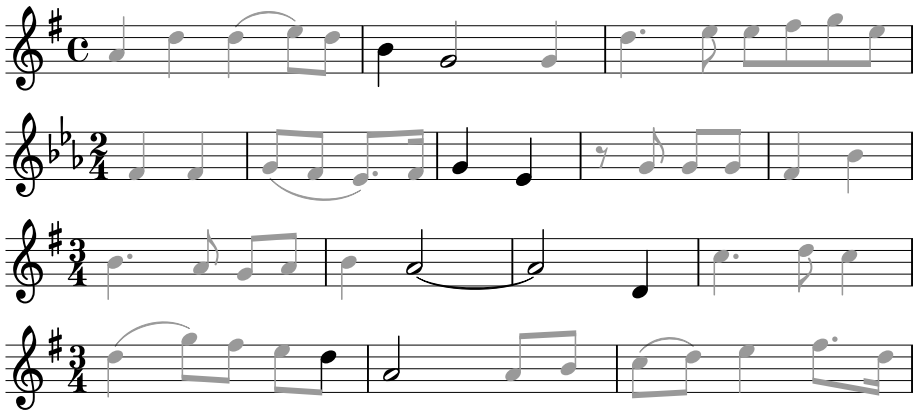
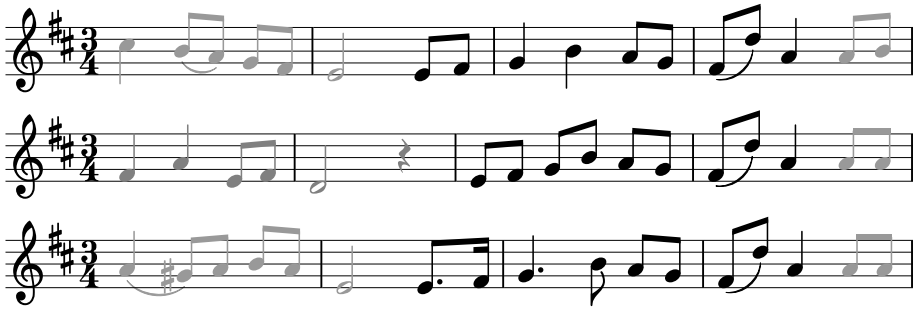
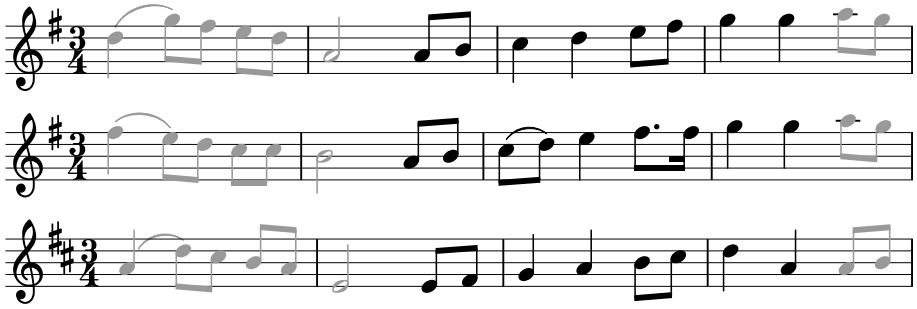


Figure 1: These four two-note occurrences are part of different patterns in MTC-ANN, but the embedding method merges all of their patterns into one.



(a) Three occurrences from a pattern in MTC-ANN erroneously included in a large, noisy cluster by the embedding method.



(b) Three occurrences from the same pattern in MTC-ANN correctly clustered together by the embedding method.

Figure 2

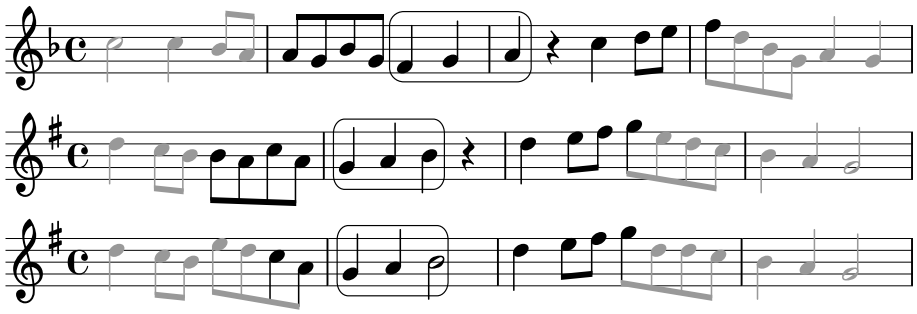


Figure 3: This pattern found by the embedding method is not present in MTC-ANN; however, there is a short three-note pattern (marked by boxes) included in MTC-ANN whose occurrences lie *within* this pattern.

that correspond to significant patterns in the ground truth. Note the very low score for completeness across both the embedding and PCA methods; because the majority of the points in the dataset are labelled as noise, misclassifying any of them into clusters effectively "breaks up" the noise class and lowers the completeness. If we ignore the noise and focus solely on where significant patterns are clustered, we note that they are mostly preserved. The supplementary material to this paper contains an additional table showing the effects of excluding individual feature categories (as defined in Section 3.2) on the clustering.

4.2 Examples

We pick one of the test sets from the row ϵ_{15} of Table 1 to investigate further. This particular clustering finds 21 patterns within six tune families that together contain a total of 77 songs; within these six tune families, MTC-ANN notes 18 significant patterns. Most of the 21 patterns in this clustering do not correspond to patterns in MTC-ANN. In particular, two of the patterns have over 800 occurrences each, almost entirely containing longer occurrences from trivial patterns, with some longer occurrences from significant patterns scattered in as well. In each figure, occurrences are marked with black notes, whereas greyed-out notes show the context in which each occurrence lies.

Only six of the 21 identified patterns have significant overlap with the patterns in MTC-ANN. One reason for this low number is that the clustering method has merged some of the identified patterns together. Figure 1 shows four occurrences from one of these six clusters; all four of these occurrences lie in different patterns of the ground truth, and the full cluster (not shown here, for lack of space) comprises the union of these four original ground-truth patterns. Since all of these occurrences contain only two notes, it is likely that these particular patterns were designated as significant due to their metrical placement in their original songs. It is incorrect that these patterns were merged together, but the fact that *only* these patterns were merged together is notable. The trivial patterns found by SIARCT-C have no shortage of descending intervals that the algorithm might have added to this particular cluster, and yet it contains only descending intervals that were marked as significant by human annotators. This suggests that our subspace-learning neural network has learned something from the "context-related" features mentioned in Section 3.2 that relates to how the human annotators decided which two-note intervals in the original songs merited significance: not enough to separate these four patterns from each other, but enough to separate them from the rest of the dataset as a group.

Figure 2b shows another notable error made in this clustering. One of the ground-truth patterns in this test set is quite large and heterogeneous, comprising 20 occurrences each containing eight or nine notes. Where the occurrences have a relatively simple contour, the clustering correctly groups them together, but it groups the more complicated occurrences with other unclassifiable, long passages in the size-800 clusters mentioned above. It is likely that, given the small size of the dataset, the learned subspace does not encode a particularly complex conception of melodic similarity, which means that longer patterns are unlikely to cluster together unless their similarities are quite pronounced. Compare the three occurrences in Figure 2a to those in Figure 2b, which were successfully clustered into a single pattern, likely as a result of their more uniform contour, and those in Figure 3, where similarity in contour appears to have caused the embedding method to *extend* a pattern existing in MTC-ANN.

5. CONCLUSIONS

We have demonstrated an approach to discovering patterns in symbolic music that maps passages of music onto a low-dimensional subspace where significant patterns form clusters, using an embedding learned from human annotations of repeated patterns. This method outperforms a traditional dimensionality reduction algorithm on common metrics used to validate clustering results against ground truth. There is evidence that the method is capable of learning some notions of pattern significance from the human annotations; though the agreement is far from perfect, and the number of returned patterns is still high, the current state of the art in pattern recognition struggles to agree with human annotations at all [32]. To more rigorously validate this approach in future research, it would be informative to compare a clustering learned from human annotations with a clustering that uses a distance measure derived from an existing melodic similarity metric.

If we continue to restrict ourselves to segment-like occurrences, then extending this approach to polyphonic music would require only a feature set capable of encoding information about polyphonic occurrences. However, to be able to find patterns within polyphonic sources more generally, we must consider occurrences as subsets of notes instead, which is combinatorially infeasible; for a piece of music with n note onsets, there are $O(n^2)$ possible segments, but $O(2^n)$ possible subsets. To address this, it would be necessary to impose limits on the time-extent and number of notes in each occurrence, or to use an existing polyphonic pattern discovery algorithm as a p reprocessing step, as we do here with SIARCT-C.

More ground-truth annotations would undoubtedly increase the accuracy of this approach, but annotations of repeated patterns are expensive to acquire, and information from one set of annotations might not generalize well to other genres. The ability to extract useful information from small sets of repeated patterns would be a more valuable tool for development of practical pattern significance measures. A hypothetical use case for this research would be an interactive pattern-discovery interface where, in lieu of changing the parameters of a manually designed heuristic, users could view a list of patterns and mark some as significant, whereupon the algorithm would re-train on that small set and use its findings to reduce the number of patterns returned to the user.

6. REFERENCES

[1] Teppo E. Ahonen, Kjell Lemström, and Simo Linkola. Compression-based similarity measures in symbolic, polyphonic music. In *Proc. of the 12th International Society for Music Information Retrieval Conference*, pages 91–96, Miami, FL, 2011.

[2] Chantal Buteau and John Vipperman. Melodic clustering within motivic spaces: Visualization in OpenMusic and application to Schumann's Träumerei. In *Proc. of the International Conference on Mathematics and Computation in Music*, Communications in Computer and Information Science, pages 59–66, 2009.

[3] Emilios Cambouropoulos. Musical pattern extraction for melodic segmentation. In *Proceedings of the XII Colloquium of Musical Informatics*, Gorizia, Italy, 2003.

[4] Emilios Cambouropoulos and Gerhard Widmer. Automated motivic analysis via melodic clustering. *Journal of New Music Research*, 29(4):303–317, 2000.

[5] Shih-Chuan Chiu, Man-Kwan Shan, Jiun-Long Huang, and Hua-Fu Li. Mining polyphonic repeating patterns from music data using bit-string based approaches. In *2009 IEEE International Conference on Multimedia and Expo*, pages 1170–1173, 2009.

[6] Tom Collins. *Improved methods for pattern discovery in music, with applications in automated stylistic composition*. PhD Thesis, The Open University, 2011.

[7] Tom Collins. PattDisc-Jun2014. Available online at http://www.tomcollinsresearch.net/publications.html. Accessed 18 November 2017, 2014.

[8] Tom Collins. 2017: Discovery of repeated patterns and sections. http://www.music-ir.org/mirex/wiki/2017:Discovery_of_repeated_themes_\%26_sections. Accessed 3rd March 2019, 2017.

[9] Tom Collins, Andreas Arzt, Sebastian Flossmann, and Gerhard Widmer. SIARCT-CFP: Improving precision and the discovery of inexact musical patterns in point-set representations. In *Proc. of the 14th International Society for Music Information Retrieval Conference*, pages 549–554, Taipei, Taiwan, 2013.

[10] Tom Collins, Robin Laney, Alistair Willis, and Paul H. Garthwaite. Modeling pattern importance in Chopin's mazurkas. *Music Perception: An Interdisciplinary Journal*, 28(4):387–414, 2011.

[11] Darrell Conklin and Christina Anagnostopoulou. Representation and Discovery of Multiple Viewpoint Patterns. In *Proc. of the International Computer Music Conference*, Havana, Cuba, 2001.

[12] Darrell Conklin and Christina Anagnostopoulou. Segmental pattern discovery in music. *INFORMS Journal on Computing*, 18(3):285–293, 2006.

[13] Timothy de Reuse. *A machine learning approach to pattern discovery in symbolic music*. Master's Thesis, McGill University, Montreal, Canada, 2019.

[14] Martin Ester, Hans-Peter Kriegel, Xiaowei Xu, and Jörg Sander. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[15] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.

[16] Jia-Lien Hsu, Chih-Chin Liu, and A.L.P. Chen. Discovering nontrivial repeating patterns in music data. *IEEE Transactions on Multimedia*, 3(3):311–325, 2001.

[17] Yong-Kyoon Kang, Kyong-I Ku, and Yoo-Sung Kim. Extracting theme melodies by using a graphical clustering algorithm for content-based music information retrieval. In Albertas Caplinskas and Johann Eder, editors, *Advances in Databases and Information Systems*, Lecture Notes in Computer Science, pages 84–97. Springer: Berlin Heidelberg, 2001.

[18] Olivier Lartillot. A musical pattern discovery system founded on a modeling of listening strategies. *Computer Music Journal*, 28(3):53–67, 2004.

[19] Olivier Lartillot. Automated motivic analysis: An exhaustive approach based on closed and cyclic pattern mining in multidimensional parametric spaces. In David Meredith, editor, *Computational Musical Analysis*, pages 273–302. Springer: Switzerland, 2016.

[20] Olivier Lartillot and Petri Toiviainen. Motivic matching strategies for automated pattern extraction. *Musicae Scientiae*, 11(1):281–314, 2007.

[21] Kjell Lemström and Jorma Tarhio. Searching monophonic patterns within polyphonic sources. In *Proc. of the 6th Conference on Content-based Multimedia Information Access*, pages 1261–1279, 2000.

[22] Chih-Chin Liu, Jia-Lien Hsu, and Arbee LP Chen. Efficient theme and non-trivial repeating pattern discovering in music databases. In *Proc. of the International Conference on Data Engineering*, pages 14–21. IEEE, 1999.

[23] Corentin Louboutin and David Meredith. Using general-purpose compression algorithms for music analysis. *Journal of New Music Research*, 45(1):1–16, 2016.

[24] Cory McKay. *Automatic genre classification of MIDI recordings*. Master's Thesis, McGill University, Montreal, Quebec, 2004.

[25] Cory McKay, Tristano Tenaglia, and Ichiro Fujinaga. jSymbolic2: Extracting features from symbolic music representations. In *Proc. of the 17th International Society for Music Information Retrieval Conference*, Late-Breaking Session. New York City, NY, 2016.

[26] David Meredith. Music analysis and Kolmogorov complexity. In *Proceedings of the 19th Colloquio di Informatica Musicale*, 2012.

[27] David Meredith. Analysis by compression: Automatic generation of compact geometric encodings of musical objects. In *Proc. of The Music Encoding Conference*, Trieste, Italy, 2013.

[28] David Meredith. COSIATEC and SIATECCompress: Pattern discovery by geometric compression. In *Proc. of the 14th International Society for Music Information Retrieval Conference*, Curitiba, Brazil, 2013.

[29] David Meredith, Kjell Lemström, and Geraint A. Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, 31(4):321–345, 2002.

[30] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in Pytorch. In *Proc. of the 31st Conference on Neural Information Processing Systems*, Long Beach, CA, 2017.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[32] Iris Yuping Ren, Hendrik Vincent Koops, Anja Volk, and Wouter Swierstra. In Search of the Consensus Among Musical Pattern Discovery Algorithms. In *Proceedings of the 18th International Symposium on Music Information Retrieval*, Suzhou, China, 2017.

[33] Pierre-Yves Rolland. Discovering patterns in musical sequences. *Journal of New Music Research*, 28(4):334–350, 1999.

[34] Pierre-Yves Rolland and Jean-Gabriel Ganascia. Automated motive oriented analysis of musical corpuses: A jazz case study. In *Proc. of the International Computer Music Conference*, pages 240–243, 1996.

[35] Andrew Rosenberg and Julia Hirschberg. V-Measure: A conditional entropy-based external cluster evaluation measure. In *Proc. of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 410–420, Prague, Czechia, 2007.

[36] Esko Ukkonen, Kjell Lemström, and Veli Mäkinen. Geometric algorithms for transposition invariant content-based music retrieval. In *Proc. of the 4th International Conference on Music Information Retrieval*, pages 193–199, Baltimore, MD, 2003.

[37] Peter van Kranenburg. *A computational approach to content-based retrieval of folk song melodies*. Doctoral Thesis, Utrecht University, 2010.

[38] Peter van Kranenburg, Martine de Bruin, Louis P. Grijp, and Frans Wiering. The Meertens tune collection. *Meertens Online Reports*, 1, 2014.

[39] Gissel Velarde and David Meredith. A wavelet-based approach to the discovery of themes and sections in monophonic melodies. *Music Information Retrieval Evaluation Exchange (MIREX 2014), Taipei, Taiwan*, 2014.

[40] Gissel Velarde, Tillman Weyde, and David Meredith. An approach to melodic segmentation and classification based on filtering with the Haar-wavelet. *Journal of New Music Research*, 42(4):325–345, 2013.

[41] Anja Volk and Peter van Kranenburg. Melodic similarity among folk songs: An annotation study on similarity-based categorization in music. *Musicae Scientiae*, 16(3):317–339, 2012.

A STUDY OF ANNOTATION AND ALIGNMENT ACCURACY FOR PERFORMANCE COMPARISON IN COMPLEX ORCHESTRAL MUSIC

Thassilo Gadermaier[1] **Gerhard Widmer**[1,2]
[1] Institute of Computational Perception, Johannes Kepler University Linz, Austria
[2] Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria
thassilo.gadermaier@jku.at, gerhard.widmer@jku.at

ABSTRACT

Quantitative analysis of commonalities and differences between recorded music performances is an increasingly common task in computational musicology. A typical scenario involves manual annotation of different recordings of the same piece along the time dimension, for comparative analysis of, e.g., the musical tempo, or for mapping other performance-related information between performances. This can be done by manually annotating one reference performance, and then automatically synchronizing other performances, using audio-to-audio alignment algorithms. In this paper we address several questions related to those tasks. First, we analyze different annotations of the same musical piece, quantifying timing deviations between the respective human annotators. A statistical evaluation of the marker time stamps will provide (a) an estimate of the expected timing precision of human annotations and (b) a ground truth for subsequent automatic alignment experiments. We then carry out a systematic evaluation of different audio features for audio-to-audio alignment, quantifying the degree of alignment accuracy that can be achieved, and relate this to the results from the annotation study.

1. INTRODUCTION

An increasingly common task in computational musicology – specifically: music performance analysis – consists in annotating different performances (recordings) of classical music pieces with structural information (e.g., beat positions) that defines a temporal grid, in order then to carry out some comparative performance analyses, which require time alignments between the performances. As manually annotating many recordings is a very time-consuming and tedious task, an obvious shortcut would be to manually annotate only one performance, and then use automatic audio-to-audio matching algorithms to align additional recordings to it, and thus also be able to automatically transfer structural annotations.

The work presented here is part of a larger project on the analysis of orchestral music performance. In this musicological context, it is crucial to understand the level of precision one can expect of the empirical data collected. The present study attempts to answer two specific questions: (1) what is the precision / consistency we can expect from human time annotations in such complex music? and (2) can automatic alignment be precise enough to be used for transferring annotations between recordings, instead of tediously annotating each recording manually? We will approach this by collecting manual annotations from expert musicians, on a small set of carefully selected pieces and recordings (Section 3), analyzing these with statistical methods (Section 4) – which will also supply us with a ground truth for the subsequent step –, then performing systematic experiments with different audio features and parameters for automatic audio-to-audio alignment (Section 5), quantifying the degree of alignment precision that can be achieved, and relating this to the results from the previous annotation study (Section 6).

2. RELATED WORK

[13] presented a case study of opera recordings that were annotated by five annotators, at the bar level. The authors used the mean values over the annotators as ground-truth values for the respective marker positions and the variance to identify sections possibly problematic to annotate, and offered a qualitative analysis of the musical material and sources for error and disagreement between annotators.

[6] deals with the alignment of recordings with possibly different structure. Their contribution is relevant for our endeavor in so far as they evaluated different audio features and parameters ranges for an audio-to-audio alignment task on a data set of, among others, symphonies by Beethoven, which matches our data set very well. [7] evaluated audio features for the audio-to-audio alignment task using several different data sets.

While many studies of alignment features do not use real human performances but artificial data, we only use ground-truth produced from human annotations (by averaging over multiple annotations per recording) of existing recordings for the evaluation of the alignment task. Furthermore, the results of our analysis of manual annotations (Step 1) will inform our interpretation of the automatic alignment experiments in Step 2 (by relating the observed

alignment errors to the variability within the human annotations), leading to some insights useful for quantitative musicological studies.

3. ANNOTATION AND GROUND-TRUTH

3.1 Annotation vs. Tapping

Our primary goal is to map the musical time grid as defined by the score, to one or more performances given as audio recordings. Due to expressiveness performance, these mapped time points may be very different between different recordings. Following [4], we will call the occurrence of one or more (simultaneous) score notes a *score event*. In our case, we were interested in annotating regularly spaced score events, for instance, on the quarter note beats.

Different methods can be employed for marking score events in a recording. One possibility is to tap along a recording on a keyboard (or other input device) and have the computer store the time-stamps. We will refer to a sequence of time-stamps produced this way as a *tapping* in the following. Producing markers this way has been termed "reverse conducting" by the Mazurka project [1] .

This is to be distinguished from what we will call an *annotation* throughout this paper. In that case, markers are first placed by tapping along, or even by visually inspecting the audio waveform, and then iteratively corrected on (repeated) critical listening. In general, we assume corrected annotations to have smaller deviations from the "true" time-stamps than uncorrected tappings, especially around changes of tempo.

3.2 Pieces, Annotators, and Annotation Process

The annotation work for this study was distributed over a pool of four annotators. Three are graduates of musicology and one is a student of the violin. The pieces considered are: Ludwig van Beethoven's Symphony No. 9, 1st movement; Anton Bruckner's Symphony No. 9, 3rd movement; and Anton Webern's Symphony Op. 21, 2nd movement (see Table 1 for details). The first two are symphonic movements, played by full classical/romantic period orchestra. The third is an atonal piece where the second movement is of a "theme and variations" form, and requires a much smaller ensemble (clarinets, horns, harp, string section). While the first two pieces can be considered to be well known even to average listeners of classical music, the Webern piece was expected to be less familiar to the annotators. It is rhythmically quite complicated, with many changes in tempo and many sections ending in a fermata. We expected it to be a suitable challenge for the annotators as well as the for the automatic alignment procedure.

The quarter beat level was chosen as (musically reasonable) temporal annotation granularity, in all three cases. The annotators were asked to mark all score events (notes or pauses) at the quarter beat level, using the Sonic Visualiser software [2], and then to correct markers such that they coincide with the score events when listening to the playback with a "click" sound together with the recording of the piece. They also had to annotate "silent" beats (i.e. general pauses) or even single or multiple whole silent bars with the given granularity. It is clear that this may create large deviations between annotators at such points, as the way to choose the marker positions is not always obvious or even meaningfully possible in these situations.

Each recording was annotated by three annotators, giving us a total of 21 complete manual annotations [2] .

[1] www.mazurka.org.uk/info/revcond/example/

Composer	Piece	Part	Section	# Events
Beethoven	Sym. 9	1st mov.	complete	1093
A. Bruckner	Sym. 9	3rd mov.	150 - end	371
A. Webern	Sym. Op. 21	2nd mov.	complete	198

Table 1. Annotated works/parts, and number of events. Granularity in all cases: quarter notes.

Composer	Conductor	Orch.	Year	Dur.	Med. SD
Beethoven	Karajan	VPO	'47	16:00	32
	Karajan	BPO	'62	15:28	32
	Karajan	BPO	'83	15:36	27
A. Bruckner	Karajan	BPO	'75	09:30	68
	Abbado	VPO	'96	10:40	52
A. Webern	Boulez	LSO	'69	03:08	47
	Karajan	BPO	'74	03:28	63

Table 2. Annotated recordings. VPO = Vienna Philharmonic Orchestra, BPO = Berlin Philharmonic, LSO = London Symphony Orchestra. Each recording was annotated by three annotators. Med. SD is the median value of standard deviations of the annotations (in milliseconds, rounded to nearest integer), for details see Sec. 4.

4. EVALUATION OF ANNOTATIONS

For a statistical analysis of this rather small number of human annotations, we need to make some idealizing assumptions. We assume that there is one clear point in time that can be attributed to each respective score event, i.e. there are "true" time-stamps τ_n, $n = 1, 2, \ldots$ for the score events we sought to annotate. If each score event is annotated multiple times, the annotated markers θ_n will show random variation around their true time-stamps, with a certain variance σ_n^2. It seems reasonable to assume the respective markers to be realizations of random variables Θ_n, each following a normal distribution, i.e. $\Theta_n \sim \mathcal{N}(\tau_n, \sigma_n^2)$.

Thus, for each event to be annotated we would expect (a large number of) annotations to exhibit a normal distribution around some mean τ_n. This is schematically illustrated in Figure 2.

However, for estimating the parameters of these distributions, rather large numbers of annotations would be required.

[2] Supplemental material to this publication is available online at 10.5281/zenodo.3260499

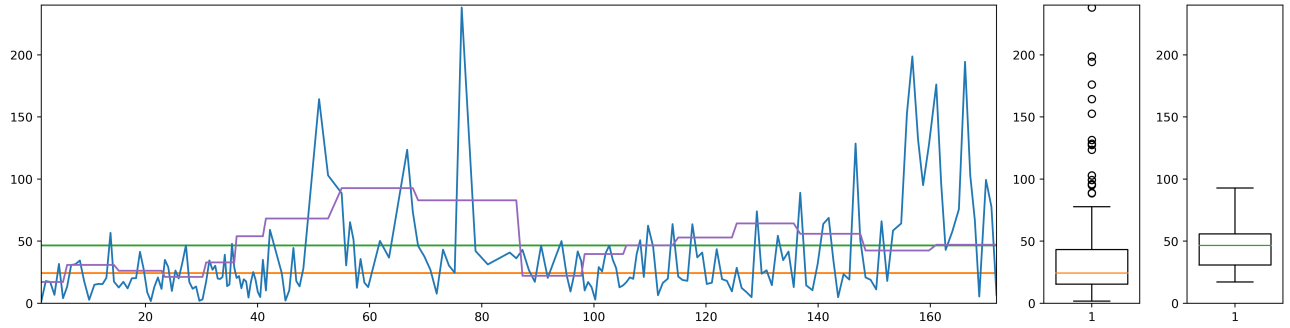


Figure 1. Standard deviations of annotations along a performance, Webern Op21-2, Boulez. Blue: computed from three markers per score event. Magenta: computed from pooled differences (details see text). Orange: median standard deviation (SD), green: median of SD from pooled differences (see boxplots right). Right: boxplots for SD (summary of blue curve) and for SD of pooled values (summary of magenta curve), central quartile is median.

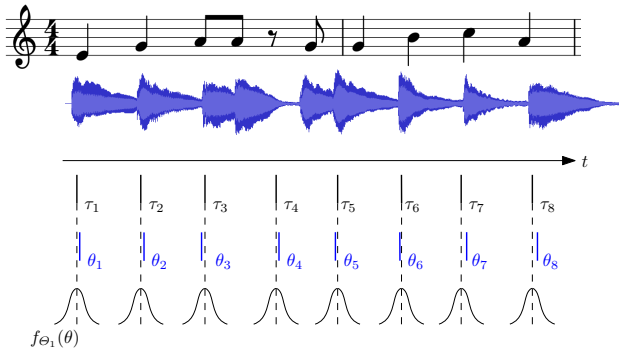


Figure 2. Modeling annotations as random variables. Musical score and waveform of a performance. Hypothetical true time-stamps τ_n. Annotation markers θ_n. Bottom row: pdfs of random variables Θ_n, each of mean τ_n.

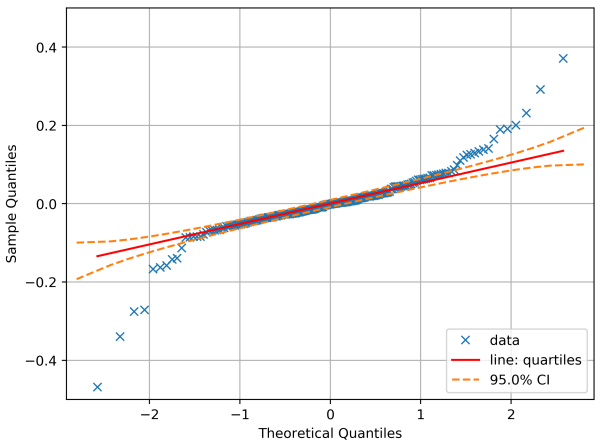


Figure 3. Webern Op21-2, Boulez. Quantile-quantile plot of the differences of a pair of annotation sequences for the whole piece. Solid red line fitted to first and third data quartile, dashed lines show ±95% confidence around this line. Non-normal data deviate strongly from area enclosed by dashed lines.

[3] has shown that with some additional assumptions, the distribution can be estimated from as little as two sequences of markers.

We follow [3] in the derivations below. If the variance σ^2 of the time stamps is assumed to be constant over time (across the whole piece or part to be annotated), subtracting two sequences θ_n^1, θ_n^2 of markers for the same score events, i.e.

$$\Delta_n = \Theta_n^1 - \Theta_n^2, \tag{1}$$

yields the variable $\Delta_n \sim \mathcal{N}(0, 2\sigma_\Theta^2)$. Note that if the mean of Δ_n is not zero, we can force it to be by suitably offsetting either Θ_n^1 or Θ_n^2 by $\bar{\Delta}_n$ – since we assume both sequences to mark the same events with mean zero, a total mean deviation can be viewed as a systematic offset by either annotator. One could then use the differences $\delta_n = \theta_n^1 - \theta_n^2$ to estimate the variance σ_Θ^2 around the true time-stamps:

$$\hat{\sigma}_\Theta^2 = \frac{1}{2N} \sum_{n=1}^{N} (\theta_n^1 - \theta_n^2)^2. \tag{2}$$

In [3], two example analyses of tap sequences were presented that support these assumptions.

We analyzed our annotation data according to these ideas. First, for each annotated recording, we calculated the time-stamp differences between each pair of annotations, according to Eq. (1), and tested the resulting distributions for normality, using the Shapiro-Wilk test. However, for all annotations created, none of the distributions is normal according to these tests. On visual inspection of the distributions of differences of annotation sequences δ_n using quantile-quantile plots (see Fig. 3), the tails of the distributions turned out to be typically significantly heavier than for a normal distribution.

We suspect that this discrepancy to the results given in [3] is most likely due to the higher complexity of our musical material, with large orchestras playing highly polyphonic and rhythmically complex music in varying tempi. It seems intuitively clear that for some sections, the deviations among annotated markers will be much smaller than in complex parts. Additionally, as we asked also silent

beats to be annotated, even during whole silent bars, we should expect substantial deviations for at least a few such events in every recording. We therefore conclude that at least the assumption of identical variance across a whole piece should be dropped (for more complex material) when more detailed information about local uncertainties of the annotation is desired.

However, it is interesting to note that locally, when the differences for only a few consecutive (around 20-30) annotated time-stamps are pooled, they conform to a normal distribution quite well. This means that the assumption of about equal variance for the annotation of score events tends to hold for short blocks of time, but rather not globally (for a whole piece), at least for the musical material considered here.

As estimating the standard deviation (as a measure of uncertainty) of each time-stamp's markers is not reliable given only few annotations, we used an alternative based on the above observation. For blocks of 24 consecutive score events (with a hop size of 12), the differences of a pair of annotation sequences were *pooled* and used to estimate the standard deviation for each respective block. The resulting, block-wise constant curve of standard deviations is shown in Fig. 1 (magenta), along with the simple standard deviation per score event, calculated from three markers (blue), for a specific recording and pair of annotations. The median of these per-block estimated standard deviations is used as a global estimate of the precision of the annotations for the respective performance, and is given for the respective performance as the right-most column in Table 2. As can be seen, the values differ substantially across the pieces as well as within the pieces, for different performances. The right-most boxplot in Fig. 1 shows a summarization of the per-block estimated standard deviations. Interestingly, for the 1st movement of Beethoven's Symphony 9 (with its relatively constant tempo), the estimated standard deviation is close to the value presented in [3], but it is considerably larger for the other pieces that exhibit more strongly varying tempo.

5. AUTOMATIC ALIGNMENT

As mentioned above, annotating a large number of performances of the same piece is a time-consuming process. A more efficient alternative would be to automatically transfer annotations from one recording to a number of unseen recordings, via audio-to-audio alignment.

5.1 Alignment Procedure and Ground-truth

The method of choice for (off-line) audio-to-audio alignment is *Dynamic Time-warping (DTW)* [10]. Aligning two recordings via DTW involves extracting sequences $X \in \mathbb{R}^{L \times D}$ and $Y \in \mathbb{R}^{M \times D}$ of feature vectors, respectively. Using a distance function $d(x_l, y_m)$, the DTW algorithm finds a path of minimum cost, i.e. a mapping between elements x_l, y_m of the sequences X, Y. An alignment is thus a mapping between pairs of feature vectors (from different recordings), each vector representing a block of consecutive audio samples. As each audio sample has an associated time-stamp (an integer multiple of the inverse of the sample rate), each feature vector, say x_l, can be associated with a time-stamp t_l^X as well, (here) representing the center of the block of audio samples. The matching of sequence elements is schematically illustrated in Fig. 4, for the "direction" $X \rightarrow Y$ (note that direction here refers to the evaluation, as will be illustrated next). For each block of X, the matching block of Y is found, and its associated time-stamp t_m^Y is subtracted from the ground-truth time-stamp g_n^Y. This produces the pairwise error sequence $e_n^{X \rightarrow Y}$. As we have ground-truth annotations for both recordings of a pair available, we can also calculate an error sequence for the "reverse" direction $Y \rightarrow X$.

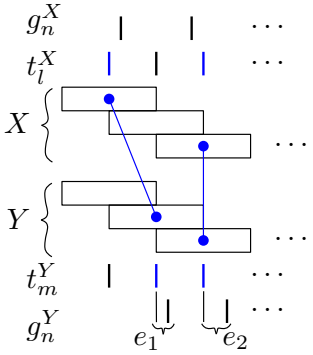


Figure 4. Matching feature vectors through DTW, and calculating errors between associated time-stamps t_m^Y and ground-truth time-stamps g_n^Y, for direction $X \rightarrow Y$. This yields the error sequence $e_n^{X \rightarrow Y}$.

The sequences of ground-truth time-stamps were produced from the multiple annotations discussed above (Section 3), by taking for each annotated score event the sample mean across the three annotations per recording. For computing the alignments, an implementation of FastDTW [12] in python was used.

5.2 Choice of Audio Features

The actual alignment process is preceded by extracting features from the recordings to be aligned. Different features have been proposed and evaluated for this task in the literature. We decided to choose only features that have proven to yield highly accurate alignments and thus small alignment errors.

[6] evaluated several different audio features separately on data sets of different music genres, among them symphonies by Beethoven. They achieved the best results overall by using 50 MFCC (in contrast to 13 or even 5 as used in [7]), for two different block lengths. As the results on these corpora, which are similar to ours, were dominated by MFCC, we decided to use these with similar configurations for our experiments. Additionally, we included a variant of MFCC (in the following addressed as "MFCC mod") following an idea described in [11], where 120 MFCC are extracted, then the first n_{skip} are discarded and only the remaining ones used. However, in contrast to their proposal we skip the subsequent extraction of chroma

information and use the MFCC directly.

The second family of features that has proven successful for alignment tasks are chroma features, which were tested as an alternative. For extracting the feature values, the implementations from LibROSA [9] were used. Besides "classical" chroma features, the variants chroma_cqt (employing a constant-Q transform) and chroma_cens were used. We decided not to include more specialized features that include local onset information, like LNSO / NC [1], or DLNCO (in combination with chroma), as they would seem to give no advantage on our corpus as suggested from the results in [6] and [5].

5.3 Systematic Experiments Performed

In order to find the best setup for audio-to-audio alignment for complex orchestral music recordings, we carried out a large number of alignment experiments, by systematically varying the following parameters:

- FFT sizes: 1024 to 8192 (chroma), up to 16384 (MFCC)
- Hop sizes MFCC: half of FFT size, for 16384 fixed to 4096
- Hop sizes Chroma: 512 and 1024, for each FFT size; additionally 2048 for chroma_cens and chroma_cqt
- Number of MFCC: 13, 20, 30, 40, 50, 80, 100
- MFCC mod: 120 coefficients, first 10, 20, ..., 80 discarded
- Distance measures: Euclidean (l_2), city block (l_1) and cosine distance.

Note that the audio signals were not down-sampled in any of the cases, but used with their full sample rate of 44.1 kHz.

All in all, a total number of 312 different alignments were computed and evaluated for each performance pair. Each alignment of each pair of performances was evaluated in both directions. As it is impossible to display all results in this paper, we will only report a subset of best results in Section 6.

6. EVALUATION OF ALIGNMENTS

6.1 Alignment Accuracy

For quantifying the alignment accuracy, we calculated pairwise errors e_n between the ground-truth time-stamps g_n for the respective recording and the matching alignment time-stamps t_l (see Fig. 4). Per pair of recordings, two error sequences are obtained, one for each evaluation direction, i.e. $e_n^{X \to Y}$ and $e_n^{Y \to X}$. As a general global measure of the accuracy of a full alignment, the mean absolute error is used, where the maximum absolute error can be seen as a measure for lack of robustness.

For reporting of the best results, we first ranked all alignments whose absolute maximum errors are below 5 seconds by their mean absolute errors. As large maximum error is taken as lack of robustness, the worst performing settings were thus discarded. For each pair of recordings, from the remaining error sequences (from originally 312 alignments per pair, each with 2 directions of evaluation), the 10 best results, in terms of mean absolute error, were then kept for further analysis. The error values for both directions of each specific alignment were then pooled, i.e. the error values were collected and analyzed jointly. A one-way ANOVA (null hypothesis: no difference in the means) was conducted for the 10 best alignments per pair of recordings, where for all cases the null hypothesis could not be rejected (recording pair with smallest p-value: $F = 0.6, p = 0.8$). Thus, as the different settings of the 10 best alignments do not result in significant differences in terms of mean error performance, the error sequences for those 10 best alignments were collected, to estimate a distribution of the absolute errors. Fig. 5 shows the empirical cumulative distribution function of the pairwise absolute errors for all 5 alignment (performance) pairs, where each curve is obtained from the 2 error sequences (both evaluation directions) of each of the 10 best alignments for the respective performance pair.

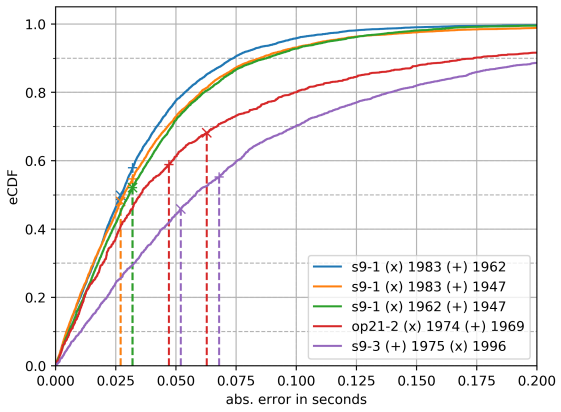


Figure 5. Cumulative distribution of absolute pairwise errors. Each curve represents pooled errors of 10 best alignments (mean absolute error) for both evaluation directions, per pair of recordings. s9-1: Beethoven, S.9, 1st Mov., s9-3: Bruckner, S.9, 3rd Mov., op21-2: Webern, Op21, 2nd Mov. (+) and (x) markers for median standard deviation of annotation, cf. Table 2.

In the following, the settings and results, in terms of mean absolute error and maximum absolute error, for the 10 best alignments are presented. For the Beethoven piece, we restricted the reporting to one pair of recordings (BPO 1962 vs. VPO 1947) due to limited space (Table 3). As can be seen from Fig. 5, the other two pairs do not differ substantially in terms of error performance, and the settings for obtaining these results are almost identical to the ones presented in the table, with an even stronger favor of the MFCC mod feature. Tables 5 and 4 show the results for the Webern and Bruckner pair of recordings, respectively.

As can be seen from the tables, best results are achieved with either MFCC or the modified MFCC. There does not seem to be a very clear pattern of which parameter setting gives best results, even within one pair of recordings. A slight advantage of medium to large FFT sizes is observed, as is a larger number of MFCC ($\geq$ 80, a number much

Feature	#MFCC	#skip	fft size	dist.	mean err.	max. err
MFCC mod	120	20	2048	l_1	38	220
MFCC mod	120	30	4096	l_2	38	411
MFCC mod	120	40	2048	l_2	39	239
MFCC	100	-	8192	l_1	40	243
MFCC	80	-	2048	l_1	40	253
MFCC mod	120	10	4096	l_1	41	318
MFCC	100	-	16384	l_1	41	318
MFCC mod	120	10	2048	l_2	41	370
MFCC mod	120	20	16384	l_1	42	285
MFCC mod	120	20	16384	cos	43	709

Table 3. Settings and results for top 10 alignments, Beethoven S9-1, BPO 1962 vs. VPO 1947. The other two cases show almost identical results (omitted for lack of space), with stronger favor of MFCC mod. Errors in ms, rounded to nearest integer.

Feature	#MFCC	#skip	fft size	dist.	mean err.	max. err
MFCC mod	120	30	2048	cos	116	4133
MFCC mod	120	20	4096	cos	123	3901
MFCC mod	120	50	2048	l_1	127	3341
MFCC mod	120	40	8192	cos	137	3597
MFCC mod	120	40	2048	cos	138	4180
MFCC mod	120	60	4096	l_2	139	2639
MFCC mod	120	10	16384	cos	144	4319
MFCC mod	120	20	2048	cos	145	4110
MFCC mod	120	20	16384	cos	150	4226
MFCC mod	120	60	16384	cos	150	4040

Table 4. Settings and results for top 10 alignments, Bruckner S9-3. Errors in ms, rounded to nearest integer.

larger than what is suggested in the literature for timbre related tasks). For the modified MFCC, skipping the first 20 to 40 out of the 120 coefficients seems a good suggestion. Interestingly, there seems to be no clear relation to the FFT size.

6.2 Relation to Human Alignment Precision

We would like to relate the accuracy achieved by automatic alignment methods to the precision with which human annotators mark score events in such recordings. This will enable us to judge the errors in the alignment methods in such a way that we cannot only say which is best, but which are probably sufficiently good for musicological studies (in relation to how precise human annotations tend to be).

Feature	#MFCC	#skip	fft size	dist.	mean err.	max. err
MFCC	80	-	2048	l_1	62	1049
MFCC	40	-	2048	l_1	65	1026
MFCC	100	-	4096	l_1	67	980
MFCC	100	-	4096	l_2	69	980
MFCC mod	120	10	4096	l_2	69	980
MFCC mod	120	20	4096	l_2	73	980
MFCC	100	-	4096	cos	76	980
MFCC	80	-	2048	cos	77	980
MFCC	100	-	8192	l_1	78	1026
MFCC mod	120	20	2048	cos	82	956

Table 5. Settings and results for top 10 alignments, Webern Op.21-2. Errors in ms, rounded to nearest integer.

By comparing the global measures of variation of the annotations (Table 2) with the mean errors obtained from the alignment study, the following can be stated. We would like the errors introduced by the alignments to be in the range of the variation introduced by human annotators. If, for example, the above estimated standard deviations are used for describing an interval (e.g. $\pm$ 1 SD) around the ground-truth annotations, then markers placed by the DTW alignment within such an interval can be taken to be as accurate as an average human annotation. However, as Tables 3 to 5 reveal, on average, the absolute errors are at least slightly (or even much in case of the Bruckner performances) larger than the estimated standard deviations, but still in a reasonable range, even for larger proportions of the score events (see Figure 5).

7. DISCUSSION AND CONCLUSIONS

Given our results, we expect the presented feature settings to be quite suitable as a first step for developing further musicological questions related to comparing multiple performances of one piece. With careful annotation of one recording, transferring the score event markers to other recordings of the same piece should yield not much worse accuracy than what is to be expected from human annotations. Detailed analyses of e.g. tempo may still need a moderate amount of manual correction, however.

An interesting application we consider is the exploration of a larger corpus of unseen recordings. Being able to establish, within a reasonable uncertainty, a common musical grid for a number of performances allows for search of (a first impression of) commonalities and differences across performances, for parameters such as tempo, or features extracted directly from the recording, such as loudness, mapped to the musical grid. This will e.g. allow the pre-selection of certain performances for more careful human annotation and further more detailed analyses. Recently, performance related data have been presented for a larger corpus in [8].

We hope to have presented some new insights with the data on annotation precision, and the applied methods for their quantification. Further work could make use of estimates of typical uncertainty of annotations to estimate, or give bounds for, the uncertainty of data derived from these. One way would be to use simple error propagation to quantify uncertainty of tempo representations, and automatically find (sections of) performances of significantly different tempo within a large corpus of recordings.

8. ACKNOWLEDGMENTS

This work was supported by the Austrian Science Fund (FWF) under project number P29840, and by the European Research Council via ERC Grant Agreement 670035, project CON ESPRESSIONE. We would like to thank the annotators for their work, as well as the anonymous reviewers for their valuable feedback. Special thanks go to

Martin Gasser for fruitful discussions of an earlier draft of this work.

9. REFERENCES

[1] Andreas Arzt, Gerhard Widmer, and Simon Dixon. Adaptive distance normalization for real-time music tracking. In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 2689–2693, 2012.

[2] Chris Cannam, Christian Landone, and Mark Sandler. Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files. In *Proceedings of the ACM Multimedia 2010 International Conference*, pages 1467–1468, 10 2010.

[3] Roger B. Dannenberg and Larry A. Wasserman. Estimating the error distribution of a single tap sequence without ground truth. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, pages 297–302, 10 2009.

[4] Simon Dixon and Gerhard Widmer. Match: A music alignment tool chest. In *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR)*, pages 492–497, 9 2005.

[5] Sebastian Ewert, Meinard Müller, and Peter Grosche. High resolution audio synchronization using chroma onset features. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1869–1872, 4 2009.

[6] Maarten Grachten, Martin Gasser, Andreas Arzt, and Gerhard Widmer. Automatic alignment of music performances with structural differences. In *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, 11 2013.

[7] Holger Kirchhoff and Alexander Lerch. Evaluation of features for audio-to-audio alignment. *Journal of New Music Research*, 40:27–41, 03 2011.

[8] Katerina Kosta, Oscar F. Bandtlow, and Elaine Chew. Mazurkabl: Score-aligned loudness, beat, and expressive markings data for 2000 chopin mazurka recordings. In *Proceedings of the International Conference on Technologies for Music Notation and Representation*, TENOR 2018, pages 85–94, 2018.

[9] Brian McFee, Colin A. Raffel, Dawen Liang, Daniel Patrick Whittlesey Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In Kathryn Huff and James Bergstra, editors, *Proceedings of the 14th Python in Science Conference*, pages 18–24, 2015.

[10] Meinard Müller, Andreas Arzt, Stefan Balke, Matthias Dorfer, and Gerhard Widmer. Cross-modal music retrieval and applications: An overview of key methodologies. *IEEE Signal Processing Magazine*, 36(1):52–62, 2019.

[11] Meinard Müller, Sebastian Ewert, and Sebastian Kreuzer. Making chroma features more robust to timbre changes. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1877–1880. IEEE, 2009.

[12] Stan Salvador and Philip Chan. Fastdtw: Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[13] Christof Weiß, Vlora Arifi-Müller, Thomas Prätzlich, Rainer Kleinertz, and Meinard Müller. Analyzing measure annotations for western classical music recordings. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 517–523, 8 2016.

MAPPING TIMING STRATEGIES IN DRUM PERFORMANCE

George Sioros **Guilherme Schmidt Câmara** **Anne Danielsen**
RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion
Department of Musicology, University of Oslo
{georgios.sioros, g.s.camara, anne.danielsen}@imv.uio.no

ABSTRACT

How do drummers express different timing styles? We conducted an experiment in which we asked twenty-two professional drummers to perform a simple rhythmic pattern while listening to a metronome. Here, we investigate the strategies they employed to express three different instructed timing profiles for the same pattern: "On", "Pushed" and Laid-back. Our analysis of the recordings follows three stages. First, we compute sixteen boolean features that capture the microtiming relations of the kick, snare and hi-hat drum onsets, between each other and with regards to the metrical grid. Second, we construct a microtiming profile (mtP) for every performance by averaging the boolean features across the recording. An mtP codifies the frequency with which the various features were found in a performance. Third, through a "similarity profiles" hierarchical clustering analysis, we identify groups of recordings with significant similarities in their mtPs. We found distinct strategies to express each intended timing profile that employ specific combinations of relations between the instruments and with regards to the meter. Finally, we created a map that summarizes the main characteristics of the strategies and their relations using a phylogenetic tree visualization.

1. INTRODUCTION

In groove performance, it has been assumed that musicians can apply different timing 'feels' to a given pattern by, amongst other things, subtly altering the temporal location of events at the 'micro-rhythmic' level by playing either slightly early ('pushed') or late ('laid-back') in relation to other players' rhythm, a metronomic beat reference or simply their own internal pulse [1, 3, 6, 7, 9, 10, 15, 19]. Typical reported values of microtiming deviations in performance range from 0 ms (no displacement) to 50 ms or more, depending on instrument, tempo and genre [2, 11, 13, 22] An instructed timing experiment by Danielsen et al. [9] showed that drummers were able to consistently play a snare-drum pattern with laid-back and pushed feel significantly behind- and ahead-of an instructed on-beat performance, respectively, with similar values. In polyphonic drumkit performance, expert drummers are able to control the degree of onset timing asynchrony between the various constituent drum instruments. These inter-instrument onset asynchronies may play a role in the production and perception of groove timing feel, since both magnitude and order of onset asynchrony between near-simultaneous events have been previously shown to affect judgements of timing in perceptual experiments with musical stimuli [12, 14, 25].

In order to explore potential interactions between instructed timing feel and various audio/motion features in drum-kit performance, a series of experiments was conducted by Câmara et al. [4] where participants played a simple 'back-beat' pattern with On-beat, Pushed and Laid-back timing feel along to a metronome. In the present study, we analyze the data from one of these experiments, limiting our focus towards investigating the extent to which professional drummers employed different strategies in order to achieve the instructed timing feel in terms of the magnitude and order of onset asynchrony between the instruments of the drum-kit themselves, as well as in relation to a metrical reference grid. We hypothesize that drummers chose different elements (read: instruments) of the rhythmic pattern to produce in sync, late and early timing performance for the On, Laid-back and Pushed timing condition, respectively. For example, in order to achieve the same timing instruction, one group of participants may have focused on the relation between two drum instruments, where one led and the other followed, while another group instead on the relation between both instruments and the metrical grid, and yet another group may have incorporated a combination of two such approaches. In other words, for the same timing instruction, drummers may produce different combinations of microtiming onset strategies in order to communicate the same intended timing feel. This article focuses on the novel analysis we developed which aims at mapping and identifying these potentially different strategies.

At the core of our analysis lie the microtiming profiles – structures that effectively codify the onset asynchronies of the instruments as the probabilities or frequencies with which they occur in the performances of the participants. A hierarchical classification of the performances based on their microtiming profiles reveals specific timing strategies, which are summarized as microtiming archetypes that capture the main characteristics of the clusters in a symbolic form. Finally, a visualization of the clustering result as a phylogenetic tree enables us to better understand and identify these strategies.

The rest of the article is divided in four sections. In section 2, we describe the experiment. In section 3, we describe the analysis method, the microtiming profiles in 3.1 and their clustering in 3.2. In section 4, we present the results of our analysis. In the final section 5, we discuss methodological issues.

2. EXPERIMENT

22 male drummers, 22-64 years of age [M = 36, SD = 11] participated in the experiment. All of them were active part-time or full-time musicians and had between 4 and 40 years of professional performance experience [M = 16, SD = 11]. All were familiar with at least one groove-based performance tradition, typically either jazz, funk/soul/R&B, hip-hop, rock, or reggae. Two participants' data were excluded from the analysis: one due to technical issues during the recording process, and the other was deemed to not have successfully understood the task based on responses from a follow up interview.

The participants were instructed to play a standard 'back-beat' pattern (see) ubiquitous in groove music and highly familiar to drummers. They performed along to a metronome (woodblock) track at a tempo deemed comfortable in a pilot of the experiment (96 b.p.m) in 3 different timing style conditions:

1. in a *laid-back* manner, i.e. behind-the-beat (condition: Laid-back)
2. in a *pushed* manner, i.e. ahead-of-the-beat (condition: Pushed)
3. in an o*n-the-beat* manner, (condition: On)

At the beginning of the experiment, a practice round was given to in order to allow for participants to accustom themselves to the following timing style conditions ('Laid-back', 'On', and 'Pushed'), which were subsequently randomized. Each timing condition trial lasted for approximately 70 seconds where participants began to play as soon as they had entrained with the timing reference track. This resulted in approximately 200 hi-hat and 50 snare and kick drum strokes captured per trial.

For our drum instrumental setup, we used the following equipment: a Gretsch acoustic metal snare drum (Gretsch Drums, CT), 7 in. deep, 14 in. wide, with a Remo Emperor X drumhead (Remo, CA) with a thin plastic muffle ring; a Gretsch 21-in. bass drum with Remo FA batter drumhead; a Pearl hi-hat stand with 14" Yamaha cymbals.

Pilot tests of the sound recordings revealed that close-microphone techniques with dynamic microphones led to too much leakage between the different drum signals, therefore AKG C411 contact microphones (AKG, Austria) were used instead and placed on the top skins of the kick and snare, and on the top cymbal of the hi-hat.

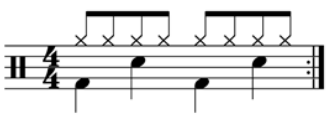


Figure 1: Standard back beat grove pattern in 4/4 meter. Upper notes in the score denote hi-hat cymbal; the middle notes, the snare drum; the bottom notes, the kick drum

3. ANALYSIS

Since the focus of this investigation is the microtiming relations of the drum instruments, we create microtiming profiles (mtPs) of the performances for all participants and instructed timing condition trials, comprised of set of features that capture those relations. Based on how similar the mtPs are, we group them using a hierarchical clustering algorithm and construct archetypes that summarize the main characteristics of each group. Finally, we map the relations between the groups' recordings using a phylogenetic tree visualization.

Our analysis and all following computations are based on the temporal location of onsets of individual strokes from each instructed timing condition recording that were calculated using an adaptation of an existing onset detection algorithm of the MIRtoolbox [17] which will be detailed in a forthcoming publication of our group.

We describe the mtPs and the measurements we use to obtain them in subsection 3.1, then present the clustering results and their visualization in subsection 3.2.

3.1 Microtiming Profiles

To obtain the mtP of a recording we first extract a set of sixteen boolean features that capture the microtiming relations between the strokes of the snare, kick drum and hi-hat cymbals relative to each other, as well as to the location of the metrical grid. The boolean features are calculated for each 4/4 measure of a timing condition trial while the mtP is calculated as the average of the boolean features across all measures of a trial. The mtPs were inspired by the motion templates designed by Müller and Röder [20] to describe geometric relations of the human body for the purpose of the analysis of body movements.

Kick on beat 1	x2 features per instrument late/early relative to the hi-hat cymbal
Snare on beat 2	
Kick on beat 3	
Snare on beat 4	
Kick + Hi-hat on beat 1	x2 features per beat both instruments late/early relative to the metrical grid
Snare + Hi-hat on beat 2	
Kick + Hi-hat on beat 3	
Snare + Hi-hat on beat 4	

Table 1: Summary of the sixteen boolean features extracted from the recordings for each bar.

Each feature tests whether an instrument is late or early with respect to a certain reference time point. The first eight features use the onsets of the hi-hat strokes as a reference and the other eight features use the metrical grid as a reference. For instance, feature 1 tests whether the kick drum follows the corresponding hi-hat cymbal,

while feature 9 tests whether both the kick and hi-hat occur after the position of the respective beat. Table 1 summarizes the sixteen binary features extracted from the recordings of each trial.

The above features depend on "tolerance" thresholds with which two instrument's strokes are considered synchronous with each other (for features 1 to 8) or with which the instruments' strokes are considered to occur late or early relative to the beat of the metrical grid (for features 9 to 16). For instance, when the inter-onset interval (IOI) between a pair of kick and hi-hat strokes is greater than the respective synchronicity threshold, feature 1 (kick later than hi-hat) is *true*. In the opposite case where IOI is sub-threshold, both features 1 and 5 (kick later and earlier than hi-hat) are *false*, since the pair is considered to be synchronous.

Furthermore, to determine the relation between the three drum strokes and the metrical grid with which drummers used as a beat reference, we need first to determine the location of this grid. Although one might intuitively assume that the onset of the sounding metronome would correspond to that location, it has been repeatedly observed that people tend to tap to a steady pulse systematically earlier than the actual pulse—a phenomenon known as negative mean asynchrony (NMA) [8]. As such, it may be assumed that the internal pulse scheme with which drummers operate with, that is, their subjective metrical grid, is slightly anticipated.

Even though percussionists and drummers tend to display lower NMA than other musicians in both in-phase synchronous tapping [21] and drumming [11] experiments, NMAs still tends to vary significantly between individuals [8]. Therefore, it is difficult to assume a single global NMA value for all the drummers. Similarly, the two thresholds values described above cannot easily be set universally. In what follows, we will describe how we obtain individual values for these parameters for each drummer based on the performance of their On timing condition, essentially turning the On recordings into a baseline reference. We will discuss the reasoning behind this choice as well as some of its implications in section 5.

For the synchronicity threshold values used in features 1-8, i.e. the tolerance with which two coinciding drum strokes are considered synchronous or not, we use the variability of the IOI between the hi-hat and coinciding kick or snare strokes on each of the four main quarter-note beats of the 4/4 metre (kick + hi-hat on beats 1 and 3, snare + hi-hat on beats 2 and 4). For each drummer, we first calculate the standard deviation of the inter-instrument IOIs of the corresponding beats in each measure of their respective On condition recording. This yields four separate values, two for each hi-hat + kick, and hi-hat + snare, feature. To be conservative, the synchronicity threshold for a drummer is chosen as the maximum of these four values, then multiplied by 2. Thus, a kick or snare stroke is considered to occur as either asynchronously late or early relative to its coinciding hi-hat stroke when their onsets satisfy the following inequalities:

Late: $Onset(i,j) - HiHat(i,j) > SyncThr(i)$

Early: $Onset(i,j) - HiHat(i,j) < -SyncThr(i)$

$SyncThr(i) = 2 \times \max_{k=1}^{4}\{STD_{ON}^{i}(IOI\ at\ beat\ k\}$

where *j* is the beat number from a recording of drummer *i*.

The synchronicity threshold used in features 9 – 16 on the other hand, i.e. the tolerance with which a stroke is considered to occur late or early relative to a corresponding subjective metrical beat, is based on the timing variability of the hi-hat strokes in the On condition. The hi-hat is chosen because it can be considered more of a 'timekeeper' instrument than the other drums, thus serving more aptly as a proxy for the beats of the drummers' subjective metrical reference. To account for the commonly observed anticipation of the beats of the metronome (henceforth abbreviated as AoB), we first calculate the mean position of the hi-hat strokes relative to the metronome in the On recordings for each drummer. 18 out of 20 drummers displayed NMA of hi-hat strokes relative to the actual metronome, consequently yielding negative AoB values. Two participants displayed either no NMA or minutely positive mean asynchrony, and in these cases the AoB was set to 0 (no anticipation). Finally, we consider any drum stroke as occurring asynchronously late or early in relation to the beats of the metrical grid according to the following inequalities:

Late: $Onset(i,j) - beat(j) - AoB(i) > 2 \times STD_{ON}^{i}(HiHat)$

Early: $Onset(i,j) - beat(j) - AoB(i) < -2 \times STD_{ON}^{i}(HiHat)$

where *j* is a drum stroke of drummer *i*, $beat(j)$ is the corresponding position of the metronome, and $STD_{ON}^{i}(HiHat)$ is the standard deviation of the IOI of the hi-hat from the respective metrical beat positions in the On recording of the same drummer. For one of the features 9-16 to be *true*, the onsets of both strokes—hi-hat and kick or snare—must be either early or late. In all other cases, including when one stroke is early and the other late, the corresponding features would be *false*.

Two main observations must be made about the boolean features. First, all boolean features form mutually exclusive pairs. For instance, feature 9 (strokes on beat 1 are early) and feature 13 (strokes on beat 1 are late) cannot be both *true* for the same bar of a recording. However, they can both be *false*, in which case the combination of the two strokes is considered on the beat. Second, features 1-8 (kick and snare relative to hi-hat) and features 9-16 (onsets relative to metrical grid) are independent. For instance, a kick onset can be early relative to the respective hi-hat onset (feature 5 *true*) while at the same time they are both late relative to the beat position (feature 9 *true*).

The final microtiming profiles (mtPs) of the recordings are calculated by averaging the boolean features across each timing condition recording for all drummers. As the boolean features take either *true* (1) or *false* (0) values, averaging them results in values in the range [0, 1], which represent the frequency with which a feature was

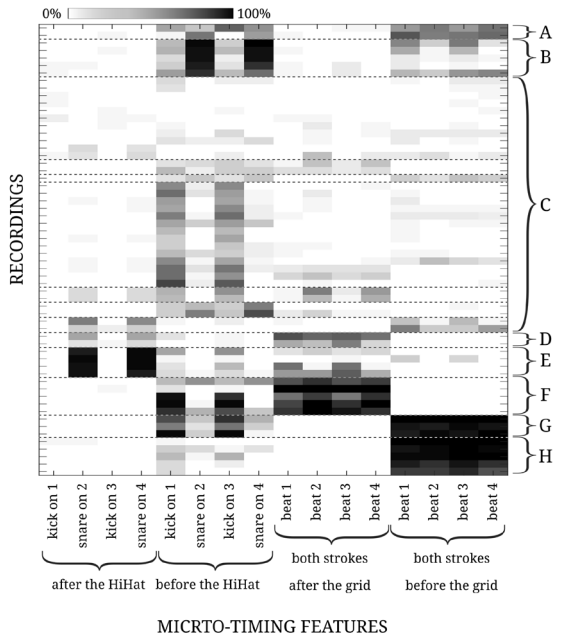


Figure 2: Microtiming profiles (mtPs) for all recordings shown as a greyscale image representing the probability or frequency with which a feature is encountered in a recording. Features are laid along the horizontal axis. On the vertical axis, recordings are sorted and grouped based on the proximity of the SIMPROF clusters (see section 3.2 and Figure 3). Horizontal dashed lines represent the cluster boundaries. The corresponding group mt archetypes are marked on the right with the letters A-H.

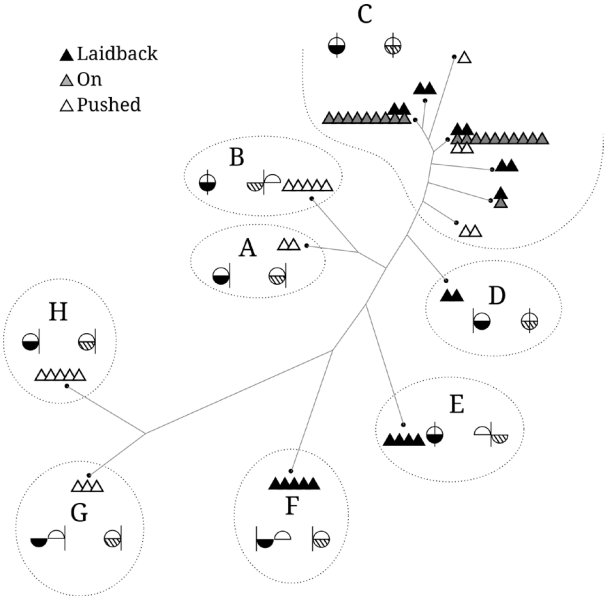


Figure 3: Hierarchical clustering presented as a phylogenetic tree (unrooted, equal daylight visualization). Each triangle corresponds to the microtiming profile of a single recording. Letters A-H are used to label the clusters. Next to each cluster the corresponding mt-Archetype is shown. Label C is assigned to a group of proximal clusters which correspond to the same mt archetype.

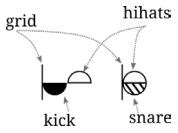


Figure 4: Explanation of an mt-Archetype symbol.

encountered in a recording. The mtPs of all the recordings are visualized in matrix form in Figure 2.

3.2 Hierarchical clustering

We sought to identify the extent to which drummers implemented distinct microtiming strategies for different timing conditions and whether they formed different groups. To this end, we used an agglomerative, hierarchical cluster analysis of the mtPs. A hierarchical clustering was preferred to other clustering methods, like k-means, since it is flexible in that it does not require an a-priori number of clusters to be determined, nor does it impose restrictions on the distribution of the data. Its only requirement is a similarity metric between the data points. We treated the mtPs as arrays of variables, where the similarity between two mtPs is the Euclidian distance between them.

Hierarchical agglomerative algorithms result in dendrograms by successively joining neighboring data points or groups of previously joined points. A linkage criterion determines the distance between groups of points as a function of the pairwise distances of the points themselves. In this study, we used the common Unweighted Group Average linkage (UPGMA) [18, p. 352].

To create clusters of similar data points, one generally "cuts" the dendrogram at different heights. Here, we borrowed methods from the fields of bioinformatics andecology to identify clusters of mtPs. We used the similarity profiles (SIMPROF) method [5]—as implemented in the Fathom Toolbox for Matlab [16]—to test the statistical significance of the branches' internal structure. SIMPROF takes the form of a series of permutation tests. Beginning at the top of a precalculated hierarchy, thesetests stop the ever finer partitioning into subgroups. When a branch in the hierarchy is deemed to have no internal structure and is therefore homogenous, it is no longer subdivided. Thus, a cluster is formed that is comprised of recordings with "exchangeable" features.

For the permutation test, we set the number of iterations to 1000 and the significance level alpha to 0.05—the probability value at which the hypothesis of an internal structure is rejected. We used the Bonferroni correction [18, p. 745] to progressively adjust the probability values for multiple simultaneous tests (see also parameter mc=*true* of the `f_disprof_clust` function of the Fathom toolbox [16]).

The results of the clustering analysis are shown in Figure 2. The dashed horizontal lines cut the mtP matrix into clusters of recordings that show statically significant similarities. In Figure 3, we present the same result as an unrooted phylogenic tree. Recordings of the various Laidback, On and Pushed performances are represented as

black, grey or white triangles, respectively, while the distance between the clusters of recordings corresponds to the UPGMA linkage criterion [18, p. 352].

As our aim is to identify distinct timing strategies in the recordings, we summarize the main characteristics of each cluster into microtiming archetypes. An archetype is computed by first averaging the probabilities of the features that belong in beats 1 and 3 (where the kick strokes occur) as well as the ones that belong in beats 2 and 4 (where the snare strokes occur). The two groups of probabilities describe the relation of the kick/hi-hat and snare/hi-hat combinations of strokes between them and in relation to the meter. Those relationships are reduced into archetypes according to whether the average probability of each feature is above or below 50%. An example of such an archetype is shown in Figure 4. In this example, the average probability of both a kick and corresponding hi-hat stroke to be late relative to the beat is above 50%. At the same time, the average probability of a kick stroke being ahead of the corresponding hi-hat stroke is also above 50%. In contrast, the snare strokes have a probability below 50% to either occur ahead or after the corresponding hi-hat stroke. In other words, most of the snare strokes are considered as synchronous with their corresponding hi-hat strokes.

Clusters with common microtiming archetypes that are relatively proximal in the phylogenetic tree are grouped together to create an overall map of strategies. In Figure 2, these groups are labeled with the letters A to H and in Figure 3 the same groups are annotated with their corresponding archetype.

4. RESULTS

The classification of the mtPs shows that the majority of the Laid-back and Pushed performances form separate homogenous clusters. The purely Laid-back clusters are characterized by generally late timing while the Pushed ones by early timing, as expected. However, the analysis reveals that drummers implemented distinct strategies to express a given timing feel by focusing on different rhythmic elements. The microtiming archetypes assigned to the various group clusters highlight those elements.

More specifically, there are 5 purely Laid-back clusters which comprise 15 out of the 20 Laid-back performances. However, 4 of those recordings (split into 2 clusters in group C) are proximal to the two On clusters and are thus subsumed by the very same archetype. The other 11 are split in three clusters each forming a separate group (D, E and F). All three groups are characterized by late strokes. On the one hand, in group E, the snare stroke is late in relation to the hi-hat and in F, both hi-hat and snare are played late relative to the metrical beat. On the other hand, in group F, the combined kick/hi-hat strokes are late relative to the beat, while the kick additionally precedes the hi-hat.

The Pushed performances are mainly found in purely Pushed clusters (18 out of the 20). However, in similar fashion to the Laid-back condition, a small portion (3) are found proximal to the On clusters in group C. 15 of the remaining Pushed performances form 4 distinct groups (A, B, G and H). All of them are characterized by the early timing of the snare strokes. In group B, the snare precedes the hi-hat although the snare/hi-hat combination is considered as synchronous with the beat. Group B exhibits the inverse pattern of the Laid-back group E (snare early in relation to hi-hat). Similarly, the Pushed group G has its counterpart in the Laid-back group F, with both instruments being early in relation to the beat instead of late but the kick still precedes the hi-hat.

In groups A and H the rhythmic pattern appear to be simply shifted early in relation to the beat. However, although the two groups correspond to the same archetype, they are relatively distant on the tree. A closer look at their mtPs in Figure 2 reveals that in group A, the strokes are anticipating the beat significantly less often than in group H. This can also be seen in the proximity of the group A to group C which is dominated by the On recordings. The similar proximity of the Laid-back group D to the On group C reflects the analogous weak late timing features of the mtPs in comparison with the other Laid-back groups (E, F).

The On performances are all found in Group C which is characterized by synchronous on-the-beat stroke onsets. Within the group, the On recordings are split into two clusters. Examining their mtPs, we see their difference stem from the tendency of some musicians to play the kick drum ahead of the hi-hat.

5. DISCUSSION

In this study, we present findings of an experiment in which professional drummers performed the same rhythmic pattern with an On, Laid-back and Pushed timing feel. We found that participants used more than one distinct onset microtiming pattern for each intended timing instruction (see section 4). A more in-depth discussion of the timing strategies and their musicological implications will be undertaken in an upcoming publication. In the present discussion, we will focus on methodological issues concerning, first, the various parameters of the analysis and their implications, and second, the interpretation of the clustering results.

Our analysis begins with the encoding of the onset asynchronies found in the performances of the drummers into sets of boolean features. Microtiming profiles of the performances are calculated then by averaging those features over the respective recordings. The mtPs codify the probability or frequency with which each feature was encountered in a recording. Finally, by clustering the mtPs we discover the recordings with similar features and group them together.

The boolean features, however simple in their definitions, depend on parameters and thresholds which are crucial to the outcome of the analysis and are closely related to the research questions. In this study, we chose to use individual values for each drummer instead of setting

global ones across all participants. This decision partly reflects the fact that phenomena such as negative mean asynchrony (NMA) typically varies between individuals. and, at the same time, naturally follows from our research question that seeks to identify individual strategies that musicians employ. The three parameters used for calculating the boolean features reflect mechanisms relevant to the perception and production of the subtle asynchronies we are studying. Therefore, setting individual values in our analysis corresponds to adopting the 'point of view' of each separate musician independently, whereas global values might instead correspond better to the perception of a 'typical' listener.

The way in which individual values are assigned to each parameter can significantly impact results depending on how the research question is formalized. In our current approach, we chose to derive the individual parameters for each musician based on their respective On performances, essentially rendering the On condition into a baseline from which the other timing conditions were compared against. For instance, whether a stroke is considered as 'on-the-beat' or not depends on whether it was performed late or early relative to the average hi-hat stroke in the On condition. In this case, the research question could perhaps be interpreted instead as "how do musicians differentiate their Laid-back or Pushed from an On timing feel".

Consequently, one might assume that the mtPs of the On recordings contain no meaningful information since, after all, they cannot be different from themselves! Nevertheless, the On performances should not be excluded from the analysis: firstly, their mtPs can still exhibit significant enough differences between the performances to classify them separately (see group C), though those differences can solely be obtained from the onset relations between the instruments themselves, and not with respect to the metrical grid. Secondly, the proximity of the Laid-back and Pushed clusters to the two On clusters in the phylogenetic visualization of Figure 3 is informative insomuch as it is telling of the strong tendency for drummers to differentiate these asynchronous timing feels from the On timing feel.

The further hierarchical clustering of the derived mtPs proved an effective means of identifying several key timing strategies implemented by the drummers. The method groups and sorts the recordings according to their similarity, revealing their relations without the need for a-priori hypotheses about the existence of specific strategies. Although in principle it is possible to analyze the data using more conventional multivariate statistical approaches, it would be difficult to formalize the hypotheses to be tested, especially considering the variety of strategies that the musicians seem to exhibit. However, in future studies, the two approaches could eventually complement each other: hierarchical clustering can assist in formalizing concrete hypotheses while conventional analyses provide more robust statistical results.

The similarity profiles method (SIMPROF) permits the clustering of statistically similar performances together. It should be noted that other techniques such as bootstrapping [23, 24] may be used as statistical means to define the boundaries of clusters in the mtPs matrix (Figure 2). We leave the exploration of these alternative techniques for a forthcoming publication.

An important parameter in SIMPROF is the significance level (alpha) with which the null hypothesis (that the differences in the mtPs inside a cluster are the result of random combination of the various features) is rejected. The value of alpha, together with the Bonferroni correction for multiple simultaneous tests, determines the level of detail of the final classification, or in other words, the size and scope of the clusters. For instance, if we do not adjust the p-values for multiple tests (Bonferroni correction parameter set to *false* in the `f_disprof_clust` function of the Fathom toolbox [16]), groups B and F are split into three and two subclusters respectively. Looking at the mtPs in Figure 2, we see that for group B this is due to the subtle tendency of some performances to play ahead of the beat. In group F, it is due to the relation of the kick and hi-hat strokes. In the more common approach to hierarchical clustering, in which dendrograms are cut horizontally, this level is controlled by the height that a dendrogram is cut.

The way mtPs are clustered together plays central role in the creation of archetypes and therefore in the characterization of the various timing strategies. Archetypes are calculated as averages of the mtPs in each cluster which are further reduced into eight boolean values. For example, if group F was to be split into two sub-clusters, they would not correspond to the same archetype but would form distinct groups. In contrast, the sub-clusters of group B discussed above correspond to the same archetypes.

Exploring and understanding the results of the clustering analysis requires a side-by-side examination of the mtP matrix (Figure 2) and the phylogenetic tree (Figure 3). The two visualizations combined offer an overview of the performances allowing for a closer examination of the finer timing relation details between the different strategies. This simultaneous multilevel view of the recordings enables us to draw conclusions about the timing strategies which would otherwise be obfuscated or oversimplified.

In conclusion, the encoding of the drum performances into boolean features and the hierarchical classification of the derived microtiming profiles effectively cluster the performances into meaningful groups. The further phylogenetic visualization and the symbolic representation of the groups through microtiming archetypes is an efficient way of mapping drummers' main timing strategies, providing an easily interpretable overview of the results. Our analysis simultaneously brings to the surface higher level rhythmic aspects common to the performances as well as the finer details that differentiate them without the one occluding the other.

6. ACKNOWLEDGMENTS

This work was partially supported by the Research Council of Norway through its Centres of Excellence scheme, project number 262762.

7. REFERENCES

[1] M. W. Butterfield, "The Power of Anacrusis: Engendered Feeling in Groove-Based Musics," *Music Theory Online*, vol. 12, no. 4, pp. 1–17, 2006.

[2] G. S. Câmara, "Swing in Early Funk and Jazz-Funk (1967–1971): Micro-Rhythmic and Macro-Structural Investigations," University of Oslo, 2016.

[3] G. S. Câmara and A. Danielsen, *Groove*, vol. 1. Oxford University Press, 2018.

[4] G. S. Câmara and A. Danielsen, "Forthcoming."

[5] K. R. Clarke, P. J. Somerfield, and R. N. Gorley, "Testing of null hypotheses in exploratory community analyses: similarity profiles and biota-environment linkage," *J. Exp. Mar. Bio. Ecol.*, vol. 366, no. 1–2, pp. 56–69, 2008.

[6] A. Danielsen, *Presence and Pleasure: The Funk Grooves of James Brown and Parliament*. Middletown, CT: Wesleyan University Press, 2006.

[7] A. Danielsen, "Pulse as Dynamic Attending. Analysing Beat Bin Metre in Neo Soul Grooves," in *The Routledge Companion to Popular Music Analysis: Expanding Approaches*, C. Scotto, K. M. Smith, and J. Brackett, Eds. London: Routledge, 2018, pp. 179–189.

[8] A. Danielsen *et al.*, "Where is the beat in that note? Effects of attack, duration, and frequency on the perceived timing of musical and quasi-musical sounds," *J. Exp. Psychol. Hum. Percept. Perform.*, vol. 45, no. 3, pp. 402–418, 2019.

[9] A. Danielsen, C. H. Waadeland, H. G. Sundt, and M. A. G. Witek, "Effects of instructed timing and tempo on snare drum sound in drum kit performance," *J. Acoust. Soc. Am.*, vol. 138, no. 4, pp. 2301–2316, 2015.

[10] A. Friberg and A. Sundström, "Swing Ratios and Ensemble Timing in Jazz Performance: Evidence for a Common Rhythmic Pattern," *Music Percept.*, vol. 19, no. 3, pp. 333–349, Mar. 2002.

[11] S. Fujii, M. Hirashima, K. Kudo, T. OhtsuKi, Y. NaKamura, and S. Oda, "Synchronization error of drum kit playing with a metronome at different tempi by professional drummers," *Music Percept.*, vol. 28, no. 1, pp. 491–503, 2011.

[12] W. Goebl and R. Parncutt, "The influence of relative intensity on the perception of onset asynchronies," in *ICMPC7, July 2002*, 2002, no. July, pp. 1–4.

[13] K. Hellmer and G. Madison, "Quantifying Microtiming Patterning and Variability in Drum Kit Recordings: A Method and Some Data," *Music Percept. An Interdiscip. J.*, vol. 33, no. 2, pp. 147–162, Dec. 2015.

[14] M. J. Hove, C. Marie, I. C. Bruce, and L. J. Trainor, "Superior time perception for lower musical pitch explains why bass-ranged instruments lay down musical rhythms," in *Proceedings of the National Academy of Sciences*, 2014, vol. 111, no. 28, pp. 10383–10388.

[15] V. Iyer, "Embodied Mind, Situated Cognition, and Expressive Microtiming in African-American Music," *Music Percept.*, vol. 19, no. 3, pp. 387–414, Mar. 2002.

[16] D. L. Jones, "Fathom Toolbox for MATLAB: software for multivariate ecological and oceanographic data analysis." College of Marine Science, University of South Florida, St. Petersburg, FL, USA., 2017.

[17] O. Lartillot, P. Toiviainen, and T. Eerola, "A Matlab Toolbox for Music Information Retrieval," in *Data Analysis, Machine Learning and Applications. Studies in Classification, Data Analysis, and Knowledge Organization.*, C. Preisach, H. Burkhardt, L. Schmidt-Thieme, and R. Decker, Eds. 2008, pp. 261–268.

[18] P. Legendre and L. Legendre, *Numerical Ecology*, 3rd ed., vol. 24. Elsevier B.V., 2012.

[19] G. Madison, "Experiencing Groove Induced by Music: Consistency and Phenomenology," *Music Percept.*, vol. 24, no. 2, pp. 201–208, Dec. 2006.

[20] M. Müller and T. Röder, "Motion Templates for Automatic Classification and Retrieval of Motion Capture Data," in *SCA '06 Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2006, pp. 137–146.

[21] B. H. Repp, "Sensorimotor synchronization: A review of the tapping literature," *Psychon. Bull. Rev.*, vol. 12, no. 6, pp. 969–992, Dec. 2005.

[22] O. Senn, L. Kilchenmann, R. Von Georgi, and C. Bullerjahn, "The effect of expert performance microtiming on listeners' experience of groove in swing or funk music," *Front. Psychol.*, vol. 7:1487, no. September, 2016.

[23] R. Suzuki and H. Shimodaira, "Pvclust: An R package for assessing the uncertainty in hierarchical clustering," *Bioinformatics*, vol. 22, no. 12, pp. 1540–1542, 2006.

[24] R. Suzuki and H. Shimodaira, "An application of multiscale bootstrap resampling to hierarchical clustering of microarray data: How accurate are

these clusters," *15th Annu. Int. Conf. Genome Informatics, Posters Softw. Demonstr.*, 2004.

[25] M. Wojtczak, A. H. Mehta, and A. J. Oxenham, "Rhythm judgments reveal a frequency asymmetry in the perception and neural coding of sound synchrony," *Proc. Natl. Acad. Sci.*, vol. 114, no. 5, pp. 1201–1206, Jan. 2017.

IMPROVING SINGING AID SYSTEM FOR LARYNGECTOMEES WITH STATISTICAL VOICE CONVERSION AND VAE-SPACE

Li Li[1], Tomoki Toda[2], Kazuho Morikawa[2], Kazuhiro Kobayashi[2], Shoji Makino[1]
[1] University of Tsukuba, Japan [2] Nagoya University, Japan
lili@mmlab.cs.tsukuba.ac.jp, tomoki@icts.nagoya-u.ac.jp

ABSTRACT

This paper proposes an improved singing aid system for laryngectomees that converts electrolaryngeal (EL) speech produced using an electrolarynx to a more naturally sounding singing voice. Although the previously proposed system employing a noise suppression process and a rule-based pitch control approach has achieved preliminary success in converting EL speech into a singing voice, there are still two major limitations. First, the converted singing voice still sounds mechanical and unnatural owing to the adverse impacts of spectrograms extracted from EL speeches, also making the effect of pitch control limited. Second, the capability and flexibility of the rule-based pitch control in modeling various singing styles are insufficient, causing the converted singing voices to lack variety. To address these limitations, this paper proposes an improved system that uses 1) a statistical voice conversion approach to convert spectrograms extracted from EL speeches into those of natural speeches and 2) a deep generative model-based approach called VAE-SPACE for pitch modification, which generates pitch patterns in a data-driven manner instead of following manually designed rules. The experimental results revealed that 1) the conversion of spectrograms was effective in improving the naturalness of singing voices, and 2) the statistical pitch control approach was able to achieve comparable results with the rule-based approach, which was very carefully designed to be specialized in singing.

1. INTRODUCTION

The voice is an essential tool used by most of people to communicate with others or express themselves. However, it is difficult for laryngectomees whose larynxes have been removed in surgery to speak or sing in a common manner since they are unable to generate glottal excitation sounds owing to the loss of their vocal folds. In consequence, this vocal disorder may significantly degrade the quality of life of laryngectomees. One popular approach that enables laryngectomees to speak again is to use an external medical device called *electrolarynx* to produce intense mechanical vibrations as an alternative to glottal excitation sounds. Electrolaryngeal (EL) speech produced using an electrolarynx is noteworthy for its intelligibility, and furthermore, it is easy for laryngectomees to learn how to use an electrolarynx, even for people with low physical fitness. However, the perceived naturalness of EL speech is unsatisfactory owing to the use of mechanically generated source excitation sounds, the fundamental frequency (F_0) contours of which are usually flat or given as predetermined patterns. This further limits the capability of the electrolarynx to assist laryngectomees in singing, where F_0 contours play an important role in providing both melodic information and details related to the naturalness and singing style [1].

To develop singing aid systems for laryngectomees, it is essential to suitably control the pitch of EL speech, i.e., F_0 contours. One existing approach is to set F_0 contours corresponding to melodies of predetermined songs and embed them in advance into the electrolarynx as a function to allowing these songs to be sung. However, the flexibility in singing with this approach is unsatisfactory because the number of embedded songs is limited and laryngectomees are solely allowed to sing in predetermined styles.

To achieve a more flexible singing aid, a system based on pitch control has recently been proposed [2]. With this system, laryngectomees are allowed to freely control melodic information such as musical scores and tempo by playing a musical instrument themselves while singing with an electrolarynx. Singing voices are then generated by applying a voice conversion approach that converts EL speeches into singing voices containing well-sung F_0 contours that are modified from the inputted musical scores according to a set of manually predefined rules [3, 4]. Furthermore, noise suppression [5] is employed to reduce the source excitation signals emitted from the electrolarynx. Although this system has achieved preliminary success as a singing aid, there are still two limitations. First, the effect of pitch control in improving the naturalness of singing voices was limited because of the fluctuations originating from the spectral features extracted from EL speeches [2], which resulted in the converted singing voices still sounding mechanical and unnatural. Second, both the capability and flexibility of the rule-based pitch control approach in modeling various singing styles are insufficient. Once the rules are determined, the system outputs certain F_0 pat-

terns containing similar characteristics without considering the personalities and emotions of singers, which is an undesirable property of singing aid systems.

To develop a system that is capable of aiding laryngectomees to sing naturally and distinctly, this paper proposes an improved system that uses 1) a statistical voice conversion (VC) approach based on the Gaussian mixture model (GMM) [6, 7] to convert spectral features extracted from EL speeches into those of natural speeches to alleviate the fluctuation problem and 2) a deep generative model-based approach called VAE-SPACE [8] to generate F_0 contours of singing voices from input musical scores. As a data-driven approach, it is expected that VAE-SPACE can learn the rules for generating natural F_0 contours from data automatically, making it possible to model different singing styles and expressions with a unified model.

2. OVERALL FRAMEWORK OF SINGING AID SYSTEM FOR LARYNGECTOMEES

Fig. 1 shows an overview of the conventional singing aid system proposed in [2] that takes a sequence of musical scores $\boldsymbol{N} = [n_1, \ldots, n_t, \ldots, n_T]$ provided by playing an instrument as melodic information in addition to an EL speech $\boldsymbol{S} = [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_t, \ldots, \boldsymbol{s}_T]$. Here, $\boldsymbol{s}_t = [s_t(1), \ldots, s_t(f), \ldots, s_t(F)]^\mathsf{T}$ denotes the short-time Fourier transform (STFT) coefficients of the EL speech at frame t, and f and $(\cdot)^\mathsf{T}$ denote the frequency index and transpose operator, respectively. The system mainly consists of three modules, namely, for voice quality enhancement, pitch control, and synthesis. With this system, singing voices are generated by a vocoder-based synthesis approach [9] that takes phonetic information and pitch information as inputs, where the former is extracted from the EL speech $\boldsymbol{S}$ enhanced by the voice quality enhancement module and the latter is obtained by modifying the input musical scores $\boldsymbol{N}$ via the pitch control module.

Note that this system can also serve laryngectomees who do not play instruments by allowing them to sing with played accompaniments, where a sequence of predetermined musical scores is given in synchronization with the accompaniments. Different from the method of embedding preset F_0 patterns into an electrolarynx and controlling the pitch by pushing a button, this system can obtain more natural singing voices since singing voices obtained with the former method are usually interrupted when the musical score changes owing to the limitation of mechanical excitation generation, and those obtained in the latter way are converted from more fluent EL speeches.

3. CONVENTIONAL SYSTEM WITH NOISE SUPPRESSION AND RULE-BASED PITCH CONTROL

3.1 Noise suppression

It is important to enhance the quality of both phonetic information and pitch information to achieve a better transformation. To obtain correct phonetic information from

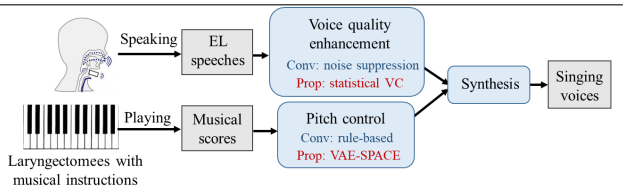


Figure 1. Flowchart of singing aid system.

an EL speech, which is usually mixed with a noisy source signal that radiates from the position of the EL attachment, the aforementioned system employs a spectral subtraction (SS) method [5] to enhance the voice quality of the recorded EL speech. A prototype noise amplitude spectrum $|l(f)|$ is calculated by averaging the amplitude spectra of the EL noise recorded with a close-talking microphone in advance. The enhanced EL speech is obtained with the enhanced amplitude spectrum $|\hat{s}_t(f)|$ and the noisy phase spectrum, where

$$|\hat{s}_t(f)| = \begin{cases} |s_t(f)| - 2|l(f)|, & (|s_t(f)| > 2|l(f)|), \\ 0, & (\text{otherwise}). \end{cases} \quad (1)$$

The phonetic information used for synthesis, i.e., spectral features and aperiodic components, is obtained by analyzing the enhanced EL speech with fixing F_0 at a constant value and using the on/off information of the electrolarynx as unvoiced/voiced information.

3.2 Rule-based pitch control

For pitch control, a rule-based F_0 modification technique [3, 4] is applied to add overshoot, vibrato, preparation, and fine fluctuation, which are four characteristics typically observed in F_0 contours of natural singing voices, into the musical scores n_t. Specifically, overshoot, vibrato, and preparation are added by applying the following infinite impulse response filter to the musical scores:

$$H(s) = \frac{k}{s^2 + 2\zeta\omega s + \omega^2}, \quad (2)$$

where ω, ζ, and k denote the natural frequency, damping coefficient, and proportional gain, respectively. Overshoot and preparation are expressed as the second-order damping model ($0 < |\zeta| < 1$), while vibrato is expressed as the second-order oscillation model ($|\zeta| = 0$). The fine fluctuation is generated from white noise that is high-pass-filtered with the cutoff frequency set at 10 Hz followed by a normalization. The modified F_0 contour can be expressed as $o_t = n_t + e_t$, where o_t and e_t denote the generated F_0 and the component including all four characteristics that is finally added to the musical score at frame t, respectively.

3.3 Limitations

The effectiveness of this system in converting EL speech into a singing voice was experimentally confirmed in [2]. However, it was also reported that undesirable fluctuations reside in F_0 contours of synthesized singing voices that

may have originated from the spectral features extracted from the enhanced EL speeches, which cause the singing voices to still sound mechanical and unnatural. The upper figure in Fig. 6 shows an example of the reanalyzed F_0 contour of a singing voice obtained with the system. Another limitation originates from the rule-based pitch control. Although the system allows laryngectomees to sing an arbitrary song with the desired melody, it is difficult for this framework to further improve the capability to express various singing styles or to generate expressive singing voices.

4. PROPOSED SYSTEM WITH STATISTICAL VC AND VAE-SPACE

To remove these indefinite spectral components affecting F_0 contours, one of the promising approaches is to transform the spectral features of EL speeches of songs into those of natural singing voices not containing these components. Statistical VC techniques [6, 7] have the potential to be used for developing such a transformation based on training data consisting of utterance pairs of the source and target voices, namely, singing voices sung using an electrolarynx (EL speeches) and in a natural way. Furthermore, it is expected that EL noise can be reduced together by training a model with the source voice being noisy EL speech.

To address the second limitation, motivated by the high flexibility of a statistical approach in modeling different voice characteristics and singing styles [10–12], we propose using a statistical parametric model for pitch control instead of the rule-based approach. Specifically, we employ VAE-SPACE [8], which uses a variational autoencoder (VAE) as an analysis-synthesis model to discover the structure of an F_0 generating process for the singing voice in a data-driven manner as well as an inverse process for estimating the underlying musical scores.

4.1 Statistical VC for converting EL speech into singing voice

Let $\boldsymbol{x}_t = [x_t(1), \ldots, x_t(d), \ldots, x_t(D)]^\mathsf{T}$ denotes the D-dimensional spectral feature extracted from EL speech $\boldsymbol{s}_t$ at frame t, where d denotes the index of the feature dimension. The aim of VC is to estimate the spectral features, F_0 contours including unvoiced/voiced (U/V) information, and aperiodic components of the corresponding natural singing voice, which are denoted by the same variable $\boldsymbol{y}_t = [y_t(1), \ldots, y_t(d), \ldots, y_t(D)]^\mathsf{T}$ for simplicity, from the noisy spectral sequence $\boldsymbol{x}_t$.

In the training step, a joint probability density function (p.d.f.) of the joint acoustic feature vectors $[\boldsymbol{X}_t^\mathsf{T}, \boldsymbol{Y}_t^\mathsf{T}]^\mathsf{T}$ is modeled with a GMM as follows:

$$P\big(\boldsymbol{X}_t, \boldsymbol{Y}_t \big| \boldsymbol{\Theta}^{(\mathrm{X,Y})}\big) = \sum_{m=1}^{M} \alpha_m \mathcal{N}\big([\boldsymbol{X}_t^\mathsf{T}, \boldsymbol{Y}_t^\mathsf{T}]^\mathsf{T}; \boldsymbol{\mu}_m^{(\mathrm{X,Y})}, \boldsymbol{\Sigma}_m^{(\mathrm{X,Y})}\big). \quad (3)$$

Here $\boldsymbol{X}_t$ denotes spectral segment feature vectors that are obtained by performing principal component analysis (PCA) for the joint vectors concatenating the spectral feature vectors of the current frame, preceding L frames, and succedding L frames extracted from source voices. $\boldsymbol{Y}_t = [\boldsymbol{y}_t^\mathsf{T}, \Delta\boldsymbol{y}_t^\mathsf{T}]^\mathsf{T}$ denotes vectors combining static and dynamic features extracted from target voices. $\boldsymbol{\Theta}^{(\mathrm{X,Y})}$ denotes a parameter set of the GMM, which consists of the weights α_m, mean vectors $\boldsymbol{\mu}_m^{(\mathrm{X,Y})}$, and convariance matrices $\boldsymbol{\Sigma}_m^{(\mathrm{X,Y})}$ of all the mixture components. Moreover, the p.d.f. of the global variance (GV) [13] of the target static feacture vectors over an utterance $\boldsymbol{v}(\boldsymbol{y}) = [v(1), \ldots, v(D)]^\mathsf{T}$ is also modeled with a Gaussian distribution, which is expressed with a set of parameters $\boldsymbol{\Theta}^{(v)} = \{\boldsymbol{\mu}^{(v)}, \boldsymbol{\Sigma}^{(v)}\}$ as

$$P\big(\boldsymbol{v}(\boldsymbol{y}) \big| \boldsymbol{\Theta}^{(\boldsymbol{v})}\big) = \mathcal{N}\big(\boldsymbol{v}(\boldsymbol{y}); \boldsymbol{\mu}^{(v)}, \boldsymbol{\Sigma}^{(v)}\big). \quad (4)$$

Here, the GV $\boldsymbol{v}(\boldsymbol{y})$ of a time sequence of the target static feature $\boldsymbol{y} = [\boldsymbol{y}_1^\mathsf{T}, \ldots, \boldsymbol{y}_T^\mathsf{T}]^\mathsf{T}$ is calculated utterance by utterance as

$$v(d) = \frac{1}{T}\sum_{t=1}^{T}\big(y_t(d) - \frac{1}{T}\sum_{\tau=1}^{T} y_\tau(d)\big)^2. \quad (5)$$

In the conversion process, a time sequence vector of the converted static feature vectors $\hat{\boldsymbol{y}} = [\hat{\boldsymbol{y}}_1^\mathsf{T}, \ldots, \hat{\boldsymbol{y}}_T^\mathsf{T}]^\mathsf{T}$ is determined by maximizing the product of the conditional p.d.f. of $\boldsymbol{Y}$ given $\boldsymbol{X}$ and the p.d.f. of the GV as

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}}\ P\big(\boldsymbol{Y} \big| \boldsymbol{X}, \boldsymbol{\Theta}^{(\mathrm{X,Y})}\big) P\big(\boldsymbol{v}(\boldsymbol{y}) \big| \boldsymbol{\Theta}^{(v)}\big)^\lambda, \quad (6)$$

$$\text{subject to } \boldsymbol{Y} = \boldsymbol{W}\boldsymbol{y}, \quad (7)$$

where $\boldsymbol{X} = [\boldsymbol{X}_1^\mathsf{T}, \ldots, \boldsymbol{X}_T^\mathsf{T}]^\mathsf{T}$ and $\boldsymbol{Y} = [\boldsymbol{Y}_1^\mathsf{T}, \ldots, \boldsymbol{Y}_T^\mathsf{T}]^\mathsf{T}$ are time sequence vectors of the source and target feature vectors, respectively. $\boldsymbol{W}$ denotes a $2DT$-by-DT matrix that extends a time sequence vector of the static feature vectors into that of the joint static and dynamic feature vectors [14], and λ is a weight parameter, which is commonly set to $2T$. By adopting an approximation with a suboptimum mixture component sequence $\boldsymbol{m} = \{m_1, \ldots, m_T\}$, the converted static feature vector sequence is determined as follow [7]:

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}}\, P\big(\boldsymbol{Y} \big| \boldsymbol{X}, \hat{\boldsymbol{m}}, \boldsymbol{\Theta}^{(\mathrm{X,Y})}\big) P\big(\boldsymbol{v}(\boldsymbol{y}) \big| \boldsymbol{\Theta}^{(v)}\big)^\lambda. \quad (8)$$

The enhanced EL speech is generated by filtering the mixed excitation signal, which is designed according to the F_0 values, U/V information, and aperiodic components estimated from the spectral segment feature vectors, with the converted spectral features.

4.2 VAE-SPACE for pitch control

VAE-SPACE [8] has been proposed as a generative model that can represent and generate F_0 contours of both speeches and singing voices. Let $\boldsymbol{z}$ denotes a sequence of parameters governing the generating process of F_0 contours $\boldsymbol{o} = [o_1, \ldots, o_T]^\mathsf{T}$. VAE-SPACE uses an encoder to estimate the parameters of a conditional distribution $q_\phi(\boldsymbol{z}|\boldsymbol{o})$ of the latent variable $\boldsymbol{z}$ given an F_0 contour $\boldsymbol{o}$, and a decoder to estimate the parameters of a conditional distribution $p_\theta(\boldsymbol{o}|\boldsymbol{z})$ of the F_0 contour $\boldsymbol{o}$ given the latent variable $\boldsymbol{z}$. The encoder and decoder are trained simultaneously so that $q_\phi(\boldsymbol{z}|\boldsymbol{o})$ becomes consistent with the true

posterior distribution $p_\theta(\boldsymbol{z}|\boldsymbol{o}) \propto p_\theta(\boldsymbol{o}|\boldsymbol{z})p(\boldsymbol{z})$. The parameters of the networks ϕ and θ can be trained by maximizing the following variational lower bound [15]:

$$\mathcal{L}(\theta, \phi; \boldsymbol{o}) = \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{o})}[\log p_\theta(\boldsymbol{o}|\boldsymbol{z})] - D_{KL}[q_\phi(\boldsymbol{z}|\boldsymbol{o})||p(\boldsymbol{z})], \quad (9)$$

where $D_{KL}[\cdot||\cdot]$ denotes Kullback-Leibler (KL) divergence. In VAE-SPACE, the latent variable $\boldsymbol{z}$ is associated with a set of interpretable parameters so that the decoder can be seen as a generative model for synthesizing F_0 contours and the encoder can be seen as an inverse problem solver that analyzes the underlying parameters of an observed F_0 contour. In the case of speech, $\boldsymbol{z}$ is associated with a phrase/accent command sequence defined in the Fujisaki model [16], while in the case of a singing voice, it is associated with a sequence of musical scores. The name "VAE-SPACE" comes from the former case, where the VAE-based method is designed to perform statistical phrase/accent component estimation (SPACE).

A typical way of modeling $q_\phi(\boldsymbol{z}|\boldsymbol{o})$ and $p_\theta(\boldsymbol{o}|\boldsymbol{z})$ is to assume a Gaussian distribution

$$q_\phi(\boldsymbol{z}|\boldsymbol{o}) = \mathcal{N}\big(\boldsymbol{z}|\boldsymbol{\mu}_\phi(\boldsymbol{o}), \operatorname{diag}\boldsymbol{\sigma}^2_\phi(\boldsymbol{o})\big), \quad (10)$$

$$p_\theta(\boldsymbol{o}|\boldsymbol{z}) = \mathcal{N}\big(\boldsymbol{o}|\boldsymbol{\mu}_\theta(\boldsymbol{z}), \operatorname{diag}\boldsymbol{\sigma}^2_\theta(\boldsymbol{z})\big), \quad (11)$$

where $\boldsymbol{\mu}_\phi(\boldsymbol{o})$, $\boldsymbol{\sigma}^2_\phi(\boldsymbol{o})$ are the encoder outputs and $\boldsymbol{\mu}_\theta(\boldsymbol{z})$, $\boldsymbol{\sigma}^2_\theta(\boldsymbol{z})$ are the decoder outputs. While the prior distribution $p(\boldsymbol{z})$ is typically modeled as a standard Gaussian distribution with zero mean and unit variance, VAE-SPACE designs it as a specific form based on the assumption that $\boldsymbol{z}$ indicates the underlying musical scores of an F_0 contour in the case of a singing voice. Specifically, in a supervised setting where pairs of F_0 contours and musical scores are availiable, we can train the VAE by maximizing the following loss function, whici tends to maximize the likelihood of $\boldsymbol{z}$, instead of minimizing the KL divergence since the prior distribution of $\boldsymbol{z}$ is known:

$$\mathcal{L}(\theta, \phi; \boldsymbol{o}) = \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{o})}[\log p_\theta(\boldsymbol{o}|\boldsymbol{z})] + \mathbb{E}_{(\boldsymbol{o},\boldsymbol{z})\sim p_D(\boldsymbol{o},\boldsymbol{z})}[\log q_\phi(\boldsymbol{z}|\boldsymbol{o})], \quad (12)$$

where $\mathbb{E}_{(\boldsymbol{o},\boldsymbol{z})\sim p_D(\boldsymbol{o},\boldsymbol{z})}[\cdot]$ denotes the sample mean over the training data. In the generation process, a $\boldsymbol{z}$ sampled from a Gaussian distribution with the musical scores $\boldsymbol{N}$ as the mean and a variance matrix with a small constant value in the diagonal is used as the input of the trained decoder. The generated F_0 contour is then used to replace that obtained by VC to generate the excitation signal with the U/V information estimated by VC.

For network architectures, a gated convolutional neural network (CNN) [17] is used to construct the encoder and decoder to capture long- and short-term dependencies in F_0 contours. The gated CNN uses a data-driven gate called gated linear unit (GLU) $\sigma(\mathbf{H}_{l-1} * \mathbf{W}^{\mathrm{g}}_l + \mathbf{b}^{\mathrm{g}}_l)$ as a nonlinear activation function to control the information passed on in the hierarchy, where $\mathbf{H}_{l-1}$ denotes the output of the $(l-1)$-th layer, $\mathbf{b}^{\mathrm{f}}_l$ and $\mathbf{b}^{\mathrm{g}}_l$ are the weight and bias parameters of the l-th layer and σ is the sigmoid function.

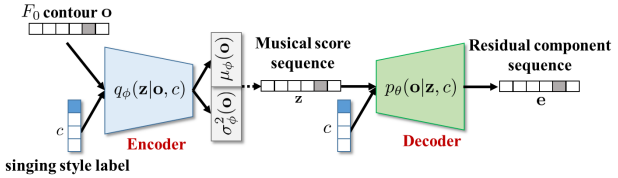


Figure 2. An overview of conditional VAE-SPACE.

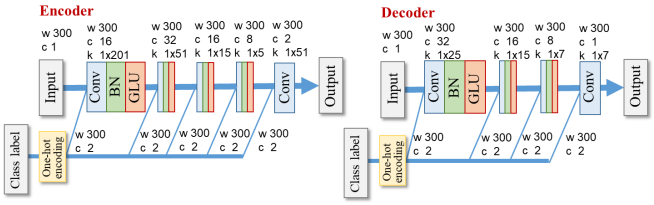


Figure 3. Network architectures of encoder and decoder used for cVAE. VAE used the same architecture excluding the class label inputs. "w", "c" and "k" denote the width, channel number and kernel size, respectively. "Conv", "BN" and "GLU" denote 1D convolution, batch normalization and gated linear unit, respectively.

To improve the performance of VAE-SPACE in modeling F_0 contours and minimize the cost of preparing pair data, we investigate two specific implementation-level problems in this paper: 1) whether a score-level alignment is necessary to train a supervised VAE-SPACE and 2) whether the performance can be improved by fine-tuning the trained decoder with the real musical score sequences generated by a sampling process during VAE pretraining. Furthermore, aiming to model different singing styles in a controllable manner, we also attempt to adopt a conditional CNN with GLUs to construct networks, which takes labels of singing styles c represented as one-hot vectors as additional inputs. The criterion of training a conditional VAE (cVAE) can be obtained merely by extending (12) [15].

5. EXPERIMENTAL EVALUATIONS

5.1 Experimental conditions

We prepared two datasets containing different pairs of data. *Dataset1* consists of EL speech samples of 21 Japanese children's songs recorded by a laryngeal speaker using an electrolarynx and the corresponding natural singing voices. 17 songs were manually segmented into 157 short phrases and used to train GMMs, and the other 4 songs were segmented into 19 phrases, and used as a test set. Each phrase was about 3∼8s long. *Dataset2* includes two versions of one Japanese song (about 4min 30s long), which were sung by a female person in normal and expressive singing styles. We segmented each song into 53 phrases and manually took alignments at the phrase and score levels, which were referred to as "unaligned" and "aligned" data, respectively.

We used the amplitude spectra as spectral feature vectors of EL speeches, and STRAIGHT analysis [9] to extract acoustic features of normal singing voices. The shift

Table 1. RMSE and standard deviation of VAE with different implementation conditions. "ft" denotes fine-tuning.

Models	RMSE
VAE-unaligned	26.1031 ± 5.5028
VAE-aligned	22.7670 ± 5.2874
VAE-aligned-ft	**21.3409 ± 5.3566**
cVAE-aligned-ft	22.2933 ± 6.3934

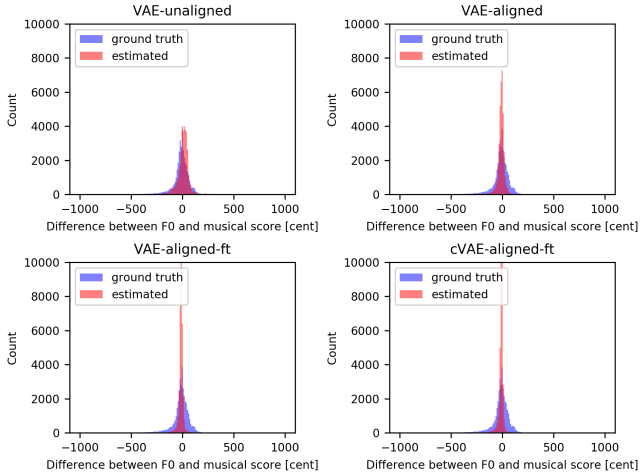


Figure 4. Histograms of the residual components e_t.

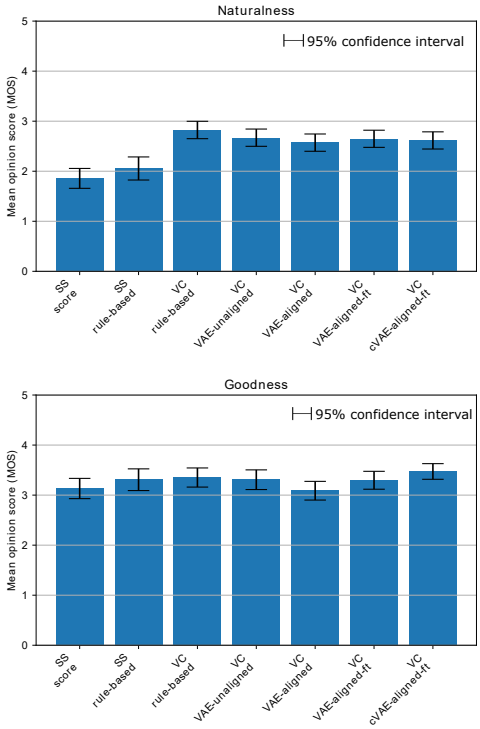


Figure 5. MOS in terms of naturalness and goodness.

length was 5 ms. The 0th through 24th mel-cepstral coefficients were used as spectral features of normal singing voices. As excitation features, a log-scaled F_0 value and aperiodic components on five frequency bands (i.e., 0-1, 1-2, 2-4, 4-6, and 6-8 kHz) were used. To obtain segmental feature vectors at each frame, we concatenated spectral feature vectors with the adjacent frames by setting $L = 4$ and performed PCA to reduce the feature dimension to 50. The numbers of mixture components of the GMMs used to estimate spectral features, aperiodic components, and F_0 including U/V information were all set at 16. Following the original VAE-SPACE paper, the output of the decoder was designed to be a sequence of residual components $\boldsymbol{e} = [e_1, \ldots, e_T]$, as shown in Fig. 2. Fig. 3 shows the architectures of the encoder and decoder. To train VAE, we used the songs sung in a normal style in *Dataset2*, and those sung in an expressive style were used as additional data for training the cVAE.

5.2 Objective evaluation

We first conducted an objective evaluation to demonstrate the performances of VAE-SPACE with different implementation conditions. We divided the songs into 7 folds and performed cross-validation. Root mean square error (RMSE) between the estimated and target F_0 contours was used as a metric to evaluate the estimation accuracy of F_0 generation. Table 1 shows the results. The objective results show that applying a score-level alignment and fine-tuning the decoder were effective in improving the accuracy of generating the target F_0 contours. However, the histograms of the residual components e_t shown in Fig. 4 reveal that both the alignment and fine-tuning decreased the dynamics of the generated F_0 contours.

Table 2. p-values calculated for method c).

	naturalness	goodness
SS+score	2.25E-11	0.1257
SS+rule-based	5.64E-07	0.7673
VC+VAE-unaligned	0.2223	0.7551
VC+VAE-aligned	0.0460	0.0561
VC+VAE-aligned-ft	0.1677	0.3413
VC+cVAE-aligned-ft	0.0976	0.3413

5.3 Subjective evaluation

To further investigate the performances of VAE-SPACE with different implementations and demonstrate the effectiveness of VC, we conducted a subjective evaluation that compared 7 methods, namely, a) SS+musical scores (SS+score), b) SS+rule-based F_0 modification (SS+rule-based), c) VC+rule-based F_0 modification (VC+rule-based), d) VC+VAE-SPACE using unaligned data (VC+VAE-unaligned), e) VC+VAE-SPACE using aligned data (VC+VAE-aligned), f) VC+VAE-SPACE using aligned data and fine-tuning (VC+VAE-aligned-ft), g) VC+cVAE-SPACE using aligned data and fine-tuning (VC+cVAE-aligned-ft). 13 evaluators participating in the experiments scored the converted singing voices in terms of the naturalness of the song and the goodness of singing using a 5-point opinion scale. The mean and 95% confidence interval of the two criteria are shown in Fig. 5 and the p-values calculated for method c) are shown in Table 2.

The results show that VC significantly improved the

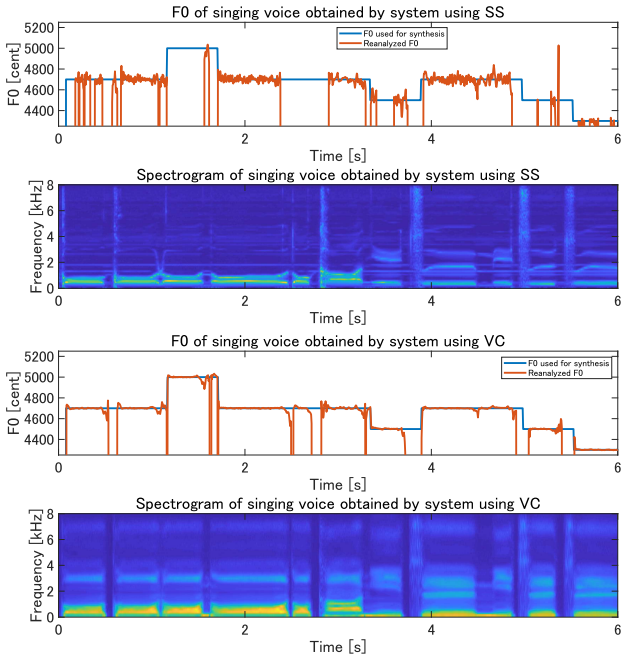


Figure 6. Examples of reanalyzed F_0 contours and spectrograms of synthesized singing voices obtained by employing SS (upper) and VC (bottom).

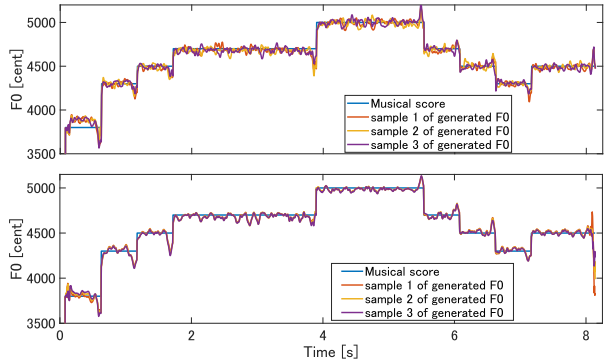


Figure 7. Generated F_0 contours with various $\boldsymbol{z}$ (upper) and class labels (bottom).

quality of converted singing voices in terms of the naturalness, which confirmed the effectiveness of the VC approach in removing the undesirable spectral components. An example of the reanalyzed F_0 contour and spectrogram is shown at the bottom of Fig. 6, which also confirmed this result. VAE-SPACE implemented with a conditional CNN achieved comparable results to the rule-based pitch control with carefully designed rules, and showed a high potential for contributing to the system. Compared with the method using unaligned data and the cVAE, the subjective results for the method using aligned data and fine-tuning were slightly lower, exhibiting the same tendency as the results shown in Fig. 4. This suggests that alignment and fine-tuning are not important in our system. We also investigated the variety of the F_0 contours generated with different sampled $\boldsymbol{z}$ and style class labels, the results of which are shown in Fig. 7. We observed some reasonable differences between the F_0 contours generated with various sampled $\boldsymbol{z}$. However, there was no notable difference observed between the F_0 contours generated with different class labels, which may have been due to the high ability of the networks to model the conditional distributions while ignoring the class labels [18, 19]. This issue will be addressed in future work.

6. DISCUSSION AND REUSABLE INSIGHTS

From the above results, it is concluded that spectrogram modification is useful and must be considered as well as F_0 control to improve the quality of singing voices. On the other hand, data alignment and fine-tuning are not essential, which means that we can increase the number of pair data at a relatively low cost to improve the estimation accuracy and the variety of styles. Furthermore, since the amount of pair data required for the VC approach is small and VAE-SPACE allows semi-supervised training with unlabeled data in addition to labeled data [8,15], it is expected that the entire system will be allowed to play its potential data efficiently, which is important for such a data-driven framework. Although cVAE-based implementation failed to represent different styles in the experiments, the framework for modeling various styles with a unified model provides us with a simple and straightforward way to control singing styles and apply style interpolation/morphing. Note that although we applied the method to model singing voices, it can also be used with other audio signals such as to generate suitable F_0 contours of musical instruments from musical instrument digital interface (MIDI) information. In addition to modeling various styles, we can extend this method to generate the F_0 contours of multiple instruments.

7. CONCLUSIONS

This paper proposed an improved singing aid system for laryngectomees based on a previously proposed system that converts EL speeches into singing voices according to the additionally inputted melodic information. The proposed system uses a statistical VC approach to transform the phonetic information extracted from EL speeches into those of natural speeches, and VAE-SPACE to perform pitch control. We investigated the importance of well-aligned pair data and the fine-tuning process for improving the performance and a conditional version of VAE-SPACE for modeling multiple singing styles with a unified model. The experimental results demonstrated that 1) the VC approach was effective in significantly improving the naturalness of singing voices, 2) the effectiveness of well-aligned pair data and fine-tuning was limited, and 3) VAE-SPACE was able to achieve comparable results to a carefully designed rule-based approach in generating F_0 contours.

8. ACKNOWLEDGEMENTS

This work was supported by JST, PRESTO Grant Number JPMJPR1657, and JSPS KAKENHI Grant 18J20059.

9. REFERENCES

[1] M. Umbert, J. Bonada, M. Goto, T. Nakano, and J. Sundberg. "Expression control in singing voice synthesis: Features, approaches, evaluation, and challenges," *IEEE Signal Processing Magazine*, Vol. 32, No. 6, pp. 55–73, 2015.

[2] K. Morikawa and T. Toda. "Electrolaryngeal speech modification towards singing aid system for laryngectomees," in Proc. *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 610-–613. 2017.

[3] T. Saitou, M. Unoki, and M. Akagi. "Development of an f0 control model based on f0 dynamic characteristics for singing-voice synthesis," *Speech communication*, Vol. 46, No. 3-4, pp. 405-–417, 2005.

[4] T. Saitou, M. Goto, M. Unoki, and M. Akagi. "Speech-to-singing synthesis system: Vocal conversion from speaking voices to singing voices by controlling acoustic features unique to singing voices," in Proc. *the 10th National Conference on Man-Machine Speech Communication (NCMMSC)*, 2009.

[5] S. Boll. "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. 27, No. 2, pp. 113-–120, 1979.

[6] T. Toda, M. Nakagiri, and K. Shikano. "Statistical voice conversion techniques for body-conducted unvoiced speech enhancement," *IEEE Trans. on Audio, Speech, and Language Processing*, Vol. 20, No. 9, pp. 2505-–2517, 2012.

[7] K. Nakamura, T. Toda, H. Saruwatari, and K. Shikano. Speaking-aid systems using gmm-based voice conversion for electrolaryngeal speech. Speech Communication, 54(1):134–146, 2012.

[8] K. Tanaka, H. Kameoka, and K. Morikawa. "Vaespace: Deep generative model of voice fundamental frequency contours," in Porc. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018)*, pp. 5779-–5783, 2018.

[9] M. Morise, F. Yokomori, and K. Ozawa. "World: a vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Trans. on Information and Systems*, Vol. 99, No. 7, pp. 1877-–1884, 2016.

[10] M. Nishimura, K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda. "Singing voice synthesis based on deep neural networks," in Proc. Interspeech, pp. 2478-–2482, 2016.

[11] Y. Ohishi, H. Kameoka, D. Mochihashi, and K. Kashino. "A stochastic model of singing voice f0 contours for characterizing expressive dynamic components," in Proc. *13th Annual Conference of the International Speech Communication Association*, 2012.

[12] K. Saino, H. Zen, Y. Nankaku, A. Lee, and K. Tokuda. "An hmm-based singing voice synthesis system," in Proc. *9th International Conference on Spoken Language Processing*, 2006.

[13] T. Toda, A. W. Black, and K. Tokuda. "Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory," *IEEE Trans. on Audio, Speech, and Language Processing*, Vol. 15, No. 8, pp. 2222-–2235, 2007.

[14] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura. "Speech parameter generation algorithms for HMM-based speech synthesis," in Proc. *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing (Cat. No. 00CH37100)*, Vol. 3, pp. 1315-–1318, 2000.

[15] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. "Semi-supervised learning with deep generative models," in *Advances in neural information processing systems*, pp. 3581—3589, 2014.

[16] H. Fujisaki. "A note on the physiological and physical basis for the phrase and accent components in the voice fundamental frequency contour," *Vocal physiology: Voice production, mechanisms and functions*, pp. 347–355, 1988.

[17] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. "Language modeling with gated convolutional networks," in Proc. *the 34th International Conference on Machine Learning* Vol. 70, pp. 933-–941, 2017.

[18] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in neural information processing systems*, pp. 2172-–2180, 2016.

[19] H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo. "ACVAE-VC: Non-parallel many-to-many voice conversion with auxiliary classifier variational autoencoder," arXiv preprint *arXiv:1808.05092*, 2018.

Session G

THE BACH DOODLE: APPROACHABLE MUSIC COMPOSITION WITH MACHINE LEARNING AT SCALE

Cheng-Zhi Anna Huang[♯] **Curtis Hawthorne**[♯] **Adam Roberts**[♯] **Monica Dinculescu**[♯]
James Wexler[†] **Leon Hong**[⋆] **Jacob Howcroft**[⋆]
{annahuang,fjord,adarob,noms,jwexler,leonhong,jhowcroft}@google.com
♯ Google Brain Magenta † Google Brain Pair ⋆ Google Doodle

1. ABSTRACT

To make music composition more approachable, we designed the first AI-powered Google Doodle, the Bach Doodle [1], where users can create their own melody and have it harmonized by a machine learning model (Coconet [22]) in the style of Bach. For users to input melodies, we designed a simplified sheet-music based interface. To support an interactive experience at scale, we re-implemented Coconet in TensorFlow.js [32] to run in the browser and reduced its runtime from 40s to 2s by adopting dilated depthwise separable convolutions and fusing operations. We also reduced the model download size to approximately 400KB through post-training weight quantization. We calibrated a speed test based on partial model evaluation time to determine if the harmonization request should be performed locally or sent to remote TPU servers. In three days, people spent 350 years worth of time playing with the Bach Doodle, and Coconet received more than 55 million queries. Users could choose to rate their compositions and contribute them to a public dataset, which we are releasing with this paper. We hope that the community finds this dataset useful for applications ranging from ethnomusicological studies, to music education, to improving machine learning models.

2. INTRODUCTION

Machine learning can extend our creative abilities by offering generative models that can rapidly fill in missing parts of our composition, allowing us to see a prototype of how a piece could sound. To celebrate J.S. Bach's 334th birthday, we designed the Bach Doodle to create an interactive experience where users can rapidly explore different possibilities in harmonization by tweaking their melody and requesting new harmonizations. The harmonizations are powered by Coconet [22], a versatile generative model of counterpoint that can fill in arbitrarily incomplete scores.

Creating this first AI-powered doodle involved overcoming challenges in user interaction and interface design, and also technical challenges in both machine learning and in infrastructure for serving the models at scale. For inputting melodies, we designed a simplified sheet music interface that facilitates easy trial and error and found that users adapted to it quickly even when they were not familiar with western music notation.

As most users do not have dedicated hardware to run machine learning models, we re-implemented Coconet in TensorFlow.js [32] so that it could run in the browser. We reduced the model run-time from 40s to 2s by adopting dilated depth-wise separable convolutions and fusing operations, and we reduced the model download size to ∼400KB through post-training weight quantization. To prepare for large-scale deployment, we calibrated a speed test to determine if a user's device is fast enough for running the model in the browser. If not, the harmonization requests were sent to remote TPU servers.

Users in 80% of sessions explored multiple harmonizations, and 53.8% of the harmonizations were rated as positive. One complaint from advance users was the presence of parallel fifths (P5s) and octaves (P8s). We analyzed 21.8 million harmonizations and found that P5s and P8s occur on average 0.365/measure and 0.391/measure respectively. Furthermore, P5s and P8s were more common when user input was out of distribution, and fewer P5s and P8s were correlated with positive user feedback.

3. RELATED WORK

Machine learning has been used in algorithmic music composition to support a wide range of musical tasks [5, 13, 19, 28, 29]. Melody harmonization is one of the canonical tasks [7, 11, 20, 26], encourages human-computer interaction [3, 14, 21, 25, 33], and is particularly approachable for novices. Different interfaces and tools have been developed to make the interaction experience more accessible. For example, in MySong [31], users can sing a melody and have the system harmonize it. In Hyperscore [12], users can draw multiple levels of "motifs" on a graphical sketchpad and have them harmonized according to the tension curve they specified. Startups such as JukeDeck and Amper Music offer APIs that allow users to describe a piece through timing and mood tags. Opensource libraries such as Magenta.js [30] allow machine learning models to be

used in digital audio work stations. For score-based interaction, FlowComposer [27] offers an augmented lead-sheet based interface, while DeepBach [17] demonstrates interactive chorale composition in MuseScore which uses standard music notation.

4. THE BACH DOODLE

4.1 A walk through of the user experience

The Bach Doodle user experience begins by demonstrating 4-part harmony using two measures of a Bach chorale, *Ach wie flüchtig, ach wie nichtig, BWV 26*. By playing the soprano line alone, followed by soprano and alto, and then all four voices, users are shown how the harmony enhances the melody. Users are then presented with two measures of blank sheet music, in the treble clef, with a key signature of C major, in standard time. There are four vertical lines in each measure to indicate the beats which give the user visual cues on where to put notes.

The user enters a monophonic melody using quarter and eighth notes. The note duration is automatic. If a note is entered on a beat, it is a quarter note by default. However, if a note is added after the beat, the existing quarter note becomes an eighth note. This simple interface makes it easy for users with no musical knowledge to input a composition, and can be seen in Figure 1. If the user clicks on the "star" button on the left-hand side of the sheet music, they can enter "advanced mode". This allows the user to input sixteenth notes anywhere on the staff. It also enables a control where the key can be changed to any of the 12 key signatures. This mode is hidden because it can be overwhelming for new users. It is also easier to make less enjoyable music this way, for example by going off key, or making the music overly complex.

Clicking the "Harmonize" button sends the generation request to either TensorFlow.js or the TPU server. When the response is ready, it is presented to the user one voice at a time, listing them out: "Soprano", "Alto", "Tenor", "Bass". The voices are color-coded to illustrate the harmonization in relation to the soprano input notes (Figure 2).

Figure 1. The user interface of the Bach Doodle, where users can input a melody and then click on the green "Harmonize" button on the bottom right to request a harmonization.

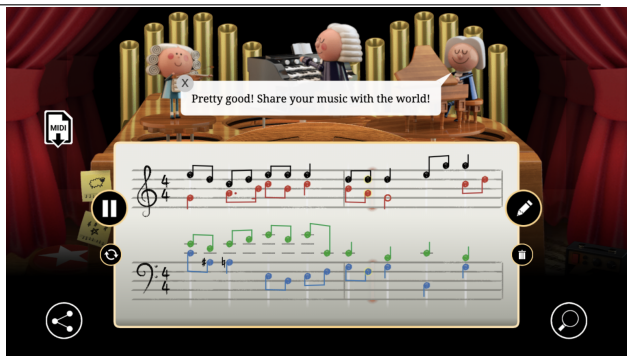


Figure 2. The harmonization returned by Coconet is notated in color, carrying the alto, tenor, and bass voices.

4.2 Design challenges

Celebrating J.S. Bach's birthday using machine learning presented many unique design opportunities as well as some user experience challenges. One of the main goals was to empower people with the feeling that they could augment their own creativity in ways not previously thought possible, by allowing them to directly collaborate with a machine learning model. Another important goal was to convey the message that machine learning is not "magical" or incomprehensible, but rather a science that can be understood. Finally, a notable challenge was to ensure that these aims would be met for a large diversity of individuals, from children who have not yet learned to read to experts of music and technology.

In order to acquire early feedback on the design, user tests were employed. Over the course of the project, dozens of people were asked to play the demos and comment on their experience through both pre-defined questions and open comments. The first user test in the development process revealed that many people do not fully understand the concept of harmony, but fortunately, further testing showed that short animated musical examples were enough for people to comprehend these concepts quickly. Also, user tests indicated that only a small subset of people could read standard music notation. Our intuition was that using standard notation, rather than a grid based interface would be intuitive and frictionless to anybody only if the user interface (UI) was responsive with animations and sound and also if the note input interface was kept simple. Further user testing of the standard notation input UI proved this to be correct.

In order to accommodate people of all ages and experiences, a common technique employed is to remove any advanced feature or unexpected delight from the core experience and instead integrate them as "easter eggs". This allows people of all skill levels to experience the full core experience without feeling frustrated, while also giving the rarer advanced user more features. While the core experience primarily allows eighth notes and tempo changes, clicking on a special button in the background additionally allowed the user to add sixteenth notes and change the key – two features that are very confusing to those without musical backgrounds.

4.3 Reusable insights

For future projects, we have shown that if the technology being used is unfamiliar or perceived as "scary" to those who know little about it, tethering the experience to a familiar story and visuals can be a successful strategy. Most people have a limited understanding of musical concepts such as harmony and standard notation, but it is possible that people of all ages can quickly acquire an intuitive understanding of musical concepts through carefully designed animated audiovisuals and a simple and responsive UI. Additionally, injecting content into loading screens could not only make loading times feel shorter but also be an excellent space for educational content. Finally, user testing is crucial when trying to create an experience using new technology that encompasses a large and diverse audience – it can reveal serious issues and shortcomings that are not obvious due to the team's own background and domain knowledge.

5. TECHNICAL CHALLENGES

In order for users to interact with Coconet via a web interface, we needed to either port it to run client-side on the user's device or host the model on a server with sufficient speed and capacity to support the number of requests we were expecting. In fact, we did both: we ported the model to TensorFlow.js (TF.js) so that it could run on user devices and added support for Tensor Processing Units (TPU) so that it could be served on Google Cloud. By running a simple test on users' devices to determine the speed of core tensor operations, we were able to determine whether to use the TF.js implementation locally, or fall back to a TPU server to handle the computation. In the end, 47.4% of all harmonizations were done locally, in TF.js.

5.1 Background: Coconet

Coconet [22] [1] is a versatile generative model of musical counterpoint that can fill in arbitrarily incomplete scores, as illustrated in Figure 3.

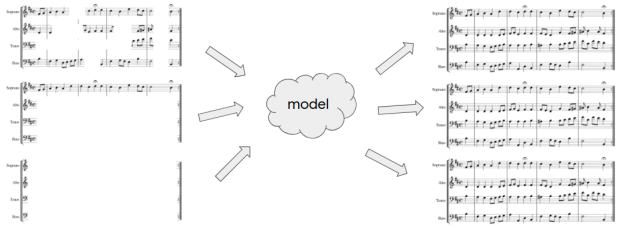


Figure 3. Coconet can be used in a wide range of musical tasks, such as completing partial scores, harmonizing melodies and generating from scratch.

Coconet represents counterpoint as a stack of piano rolls encoded in a binary three-dimensional tensor $\mathbf{x} \in \{0,1\}^{I \times T \times P}$, where I, T, and P denotes the number of instruments, time steps, and pitches respectively. $\mathbf{x}_{i,t,p} = 1$ if the ith instrument plays pitch p at time t. Each instrument is assumed to play exactly one pitch at a time, therefore $\sum_p \mathbf{x}_{i,t,p} = 1$ for all (i,t) positions. We also focus on four-part Bach chorales as used in prior work [2, 4, 8, 16, 18, 24], and assume $I = 4$ throughout.

Conventional approaches often factorize the joint distribution $p(\mathbf{x})$ into conditional distributions of the form $p(\mathbf{x}_k \mid \mathbf{x}_{<k})$, where k indexes a sequence in some predetermined ordering such as chronological. In contrast, Coconet is an instance of orderless NADE [34, 35] and offers direct access to all conditionals of the form $p(\mathbf{x}_{i,t} \mid \mathbf{x}_C)$, where C selects a fragment of a musical score $\mathbf{x}$ and $(i,t) \in \neg C$ is in its complement (i.e. the missing parts). To train Coconet, we sample a training example $\mathbf{x}$, choose uniformly how many variables to erase, i.e. $|\neg C| \sim U(1, D)$, and then choose uniformly the particular subset of variables $\neg C$ to erase. The input $\mathbf{X} \in \{0,1\}^{2I \times T \times P}$ is obtained by erasing the piano rolls $\mathbf{x}$ to obtain incomplete piano rolls $\mathbf{x}_C$ and concatenating this with the corresponding masks, as shown in Figure 4 (top left) where the yellow gaps indicate erased positions with all pitches set to zero.

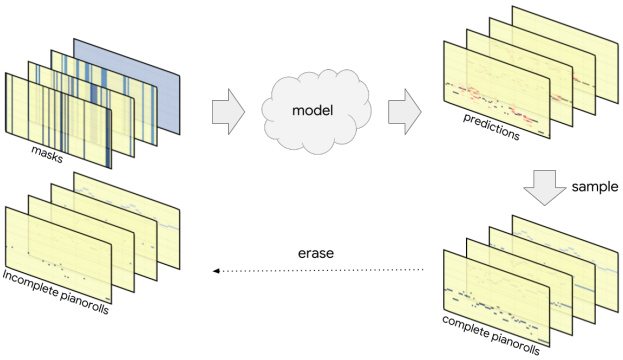


Figure 4. Coconet's generation loop using Gibbs sampling, alternating between (top) filling in the missing parts and (bottom) erasing random parts to improve the score through rewriting.

The output predictions for each (i,t) position is a softmax over the set of pitches P (top right of Figure 4). The negative loglikelihood loss is given below, which involves reweighing by the number of variables erased to ensure that all conditionals are trained equally.

$$\mathcal{L}(\mathbf{x}; C) = -\frac{1}{|\neg C|} \sum_{(i,t) \in \neg C} \sum_p \mathbf{x}_{i,t,p} \log p(\mathbf{x}_{i,t,p} \mid \mathbf{x}_C, C)$$

In contrast to generating from left to right in one pass, Coconet uses Gibbs sampling to improve sample quality through rewriting (see [22] for convergence analysis). Figure 4 shows how the procedure iterates between filling in missing parts and then erasing other parts so that they could be rewritten given the updated context.

5.2 Improving and speeding up Coconet

The original Coconet uses dense convolutions, where filter weights and their mixing weights W are fully connected (Equation 1). This makes it unable to fully leverage GPU parallelization in TF.js (see Section 5.3.2).

[1] Blog post: https://magenta.tensorflow.org/coconet Code: https://github.com/tensorflow/magenta/tree/master/magenta/models/coconet

Let s, q index the pitch and time dimension in filters, and i, j the input and output channels. In dense convolutions, each output position (p, t, j) indexed by pitch, time and output channel is a sum over the resultant input channels and also over the positions in each filter (given by the neighbourhood function). For a 3-by-3 filter this summing is over the 9 positions.

In contrast, depthwise separable convolutions [6], shown in Equation 2, factorizes the dense tensor W into a depthwise tensor V and a pointwise U. As a result, the multiplications between V and X can be parallelized over the input channels i in the inner sum of Equation 3.

$$Y^{\text{dense}}_{p,t,j} = \sum_i \sum_{s,q \in \text{neighborhood(p, t)}} W_{s,q,i,j} X_{s,q,i} \quad (1)$$

$$Y^{\text{dsep}}_{p,t,j} = \sum_i \sum_{s,q \in \text{neighborhood(p,t)}} U_{i,j} V_{s,q,i} X_{s,q,i} \quad (2)$$

$$= \sum_i U_{i,j} \sum_{s,q \in \text{neighborhood(p,t)}} V_{s,q,i} X_{s,q,i} \quad (3)$$

To further speed up Coconet, we adopt dilated convolutions to grow the receptive field exponentially to reduce the number of layers needed. As in [36], where in each block the dilation factors double in each layer for both the pitch and time dimension and then the block repeats.

The original Coconet was trained on eight measures (T=128). However, the Bach Doodle is designed for two measures (T=32), so we retrained the model with the original architecture and saw that the loss increased from 0.57 to 0.62 (show in Table 1), possibly because there is less context. Switching from dense to depthwise separable convolutions reduced the loss, requiring more filters but fewer layers. Since Tensorflow.js allows for parallelization across filters, this still resulted in much faster generation (see Section 5.3.2). Dilated convolutions reduced both the number of layers and number of filters and also reducing the loss. The particular scheme we used is 7 blocks of dilation rates $(1, 2, 4, 8, 16, 16)$ for the pitch dimension and $(1, 2, 4, 8, 16, 32)$ for the time dimension.

Table 1. Comparing frame-wise negative loglikelihood (NLL) on the 16th-note resolution as in [22] and the generation time (in seconds) when model was ported to Tensorflow.js (see Section 5.3.2 for details). The bottom three rows are all trained on two-measure (T=32) random crops.

Convolution type	NLL	run time
Dense (T=128), 64L, 128f	0.57	
Dense (T=32), 64L, 128f	0.62	> 40s
Depthwise separable, 48L, 192f	0.59	7s
Dilated, 45L (7 blocks), 128f	0.58	∼4s

5.3 Porting Coconet to the Browser

JavaScript is the standard language for browser-based computation, but native JavaScript is too inefficient to handle the the amount of computation required by Coconet in a reasonable time for the interaction we desired. TF.js is a javascript library for GPU-accelerated machine learning. It makes use of WebGL [2] to leverage the parallel processing power of GPUs to speed up machine learning operations, supporting the development and training of models, as well as deployment of trained models on web browsers. By enabling users to run trained models directly in their web browsers, it alleviates the need for remote servers to run those models. This can enable faster, more interactive experiences between a user and a machine learning system.

While some models can easily be ported to TF.js using a conversion script, Coconet's Python TensorFlow implementation used some ops that did not yet exist in TF.js (e.g., `cumsum`), and we also needed the flexibility to optimize the performance of the model for our use case. We therefore manually re-implemented Coconet in TF.js ourselves and have made the code opensource [3] . We also contributed missing ops to TF.js with WebGL fragment shader code for GPU acceleration.

5.3.1 UI Challenges

TF.js makes use of the async/await pattern for access to outputs of models and individual TensorFlow operations. During inference, users receive a callback for when GPU operations have completed and the result is ready to be consumed. In this way, there is no blocking of the UI while waiting for model results. In practice, with large models like Coconet (which includes many repeated sampling steps of a deep network), it is still important to cede control back to the UI explicitly during the course of the model operations, which can be done with the `tf.nextFrame()` operation. Our op-by-op code port of the network allowed us to add these occasional UI breaks, which avoided a poor user experience where the page would freeze for multiple seconds during model prediction.

5.3.2 Performance Challenges

The initial port of Coconet to TF.js took over 40 seconds to do one harmonization. For a satisfying user experience, we needed to lessen this latency to below 5 seconds. While TF.js is able to take advantage of GPU acceleration, WebGL does not directly support the types of tensor operations used in deep learning. Instead, these operations must be implemented as shader programs, which were originally intended to compute the color for pixels during graphics rendering. This mismatch leads to inefficiencies that sometimes vary by operation. It turned out that the (unavoidably) inefficient shader implementation of convolutional layers were the main culprit. By switching to depthwise-separable convolutions, however, we were able to avoid many of these performance issues, reducing generation time to 7 seconds.

As we run Coconet through 64 Gibbs sampling steps, any improvement to operations that are used in this loop

[2] `https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API`

[3] `https://tensorflow.github.io/magenta-js/music/classes/_coconet_model_.coconet.html`

could lead to a significant saving. We wrote a custom operation using WebGL shaders to fuse together the operations used in our initial TF.js implementation of the annealing schedule. This schedule by [37] is a sequence of simple element-by-element operations that is run on every sampling step during harmonization. Because of the simplicity of the operations (a scalar subtraction, multiplication, division, and a max operation), we were able to easily fuse them into a single operation that avoided the overhead of executing multiple shader programs on the GPU, speeding up inference by about 5%. The combined savings of adding depthwise-separable convolutions, shrinking the model by using dilated convolutions, and using the fused schedule operation resulted in a reduction of the model latency from 40s to 2s.

5.3.3 Download Size

Due to the number of users we intended to reach as well as the variety of locations, devices, and bandwidth limits they would have, we needed to ensure the download size of the model weights was as small possible. To achieve this goal, we implemented support for post-training weight quantization and contributed it to TF.js. This quantization compresses each `float32` weight tensor by mapping the full range of its dimensions down to 256 uniformly-spaced buckets in order to represent them as `int8` values, which are then stored along with `float32` min and scale values used to recover the range. During model initialization, the weights are converted back to `float32` tensors using linear interpolation. By using these quantization, we were able to reduce the size of the downloaded weights by approximately 4, resulting in a payload of $\sim$ 400KB without any noticable sacrifice in quality.

5.4 Balancing Load Between Tensorflow.js and TPU

We ideally wanted to run the harmonization model completely on end-user devices using TF.js to avoid the need for serving infrastructure, which adds cost, effort, and additional points of failure. But the speed of harmonization differs by user device, with older and lower-end devices taking longer to run the TF.js model code. For devices where harmonization take more than a few seconds, the harmonization is instead done by the cloud-served model. The first step in checking if a device can run harmonization locally is to check if WebGL is supported on the device, since that is required for using GPU-acceleration through TF.js. If WebGL is supported then we perform a speed test on the model, running a sample melody through its first four layers. If the latency of this model inference is below a set threshold, then the TF.js version is used. As there is overhead on the first inference of a model in TF.js, due to initial loading of the model weight textures onto the GPU, we actually run the speed test twice and use the second measurement to make the decision.

6. DATASET RELEASE AND ANALYSIS

6.1 Data structure

Every user who interacted with the Bach Doodle had the opportunity to add their composition to a dataset. We make this entire dataset available at `https://g.co/magenta/bach-doodle-dataset` under a Creative Commons license. Of more than 55 million requests, the user contributed dataset contains over 21.6 million miniature compositions. The compositions are split across 8.5 million sessions. Each session represents an anonymous user's interaction with the Bach Doodle over a single pageload and may contain multiple data points. Each data point consists of the user's input melody, the 4-voice harmonization returned by Coconet, as well as other metadata: the country of origin, the user's rating, the composition's key signature, how long it took to compose the melody, and the number of times the composition was listened to.

6.2 Analysis

We present some preliminary analysis of the dataset to shed some light on how users interacted with the doodle. Out of the 21.6 million sequences in the dataset, about 14 million (or 65.7%) are unique pitch sequences, that are not repeated anywhere in the dataset (without considering timing information). Overall, the median amount of time spent composing a sequence was 25.5 seconds, and sequences were listened to for a median of 3 loops, with a total of 78.2 million loops listened across the entire dataset.

The sequences come from 109 different countries, with the United States, Brazil, Mexico, Italy, and Spain ranking in the top 5. Countries that had a small number of sequences were all grouped together in a separate category, to minimize the possibility of identifying users. While many sessions ($\sim$20%) contained only one request for harmonization (shown in Figure 5), most sessions had 2 or more harmonizations, either of the same melody, or of a different one. As shown in Figure 6, more than 5 million of the input sequences used the maximum number of notes in the default version of the doodle, which is 16. It is interesting to note that despite being an Easter egg, 7.6% of user sessions discovered the advanced mode that allowed them to enter longer sequences.

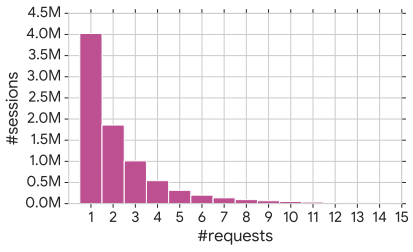


Figure 5. Histogram: number of requests per session

The doodle has 3 presets: *Twinkle Twinkle Little Star*, *Mary had a Little Lamb*, and the beginning to *Bach's Toccata and Fugue in D Minor, BWV 565*, which are the 3 most repeated sequences. However, there are also shows

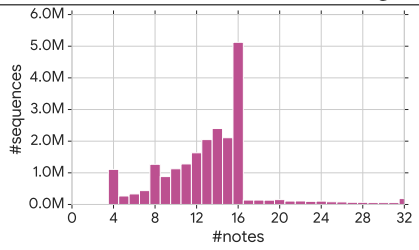


Figure 6. Histogram: length of sequences

some surprising runner ups, such as Beethoven's *Ode to Joy*, and *Megalovania*, a popular song from the game *Undertale*, as well as some regional hits [4]. Overall, users enjoyed their harmonizations, with 53.8% of all compositions rated as "Good". Figure 7 gives the breakdown of user ratings.

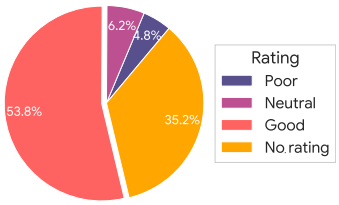


Figure 7. Breakdown of user ratings on harmonized compositions.

6.3 Parallel Fifths and Parallel Octaves

The Coconet model that powered the Bach Doodle was trained to produce harmonizations in the style of Bach chorales, and one well known characteristic of Bach's counterpoint writing is how carefully he followed the rule of avoiding parallel fifths (P5s) and parallel octaves (P8s). However, one complaint from advanced users of the app was the presence of P5s and P8s in the output of the model. Here, we present some analysis of how frequently and under what circumstances such outputs occurred. To identify the P5 and P8 occurrences, we used *music21* [9].

First, we looked at how frequently P5s and P8s appeared in our training data. We were surprised to find that in the 382 Bach chorale preludes we used in our train and validation sets, there were 132 instances of P5s (0.023/measure) and 51 instances of P8s (0.009/measure). Given this prevalence, the model may learn to output this kind of parallel motion. However, many of these instances can be "excused" because they occur under special circumstances such as at phrase boundaries or when using non-chord tones [10, 15]. Unfortunately, our training data does not include key signatures, time signatures, or fermatas, so the model likely learned to treat P5s as more permissible than was actually the case in Bach's music.

We then examined the output of the model to see if P5s/P8s occurred more frequently when user input was outside the training distribution and if the absence of

[4] Visit `https://g.co/magenta/bach-doodle-dataset` to interact with visualizations of the top repeated melodies overall and in each region, as well as regional unique hits.

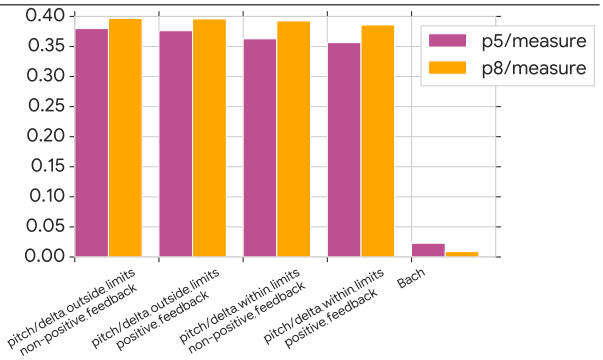


Figure 8. Parallel fifths and octaves per measure

P5s/P8s was correlated with positive user feedback. We first split the output based on whether users gave positive feedback or not (non-positive feedback includes neutral, negative, and the absence of feedback). Next, we split based on whether user input was within the same pitch range as the soprano lines in the training data (MIDI pitches from 60 through 81) and whether the maximum delta between consecutive pitches exceeded that of the training data (1 octave).

In total, we found 15,816,599 P5s (0.365/measure) and 16,949,818 P8s (0.391/measure) in the model output. Results split into the four categories are shown in Figure 8. As hypothesized, P5s/P8s were more common when user input was out of distribution, and their absence correlated with positive user feedback. A Kruskal-Wallis H test for both the number of P5s and P8s showed that there is at least one statistically significant difference between the four categories with $p < 1\text{e}{-4}$. Further, Mann-Whitney rank tests between the categories showed significant differences, each with $p < 1\text{e}{-4}$. The correlation between fewer P5s/P8s and positive user feedback is particularly interesting. This could either indicate that users prefer music with fewer P5s/P8s or it could simply mean that when the model produces poor output, P5s/P8s tend to be a feature of that output. In any case, the presence of P5s/P8s seems to be a useful proxy metric for model output quality. In future work, it could be a useful signal during training (similar to [23]), evaluation, or perhaps even during inference where it could trigger additional Gibbs sampling steps.

7. CONCLUSION

The Bach Doodle enabled large-scale participation in baroque-style counterpoint composition through an intuitive sheet music interface assisted by machine learning. We hope this encourages more creative apps that allow novices and artists to interact with music composition and machine learning in approachable ways. With this paper, we are releasing a dataset of 21.6 million instances of human-computer collaborative miniature compositions, along with meta-data such as user rating and country of origin. We hope the community will find it useful for ethnomusicological studies, music education, or improving machine learning models.

8. ACKNOWLEDGEMENTS

Many thanks to Ann Yuan, Daniel Smilkov and Nikhil Thorat from Tensorflow.js for their expert assistance. A big shoutout to Pedro Vergani, Rebecca Thomas, Jordan Thompson and others on the Doodle team for their contribution to the core components of the Doodle. Thank you Lauren Hannah-Murphy and Chris Han for keeping us on track. Thank you to Magenta colleagues for their support. Thank you Tim Cooijmans for co-authoring the blog post.

9. REFERENCES

[1] Celebrating Johann Sebastian Bach. `https://www.google.com/doodles/celebrating-johann-sebastian-bach`. Accessed: 2019-04-04.

[2] Moray Allan and Christopher KI Williams. Harmonising chorales by probabilistic inference. *Advances in neural information processing systems*, 17:25–32, 2005.

[3] Gérard Assayag, Georges Bloch, Marc Chemillier, Arshia Cont, and Shlomo Dubnov. Omax brothers: a dynamic yopology of agents for improvization learning. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 125–132. ACM, 2006.

[4] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *International Conference on Machine Learning*, 2012.

[5] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. Deep learning techniques for music generation-a survey. *arXiv preprint arXiv:1709.01620*, 2017.

[6] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[7] Ching-Hua Chuan and Elaine Chew. A hybrid system for automatic generation of style-specific accompaniment. In *Proceedings of the 4th International Joint Workshop on Computational Creativity*, pages 57–64. Goldsmiths, University of London London, 2007.

[8] David Cope. *Computers and musical style*. Oxford University Press, 1991.

[9] Michael Scott Cuthbert and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In *International Society for Music Information Retrieval*, 2010.

[10] Luke Dahn. Consecutive 5ths and octaves in bach chorales. `https://lukedahn.wordpress.com/2016/04/15/consecutive-5ths-and-octaves-in-bach-chorales/`. Accessed: 2019-04-12.

[11] Mary Farbood and Bernd Schöner. Analysis and synthesis of palestrina-style counterpoint using markov chains. In *Proceedings of the International Computer Music Conference*, 2001.

[12] Morwaread M Farbood, Egon Pasztor, and Kevin Jennings. Hyperscore: a graphical sketchpad for novice composers. *IEEE Computer Graphics and Applications*, 24(1):50–54, 2004.

[13] Jose D Fernández and Francisco Vico. Ai methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48:513–582, 2013.

[14] Rebecca Anne Fiebrink. Real-time human interaction with supervised learning algorithms for music composition and performance. *PhD dissertation, Princeton University*, 2011.

[15] George Fitsioris and Darrell Conklin. Parallel successions of perfect fifths in the bach chorales. *MUSICAL STRUCTURE*, page 52, 2008.

[16] Kratarth Goel, Raunaq Vohra, and JK Sahoo. Polyphonic music generation by modeling temporal dependencies using a RNN-DBN. In *International Conference on Artificial Neural Networks*, 2014.

[17] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. Deepbach: a steerable model for bach chorales generation. In *International Conference on Machine Learning*, pages 1362–1371, 2017.

[18] Gaëtan Hadjeres, Jason Sakellariou, and François Pachet. Style imitation and chord invention in polyphonic music with exponential families. *arXiv preprint arXiv:1609.05152*, 2016.

[19] Dorien Herremans, Ching-Hua Chuan, and Elaine Chew. A functional taxonomy of music generation systems. *ACM Computing Surveys (CSUR)*, 50(5):69, 2017.

[20] Dorien Herremans and Kenneth Sörensen. Composing fifth species counterpoint music with a variable neighborhood search algorithm. *Expert systems with applications*, 40(16):6427–6437, 2013.

[21] Cheng-Zhi Anna Huang, Sherol Chen, Mark Nelson, and Doug Eck. Mixed-initiative generation of multichannel sequential structures. In *International Conference on Learning Representations Workshop Track*, 2018.

[22] Cheng-Zhi Anna Huang, Tim Cooijmnas, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. *ISMIR*, 2017.

[23] Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning

of sequence generation models with kl-control. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1645–1654. JMLR. org, 2017.

[24] Feynman Liang. Bachbot: Automatic composition in the style of bach chorales. *Masters thesis, University of Cambridge*, 2016.

[25] Francois Pachet. The continuator: Musical interaction with style. *Journal of New Music Research*, 32(3):333–341, 2003.

[26] François Pachet and Pierre Roy. Musical harmonization with constraints: A survey. *Constraints*, 6(1):7–19, 2001.

[27] Alexandre Papadopoulos, Pierre Roy, and François Pachet. Assisted lead sheet composition using flowcomposer. In *International Conference on Principles and Practice of Constraint Programming*, pages 769–785. Springer, 2016.

[28] George Papadopoulos and Geraint Wiggins. Ai methods for algorithmic composition: A survey, a critical view and future prospects. In *AISB Symposium on Musical Creativity*, volume 124, pages 110–117. Edinburgh, UK, 1999.

[29] Philippe Pasquier, Arne Eigenfeldt, Oliver Bown, and Shlomo Dubnov. An introduction to musical metacreation. *Computers in Entertainment (CIE)*, 14(2):2, 2016.

[30] Adam Roberts, Curtis Hawthorne, and Ian Simon. Magenta. js: A javascript api for augmenting creativity with deep learning. 2018.

[31] Ian Simon, Dan Morris, and Sumit Basu. Mysong: automatic accompaniment generation for vocal melodies. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 725–734. ACM, 2008.

[32] Daniel Smilkov, Nikhil Thorat, Yannick Assogba, Ann Yuan, Nick Kreeger, Ping Yu, Kangyi Zhang, Shanqing Cai, Eric Nielsen, David Soergel, et al. Tensorflow. js: Machine learning for the web and beyond. *arXiv preprint arXiv:1901.05350*, 2019.

[33] Bob L Sturm, Oded Ben-Tal, Una Monaghan, Nick Collins, Dorien Herremans, Elaine Chew, Gaëtan Hadjeres, Emmanuel Deruty, and François Pachet. Machine learning research that matters for music creation: A case study. *Journal of New Music Research*, 48(1):36–55, 2019.

[34] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1):7184–7220, 2016.

[35] Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. In *In Proceedings of the International Conference on Machine Learning*, 2014.

[36] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio.

[37] Li Yao, Sherjil Ozair, Kyunghyun Cho, and Yoshua Bengio. On the equivalence between deep nade and generative stochastic networks. In *In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2014.

SCALABLE SEARCHING AND RANKING FOR MELODIC PATTERN QUERIES

Philippe Rigaux
Cedric Lab, CNAM
philippe.rigaux@cnam.fr

Nicolas Travers
Leonard de Vinci, Research Center
Cedric Lab, CNAM
nicolas.travers@devinci.fr

ABSTRACT

We present the design and implementation of a scalable search engine for large Digital Score Libraries. It covers the core features expected from an information retrieval system. Music representation is pre-processed, simplified and normalized. Collections are searched for scores that match a melodic pattern, results are ranked on their similarity with the pattern, and matching fragments are finally identified on the fly. Moreover, all these features are designed to be integrated in a standard search engine and thus benefit from the horizontal scalability of such systems. Our method is fully implemented, and relies on ELASTICSEARCH for collection indexing. We describe its main components, report and study its performances.

1. INTRODUCTION

We consider the problem of searching large collections of digital scores encoded in a symbolic format, typically MusicXML [14], MEI [22, 28], or the forthcoming format of the W3C Music Notation Group [23]. These encodings are now mature and stable, and we can expect to witness in the near future the emergence of very large Digital Score Libraries (DSL). A representative example of such endeavors is the OpenScore initiative, which aims at publishing high-quality encoding of public domain sheet music. This potentially represents millions of scores, and gives rise to strong needs in terms of collection management tools tailored to the peculiarities of music representation.

In the present paper, we focus on the *content-based retrieval* problem. We consider the search mechanism where a user submits a monophonic *query pattern* in order to retrieve, from a very large collection of scores, those that contain one or more fragments "similar" to this pattern. We further require the search system to be *scalable*.

With this objective in mind, we propose two main contributions. First, we expose the design of the core modules of a search engine, namely pre-processing and data normalization, pattern-based search, ranking, and on-line identification of fragments that match the pattern query. Second, we propose a list of guidelines for integrating these modules in a standard information retrieval system, with two main benefits: reduction of implementation efforts, and horizontal scalability.

Our approach is summarized by Figure 1. Pre-processing, matching, occurrence extraction and ranking are standard steps in text-based information retrieval systems, adapted here to the specificities of music representation. Tokenization, stemming and lemmatization [21] are, in our case, replaced by a so-called *normalization* that simplifies the representation and improves the robustness of the result. The *matching step* operates on the normalized representation (query pattern and score content). Normalization and matching are detailed in Section 3.

Obtaining a full score in the result would be of little use if we were not able to identify all the fragments that actually match the pattern, called *pattern occurrences*. This is necessary, for instance, to highlight them in the user interface. The algorithm is described in Section 4.

Finally, the set of matching scores are sorted according to the similarity of their occurrences to the pattern. While pattern matching mostly relies on the melodic profile, the ranking method focuses on the rhythm (Section 5). Their combination produces results with highly relevant scores.

The rest of the paper (Section 6) covers our second contribution, namely the integration of our music retrieval components in a standard search engine. For the sake of concreteness, we detail this integration with ELASTICSEARCH (https://elastic.co).

We finally position our work with respect to the state of the art and lists some useful extensions that could enrich the search functionalities (Section 7).

2. PRELIMINARIES: SCORES, FRAGMENTS, AND PATTERNS

Given a *melodic pattern* P as input, the *pattern matching operation* retrieves all the *scores* such that at least one *fragment* matches P. Our approach focuses on the pitch and duration features, generally considered as the most important parameters for melodic similarity [27]. We model a score as a synchronization of *voices*, and each voice as a sequence of elements $< e_1, e_2, \cdots, e_n >$ with e_i in $\mathcal{E} \times \mathcal{D}$, where $\mathcal{E}$ is the domain of musical "events" (notes, chords, rest) and $\mathcal{D}$ the musical duration.

Figure 1. Overview of the main indexing and matching steps

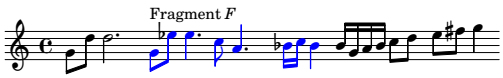


Figure 2. Voice V_I

Let us take the example of a voice V_I (Fig. 2). The blue fragment, denoted by F in the following, is used to illustrate the matching operation. V_I encodes a melody beginning with a G4 (semi-quarter), followed by a D5 (idem), a D5 (dotted half), etc. Using some pitches encoding mechanism, for instance the chromatic notation (number of semitones from the lowest possible sound), one obtains the representation of this voice as a sequence of pairs:
$V_I =< (34,8);(41,8);(41,3);(34,8);(42,8);(42,6);(39,8);\cdots >$

Given a voice V, we can derive other representations thanks to *transformation* functions. We consider two main categories of transformations: *simplifications* and *mutations*. They will be used in the normalization process.

Definition 1 (Simplifications) *A voice V can be transformed by the following simplification functions: $\epsilon(V)$, the sequence of pitches, $\pi(V)$ the sequence of pitch intervals between events in $\epsilon(V)$, and $\rho(V)$ the sequence of duration (without events) of V.*

Intuitively, $\epsilon(V)$ captures the melodic profile (sequence of note heights), $\pi(V)$ the relative evolution of pitches' heights, and $\rho(V)$ the rhythmic profile of a voice. Applied to V_I one obtains:

1. $\epsilon(V_I) =< 34,41,41,34,42,42,39,36,\ldots >$
2. $\pi(V_I) =< 7,0,-7,8,0,-3,-3,\ldots >$
3. $\rho(V_I) =< 8,8,3,8,8,6,8,6,\ldots >$

Definition 2 (Mutations) *A mutation $M_{I_1 \to I_2}$ maps an interval I_1 to another interval I_2. A* mutation family *is a set of mutation functions.*

For example, $\mathcal{M}_D = \{M_{1\to2}, M_{3\to4}, M_{8\to9}\}$ denotes a subset of the family of *diatonic mutation* that transforms a minor second, third or sixth in, respectively, their major counterpart, and conversely.

Finally, a *fragment* is any subsequence of a voice. We will use the word "pattern" to denote the fragment supplied by some user as a search criteria.

3. NORMALIZATION AND MATCHING

The matching operation is a Boolean procedure that tells whether the pattern P and a fragment F are similar to one another. This similarity concept is subject to a trade-off between the precision (relevant part of the result) and the recall (global relevant scores over the result). This is a traditional information retrieval issue. Let us examine how it is translated in the realm of symbolic music representation.

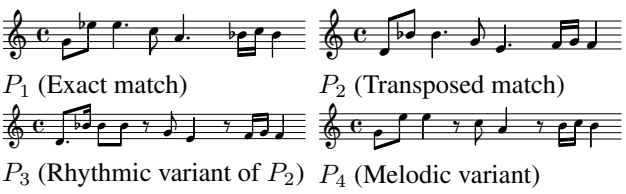


Figure 3. Several matching interpretations

3.1 Discussion

Fig. 3 shows several pattern variants, candidates to match with the fragment F of V_I (blue note heads in Fig. 2).

Exact Match. The strictest matching definition requires both the sequence of pitches (resp. $\epsilon(F)$ and $\epsilon(P)$) and the sequence of durations (resp. $\rho(F)$ and $\rho(P)$) to be identical. If we stick to this definition, F matches only with pattern P_1. The precision is then maximal but we will miss results that seem intuitive. F will not match for instance with the transposed pattern P_2, all other things being equal. This is probably too strict for most applications.

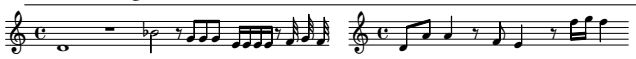
Transposed Match. Accepting transposition means that we ignore the absolute pitch and focus only on intervals, i.e., we compare $\pi()$ and $\rho()$, introducing flexibility in the melodic correspondence. In that case P_2 matches F.

Rhythmic Match. Next, consider pattern P_3 (Fig. 3), a rhythmic variant of P_2 that does not match F by exact rhythmic matching. Again, this definition seems too strict, since short rests, or slight duration adjustments, can typically be added or removed from a voice to denote a specific articulation, without severely affecting the music itself. P_3 matches F if we compare only $\pi(P_3)$ and $\pi(F)$, and ignore $\rho()$. Note that rhythmic changes involve not only rests and durations, but also repeated notes.

Melodic Match. Finally, P_4 is a pattern where intervals have been mutated. The initial minor sixth is replaced by a major sixth. Since such mutations can be found in imitative styles (e.g., counterpoint), it can make sense to accept them as part of the matching definition.

How far are we ready to go in the transformation process? Fig. 4 shows two extreme examples. Pattern P_5 matches F with respect to the sequence of intervals ($\pi(P_5) = \pi(F)$), whereas pattern P_6 is a rhythmic match ($\rho(P_5) = \rho(F)$). It seems clear that these patterns are quite far from the considered fragments and that, at the very least, they should not be given the same importance in the result set than the previous ones.

Music similarity has been studied for decades now. It seems obvious that there is no ideal solution that would suit all situations [12, 15] since similarity judgments depend on many aspects. However, our goal here is *not* to compute all the similarities, but to provide a filtering mechanism that gets rid of the scores that do not match the query pattern.



P_5 (Melodic matching) P_6 (Rhythmic matching)

Figure 4. Two examples that match F on one dimension

F and P_1 P_2, P_3, P_5

Figure 5. Voice normalization.

Algorithm 1 Voice normalization

```
1: procedure VNORM(V)
2: Input: A voice V
3: Output: A voice V', normalization of V
4:     V' ← V
5:     Normalize all note durations in V' to a quarter.
6:     Merge repeated notes from V'.
7:     Remove rests from V'
8:     return V'
```

This mechanism should be simple, efficient, and plugable in a standard search engine. It does not require to apply a costly similarity function to the whole collection.

In this perspective, matching-based retrieval is a first step operated to filter out a large part of the collection. A simplified similarity function can be used to top rank relevant scores. It is then easy to develop further investigations (*e.g.,* specialized similarity function) on the result set.

3.2 Normalization

It is generally considered that rhythm plays a prominent role in the perception of similarity. We therefore rank the result according to the rhythmic likeness of each retrieved fragment with the pattern. Filtering is based on the melodic profile, and its impact depends on how we simplify this profile in the normalization step. Two extreme choices are either to keep the exact sequence, increasing the precision, or to extract the melodic contour, increasing the recall.

In information retrieval systems, normalization is part of pre-processing steps, usually called *analyzers* and can be tuned by the administrator. This flexibility should be adopted for the symbolic music retrieval as well.

Our current implementation relies on the VNORM() algorithm (Alg. 1), applied to both the pattern and each voice in the corpus. Fig. 5 shows the voice normalization applied on patterns F, P_1 (top) and P_2, P_3, P_5 (bottom). In both cases the sequence of intervals obtained by $\pi()$ on the normalization is <6,-3,-3,1,2,-2>.

3.3 Matching

Definition 3 *A score S* matches *a pattern P iff, for* at least *a voice v in S, and* at least *a pair $[b, e], e > b$ of offsets (positions) in v, where the set of voice fragments $v[b] \cdots v[e]$ that match P are called the* matching occurrences *of S.*

$$\pi(\text{VNORM}(P)) = \pi(\text{VNORM}(v[b] \cdots v[e]))$$

4. FINDING MATCHING OCCURRENCES

Once matching scores have been extracted from the repository, it is necessary to identify the corresponding sequences of pitches that match the given query pattern (on normalized *n*-grams). For this, we need to look forward to exact match between intervals in the query pattern, and pitches in score's voices. Algorithm 2 produces a list of matching pitches which will be ranked in the next section.

Algorithm 2 Finding matching occurrences

```
1: procedure FINDINGOCCURRENCES(V, Q)
2: Input: A voice V, a query Q of intervals
3: Output: A set L of fragments
4:     for p in LCS(V, Q) do        ▷ List of matching patterns
5:         L ← L ∪ p
```

This procedure processes a voice V with a given query pattern Q. The `LCS` [1, 20] algorithm (*Longest Common Subsequence*) will give in output each matching pattern in V. It will verify if the pattern Q matches any interval between two successing pitches from V. The LCS algorithm iterates on each pitch of V to check each eligible subsequence, especially for matching patterns contained into repeating subfragments in Q.

5. RANKING

Given a set of fragments that match a pattern P, we now want to sort them according to a similarity measure, and put on top the ones that are closest to P. Assume that the search pattern is our previous F (Fig. 2, blue heads) and that the result set is $\{P_1, P_2, P_3, P_5\}$. For all, function $\pi()$ composed with the normalization VNORM yields <6,-3,-3,1,2,-2>. Intuitively P_1 and P_2 should be ranked first, and P_5 should be ranked last.

It is important to note that this ranking applies to fragments that have an identical melodic pattern. We take advantage of this specificity to operate at two levels. The first level measures the similarity of the "melodic rhythm", i.e., the respective duration of pairwise intervals in each fragment. To this end we define the notion of *blocks*.

Definition 4 *Let F be a fragment such that $\pi(\text{VNORM}(F)) = < I_1, \cdots, I_n >$. By definition of π and* VNORM*, each interval $I_j, j \in [1, n]$ is represented in F by a sequence $< p_1^i, e_2^i, \cdots, e_{k-1}^i, p_k^i >$ such that:*

- *p_1^i and p_k^i are two distinct pitches, and $interval(p_1^i, p_k^i) = I_i$*
- *each $e_l^i, l \in [2, k-1]$ is either a rest, or a pitch such that $e_l^i = p_1^i$*

We call $< p_1^i, e_2^i, \cdots, e_{k-1}^i >$ the block *B_i of I_i in F.*

A block is the largest subsequence of a fragment that covers a non-null interval. The concept of block is illustrated by Fig. 6 for P_1, P_3 and P_5.

The first level of the ranking function evaluates the similarity of two fragments F_1 and F_2 by comparing the blocks

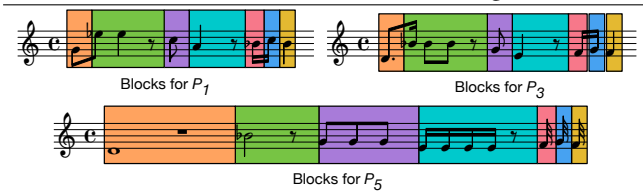


Figure 6. Blocks for P_1, P_3, and P_5

Algorithm 3 Ranking procedure

procedure RANKING(F_1, F_2)
Input: F_1, F_2, such that $\pi(\text{VNORM}(F_1))$=$\pi(\text{VNORM}(F_2))$
Output: a similarity $s \in [0, 1]$
  $s \leftarrow 0$; $d_1 \leftarrow duration(F_1)$; $d_2 \leftarrow duration(F_2)$
  for $i := 0$ **to** n **do** ▷ Loop on the blocks
    $s \leftarrow s + |dur(B_i^1)/d_1 - dur(B_i^2)/d_2|$
  if $s = 0$ **then** $s \leftarrow TieBreaking(F_1, F_2)$
  return $s/2$ ▷ Euclidian normalization [19]

pairwise durations. The rationale is that if these durations are similar, the only difference lies in either repeated notes or rests inside each block. Fig. 6 shows for instance that block durations in P_1 and P_3 are exactly the same, which makes them almost identical. The difference is internal to each block (for instance block 2, in green). On the other hand, P_1 and P_5 turn out to be quite dissimilar.

The RANKING function is simple and efficient (Algorithm 3). We first normalize the fragment duration, and sum up the difference of durations between pairs of blocks.

If it turns out that all block durations are pairwise identical, a tie-breaking function has to be called. This is the only situation where we might have to examine internal of blocks. By definition of blocks, this internal representation only consists of rhythmic data: rests and repeated notes. Any standard text comparison method (edit distance, Levhenstein distance) can be used.

6. INDEXING

We now describe how our functions can be integrated in a search engine. For the sake of concreteness, our description relies on ELASTICSEARCH, but the method works for any similar system (e.g., Solr) that uses inverted index.

6.1 Encoding

An index in ELASTICSEARCH is built on JSON documents. Each field in such a document can be either *indexed*, *stored* or both. Indexing a field means that ELASTICSEARCH supports full-text searches on the field's content. Storing a field means that the field's content is stored in the index. Our index features an *n*-gram field for searching, and a `sequence` field for the ranking.

Given a voice V and the sequence of intervals of its normalization $\pi(\text{VNORM}(V)) =< I_1,\cdots,I_k >$, we compute the list of n-grams $\{<I_i, \cdots, I_{i+n-1} >, i \in [1, k-n+1]\}$, where n, the n-gram size, is an index configuration parameter. If, for instance, the sequence of intervals is <6,-3,-3,1,2,-2>, the list of 3-grams is {<6,-3,-3>, <-3,-3,1>, <-3,1,2>, <1,2,-2>}.

Each n-gram is then encoded as a character string which constitutes a *token*. These tokens are finally concatenated in a large character string, separated by a white space. Positive integers are encoded with `a`, `b`, `c`, etc., and the minus sign by `m`, as illustrated in the following:

```
{"query": {"match_phrase":
  {"ngram": "fmcmc mcmca mcab abmb"} } }
```

6.2 Searching

We can then run keyword queries and, more importantly, *phrase queries* where ELASTICSEARCH retrieves the fields that contain a list of tokens that appear in a specific order. The previous query shows the"*match_phrase*" query which searches the *n*-gram sequence.

The search engine then does the rest of the job for us. It finds all the indexed documents such that the *n*-gram field contains the phrase. However, by default, ranking is based on textual features that do not match what we expect. We therefore need to replace the default ranking method.

6.3 Ranking

Ranking functions can be overridden in ELASTICSEARCH [32, 35]. To this end, we must provide a Java function that implements the ranking method exposed in Section 5. This function is called at query time and produces a similarity score for each voice. The result is sorted on this value, and made accessible to the client application via an iterator-like mechanism [11] called *SearchScript*.

Our ranking function operates on a voice to identify the matching occurrences, and to measure the similarity between the search pattern and each occurrence. We must store in ELASTICSEARCH an encoding of the voice that can be accessed during the query evaluation.

6.4 Query expansion

Our method relies on a strict matching of a sequence of intervals. Since accepting unbounded melodic transformations would likely return the whole database, we can consider those that can be seen as meaningful from a musical point of view. For instance, diatonic mutations of intervals (e.g., accepting both minor and major thirds or sixths in the matching operation) probably makes senses and can improve significantly the recall of our method.

We have integrated the synonym query expansion feature of ELASTICSEARCH engine (e.g., the ability to map "car" to "vehicle") to implement this feature. To achieve this, we decided to produce a list of synonyms for major thirds or sixths. These means that an interval '`c`' can be similar to '`d`' or interval '`h`' to '`i`'. The following *n*-grams are then considered to be: similar`cbh, dbh, dbi, cbi`

In order to integrate the list of synonyms to ELASTICSEARCH, it is given to the index as an analyzer *"melodic_transformation"* (illustrated in the query Section 6.5) and matching *n*-grams are merged at query time.

To find the matching occurrences, we need to modify Algorithm 2 in order to integrate the synonyms where intervals (in the `LCS` algorithm) for major third and sixth

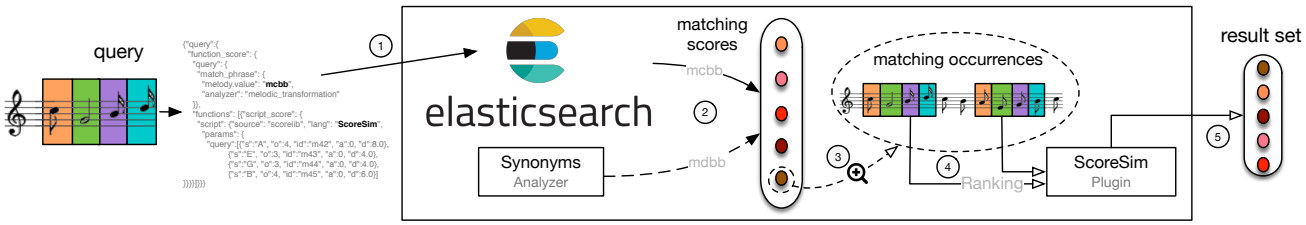


Figure 7. `ScoreSim` integration in ELASTICSEARCH

can be authorized, then: $1.5 \sim 2$ and $3 \sim 3.5$. ELASTICSEARCH can be further enhanced by taking into account a similarity measure between those synonyms (*e.g.*, oriented graph weighted with similarity values between synonyms).

6.5 Implementation

The integration of our approach in ELASTICSEARCH needs to preprocess musical scores to normalize them, and to compute the ranking in the search engine on query-time.

The first step consists in a Python scripts that normalizes voices from scores (Alg. 1), extracts *n*-grams, and produce JSON documents sent to the ELASTICSEARCH REST API. Documents also contain corpus and opus ids, syllables from lyrics voices which can be queried.

Second, to take into account synonyms in ELASTICSEARCH, the list of *n*-gram synonyms needs to be set offline. We have generated the list of all combinations of third and sixth transformation for each possible *n*-gram available in the repository. This list is imported in ELASTICSEARCH and then processed on-the-fly for each query.

The third step integrates the `ScoreSim` scoring module as a Java program in ELASTICSEARCH. For this, a *SearchScript* needs to be inherited in order to produce a plugin for ELASTICSEARCH. This plugin takes queries' parameters and instantiates a scoring function which will process every matching scores. The scoring function `ScoreSim` implements Algorithms 2 and 3.

The query below integrates all features that are proposed in our approach: *n*-gram search (*melody.value*), synonyms analyzer (*melodic_transformation*), and the `ScoreSim` function (*script_score*). In the latter, the parameter "*query*" gives the list of pitches used in order to produce the score value from Algorithm 3. The query params gives the sequence of the music items (octave, pitch and duration) that constitutes the pattern in order to provide distances and rank the result set.

```
{"query":{ "function_score": {
    "query": { "match_phrase": { "melody.value": "mcbb",
                    "analyzer": "melodic_transformation"}},
    "functions": [{"script_score": {
      "script": {"source": "scorelib", "lang": "ScoreSim",
        "params": {"query":[
            {"s":"A", "o":4, "id":"m42", "a":0, "d":8.0},
            {"s":"E", "o":3, "id":"m43", "a":0, "d":4.0},
            {"s":"G", "o":3, "id":"m44", "a":0, "d":4.0},
            {"s":"B", "o":4, "id":"m45", "a":0, "d":6.0}]}}}}]}}}
```

Figure 7 shows the querying process with the following steps: **1)** transform the melodic pattern into the ELASTICSEARCH DSL (Domain Specific Language), **2)** ELASTICSEARCH gets all the matching score corresponding to the given *n*-gram and eventually to their synonyms, **3)** instantiate the `ScoreSim` plugin and process every score, **4)** extract occurrences on each instance and then its score value, **5)** and finally, ELASTICSEARCH sorts the whole result-set according to the produced scores and sends the result.

	q1	q2	q3	q4
Querying pattern	`demc`	`mcmcmc`	`bmbb`	`bmbmc`
Nb of matching scores	204	719	877	2,225
Nb with query expansion	242	1867	877	2,236

Table 1. Query patterns

6.6 Performance

In order to study the impact of our approach on the computation time, we apply different queries on our corpora. The corpora size varies by cumulating several corpus, representing up to 4,950 polyphonic scores. It is composed of the whole corpora available here [1] composed of *Francœur, Méthodes, Motet, Psautiers, Sequentia, Timbres*, etc.

To study the effect of the matching process, we have chosen four different queries to apply with various patterns, from infrequent to more frequent ones, based on the popularity of the stored *n*-grams. Table 1 gives for each query pattern the corresponding total number of matches in the corpora and the number after applying the query expansion (synonyms). We can see that query q2 is clearly expanded since `mcmcmc` has 7 synonyms and produces a large number of matches (2.5 times more). At the opposite, query q3 has no synonyms and do not enlarge its result-set.

Figure 8 shows the evolution of the number of matching scores wrt the corpus size. It gives both matching scores for normal (plain lines) and expanded queries (dashed lines). According to the synonyms, we can see that q1 and q4 provide few more matchings, while q2 witnesses a different behavior where the number of matchings grows proportionally with the number of synonyms.

The execution time is plotted in Figure 9 for the 4 different queries. It shows both normal pattern queries (plain lines) and expanded queries (dashed lines). This allows to investigate both the robustness with respect to various results sizes, and issues related to false positives.

Each query is sub-linear in the result size. Query q4 is a frequent pattern which returns 2,225 matching scores (45% of the corpora). It is executed in 277 ms. The small number of synonyms has few impacts on the global processing time. At the opposite, q1 is extremely efficient due to its

[1] NEUMA repertory: `http://neuma.huma-num.fr/home/`

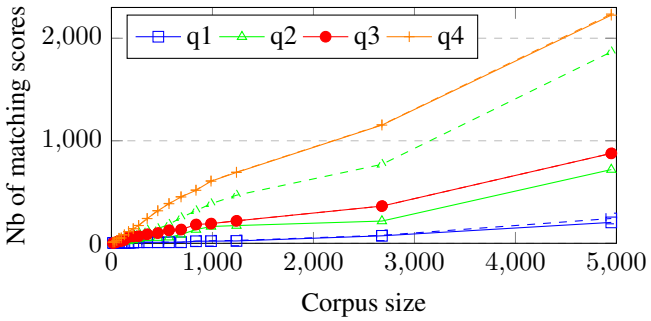


Figure 8. Nb of matching with and without synonyms

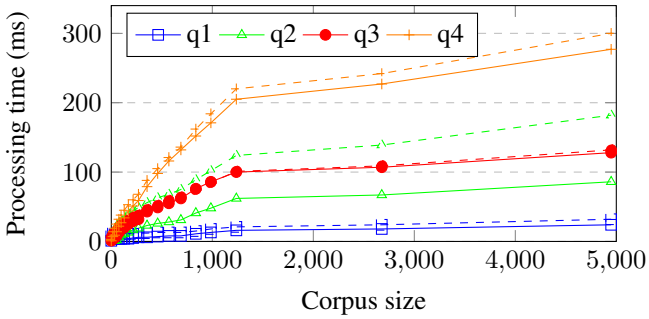


Figure 9. Execution time with and without synonyms

selectivity. It produces 204 scores in 22 ms. One interesting effect can be seen for query q2 where the number of matching synonyms leads to more computation time but it only 2.1 times more (for 2.5 times more matching scores).

Those experiments show that the ranking function takes less than 0.12 ms to process each score on a single ELASTICSEARCH node (server). The corpus can be spread on several nodes in order to scale it up horizontally.

7. RELATED WORK

Our approach combines similarity searches based on textual music encoding, scalable search, and rhythm-based ranking. The novelty of this approach is their association in a consistent setting. The rather trivial implementation makes, in our opinion, our solution quite attractive.

Music similarity has been an active MIR research topic over the last decades [5]. The general goal is to evaluate the likeness of two musical sequences. A major problem raised by this definition is that similarity judgments are highly dependent on both the content being compared and on the user taste, culture, and experience [12, 15]. A recent survey [34] summarizes the recent trends observed in the SMS track of the MIREX competition.

A similarity method is characterized by the choice of musical parameters. Pitches and durations are generally considered as expressive enough. Using sequences to represent both parameters is primarily motivated by our objective to integrate our methods in a standard search engine, and to benefit from an index structure. Some important parameters, e.g., metric accent, structure or harmonic are ignored because they often lead to tree-based encodings that are hardly indexable. Geometric approaches, such as [33], are also less suitable in this indexing perspective. Multidimensional structures are complex, and their performances are known to fall down as the dimension increases [29], and not yet integrated to off-the-shell search engines.

Textual encoding of symbolic music representation is an attractive idea in order to use text algorithms. The *HumDrum* toolkit [17] relies on a specialized text format and adapts Unix file inspection tools for music analysis. Exact and approximate string matching algorithms for melody matching have been used in *ThemeFinder* [18,30] or *Musipedia* [26]. Many algorithms for efficient computation of similarity matching through exhaustive searches have been proposed [2–4, 7]. Specialized rhythm similarity functions are proposed and compared in [31].

Text-based approaches are simple solutions, with two important limitations. First, combining pitches and rhythm in a single character string for instance is not easy, and small music variants may result in important syntactic differences. Second, these methods do not scale since the whole database has to be inspected for each query. Several indexing methods have been suggested for the edit distance [8, 24]. The *Dynamic Time Warping* distance is another popular method, for which sophisticated indexing structures have been proposed [13, 16]. None of them is available beyond research prototypes.

The easiest way to benefit from an inverted index is to split musical sequences in n-grams. This has been experimented in several earlier proposals [6, 9, 10, 25]. Each n-gram plays the role of a "token" and search methods apply.

Ranking is an essential part of an information retrieval system. We believe that our proposal, which combines **1)** a pre-processing normalization step, **2)** a melodic profile search and **3)** a rhythmic profile ranking, completes earlier attempts to adapt text-based retrieval to music retrieval, and results in a complete workflow which achieves a satisfying trade-off between the filtering impact, the ranking relevancy and the overall efficiency.

8. CONCLUSION

We described in this paper a practical approach to the problem of indexing pattern-based searches in a large score library. Our solution fulfills three major requirements for an information retrieval system: (i) it supports search with a significant part of flexibility, (ii) it proposes a ranking method consistent with the matching definition, and (iii) it brings scalability thanks to its compatibility with the features of state-of-the-art search engines. We fully implemented our solution, including the internal ranking function for ELASTICSEARCH, and we will be pleased to supply our software components to any interested institution that wishes to propose a content-based search mechanism.

Score analyzers extension. Alg. 1 with the normalization of voices can be extended with ad hoc music *analyzers*: management of grace notes, the simplification of melodic profiles, treatment of repeated notes, or cross-voice melodies, to name a few.

Acknowledgments. This work is funded by the French ANR project MUNIR

9. REFERENCES

[1] Lasse Bergroth, Harri Hakonen, and Timo Raita. A survey of longest common subsequence algorithms. In *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000*, pages 39–48, Sep. 2000.

[2] Emilios Cambouropoulos, Maxime Crochemore, Costas S. Iliopoulos, Manal Mohamed, and Marie-France Sagot. A Pattern Extraction Algorithm for Abstract Melodic Representations that Allow Partial Overlapping of Intervallic Categories. In *Proceeding of International Conference on Music Information Retrieval (ISMIR)*, pages 167–174, 2005.

[3] Domenico Cantone, Salvatore Cristofaro, and Simone Faro. Efficient Algorithms for the delta-Approximate String Matching Problem in Musical Sequences. In *Proc. of the Prague Stringology Conf. (Stringology)*, pages 33–47, 2004.

[4] Domenico Cantone, Salvatore Cristofaro, and Simone Faro. Solving the (δ, α)-Approximate Matching Problem Under Transposition Invariance in Musical Sequences. In *Proceeding of International Conference on Music Information Retrieval (ISMIR)*, pages 460–463, 2005.

[5] MA Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.

[6] Chuan-Wang Chang and Hewijin Christine Jiau. An Efficient Numeric Indexing Technique for Music Retrieval System. In *Proc. IEEE Intl Conf. on Multimedia and Expo (ICME)*, pages 1981–1984, 2006.

[7] Raphaël Clifford and Costas S. Iliopoulos. Approximate string matching for music analysis. *Software Computing*, 8(9):597–603, 2004.

[8] Camélia Constantin, Cédric du Mouza, Zoé Faget, and Philippe Rigaux. The melodic signature index for fast content-based retrieval of symbolic scores camelia constantin. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*, pages 363–368, 2011.

[9] Shyamala Doraisamy and Stefan M. Rüger. A Polyphonic Music Retrieval System Using N-Grams. In *Proceeding of International Conference on Music Information Retrieval (ISMIR)*, pages 204–209, 2004.

[10] J. Stephen Downie. *Evaluating a simple approach to music information retrieval: Conceiving melodic n-grams as text*. PhD thesis, Univ. Western Ontario, 1999.

[11] ElasticSearch. Advanced scripts using script engines, 2018. `https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-scripting-engine.html`.

[12] Jeff Ens, Bernhard E. Riecke, and Philippe Pasquier. The Significance of the Low Complexity Dimension in Music Similarity Judgements. In *Proceeding of International Conference on Music Information Retrieval (ISMIR)*, pages 31–38, 2017.

[13] Klaus Frieler, Frank Höger, Martin Pfleiderer, and Simon Dixon. Two web applications for exploring melodic patterns in jazz solos. In *Proceeding of International Conference on Music Information Retrieval (ISMIR)*, pages 777–783, 07 2018.

[14] Michael Good. *The Virtual Score: Representation, Retrieval, Restoration*, chapter "MusicXML for Notation and Analysis", pages 113–124. W. B. Hewlett and E. Selfridge-Field, MIT Press, 2001.

[15] Malo Cameron Jones, J. Stephen Downie, and Andreas F. Ehmann. Human Similarity Judgments: Implications for the Design of Formal Evaluations. In *Proceeding of International Conference on Music Information Retrieval (ISMIR)*, pages 539–542, 2007.

[16] Eamonn J. Keogh and Chotirat (Ann) Ratanamahatana. Exact Indexing of Dynamic Time Warping. *Knowl. Inf. Syst.*, 7(3):358–386, 2005.

[17] Ian Knopke. The Perlhumdrum and Perllilypond Toolkits for Symbolic Music Information Retrieval. In *Proceeding of International Conference on Music Information Retrieval (ISMIR)*, pages 147–152, 2008.

[18] Andreas Kornstaedt. Themefinder: A web-based melodic search tool. *Computing in Musicology*, 11:231–236, 1998.

[19] Günter Lettl and Peter M. Gruber. Isometries of the Space of Convex Bodies in Euclidean Space. *Bulletin of the London Mathematical Society*, 12(6):455–462, 1980.

[20] David Maier. The complexity of some problems on subsequences and supersequences. *Journal of ACM*, 25(2):322–336, April 1978.

[21] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. Online version at *http://informationretrieval.org/*.

[22] Music Encoding Initiative. `http://www.music-encoding.org`, 2015. Accessed April 2019.

[23] Music Notation Community Group. `https://www.w3.org/community/music-notation/`, 2018. Accessed April 2019.

[24] Gonzalo Navarro, Ricardo Baeza-yates, Erkki Sutinen, and Jorma Tarhio. Indexing Methods for Approximate String Matching. *IEEE Data Engineering Bulletin*, 24(4):19–27, 2001.

[25] Giovanna Neve and Nicola Orio. Indexing and Retrieval of Music Documents through Pattern Analysis and Data Fusion Techniques. In *Proceeding of International Conference on Music Information Retrieval (ISMIR)*, pages 216–223, 2004.

[26] Lutz Prechelt and Rainer Typke. An interface for melody input. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 8(2):133–149, 2001.

[27] Jon B. Prince. Contributions of pitch contour, tonality, rhythm, and meter to melodic similarity. *Journal of experimental psychology*, 40(6):2319–2337, 2014.

[28] Perry Rolland. The Music Encoding Initiative (MEI). In *Proceedings of the International Conference on Musical Applications Using XML*, pages 55–59, 2002.

[29] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2006.

[30] Craig Stuart, Yi-Wen Liu, and Eleanor Selfridge-Field. Search-effectiveness measures for symbolic music queries in very large databases, 2004.

[31] Godfried Toussaint. A comparison of rhythmic similarity measures. In *Proceeding of International Conference on Music Information Retrieval (ISMIR)*, pages 242–245, 2004.

[32] Nicolas Travers. *Web Data Management*, chapter Putting into Practice: Full-Text Indexing with LUCENE, pages 355–363. Cambridge University Press, 2012.

[33] Rainer Typke, Panos Giannopoulos, Remco C. Veltkamp, Frans Wiering, and René van Oostrum. Using transportation distances for measuring melodic similarity. In *Proceeding of International Conference on Music Information Retrieval (ISMIR)*, 2003.

[34] Valerio Velardo, Mauro Vallati, and Steven Jan. Symbolic melodic similarity: State of the art and future challenges. *Computer Music Journal*, 40:70–83, 2016.

[35] Cailan Zhou, Bin Feng, and Zhihao Li. Research and implementation of the small-scale search engine based on lucene. In *Proceedings of the 2008 International Conference on Computer Science and Software Engineering - Volume 02*, CSSE '08, pages 377–380. IEEE Computer Society, 2008.

ADAPTIVE TIME–FREQUENCY SCATTERING FOR PERIODIC MODULATION RECOGNITION IN MUSIC SIGNALS

Changhong Wang[1], Emmanouil Benetos[1], Vincent Lostanlen[2], Elaine Chew[3]
[1]Centre for Digital Music, Queen Mary University of London, UK
[2]Music and Audio Research Laboratory, New York University, NY, USA
[3]CNRS-UMR9912/STMS IRCAM, Paris, France
{changhong.wang,emmanouil.benetos}@qmul.ac.uk;
vincent.lostanlen@nyu.edu; elaine.chew@ircam.fr

ABSTRACT

Vibratos, tremolos, trills, and flutter-tongue are techniques frequently found in vocal and instrumental music. A common feature of these techniques is the periodic modulation in the time–frequency domain. We propose a representation based on time–frequency scattering to model the inter-class variability for fine discrimination of these periodic modulations. Time–frequency scattering is an instance of the scattering transform, an approach for building invariant, stable, and informative signal representations. The proposed representation is calculated around the wavelet subband of maximal acoustic energy, rather than over all the wavelet bands. To demonstrate the feasibility of this approach, we build a system that computes the representation as input to a machine learning classifier. Whereas previously published datasets for playing technique analysis focus primarily on techniques recorded in isolation, for ecological validity, we create a new dataset to evaluate the system. The dataset, named CBF-periDB, contains full-length expert performances on the Chinese bamboo flute that have been thoroughly annotated by the players themselves. We report F-measures of 99% for flutter-tongue, 82% for trill, 69% for vibrato, and 51% for tremolo detection, and provide explanatory visualisations of scattering coefficients for each of these techniques.

1. INTRODUCTION

Expressive performances of instrumental music or singing voice often abound with vibratos, tremolos, trills, and flutter-tongue. A common feature of these four playing techniques is that they all result in some periodic modulation in the time–frequency domain. However, from a musical standpoint, these techniques convey distinct stylistic effects. Discriminating between these spectrotemporal patterns requires a compact and informative representation that remains stable to time shifts, time warps, and frequency transpositions. Time–frequency scattering [2] provides such mathematical guarantees. Besides the local invariance to translation and stability to deformation provided by the scattering transform [1, 10], time–frequency scattering goes further by applying frequential scattering along the log-frequency axis. This operation provides invariance to frequency transposition and captures regularity along log-frequency dimension.

Prior work in the representation of vibrato and tremolo can be divided into three broad categories: F_0-based representations [4, 14, 20], template-based techniques [5], and modulation spectra [13, 15, 17]. The error-prone stage of fundamental frequency estimation hinders the performance of F_0-based methods. Template-based methods may work for vibratos with a large modulation extent (frequency variation), while for subtly-modulated vibratos, both the definition of templates and the matching between templates and test segments are problematic. The modulation spectra is another well-known representation for modulated sounds, which is averaged on the audio clip level [17]. It may work well for long-term music information retrieval tasks such as genre classification or instrument recognition, but struggling with providing temporal positions for short-duration playing technique recognition. To our knowledge, there is not yet any computational research that compares and discriminates between these periodic modulations in real-world music pieces.

Besides the question of coming up with an adequate signal representation, there is a critical need for human-annotated playing techniques in audio recordings. Up to now, most of the available research literature has focused on playing techniques that have been recorded in highly controlled environments [8, 16, 19]. Yet, recent findings demonstrate that, in the context of a music piece, playing techniques exhibit considerable variations as compared to when they are played in isolation [18]. For periodic modulations, these variations are more evident in folk music, which highly depends on the interpretation of the performer. Such inter-performer variability in folk music performance necessitates data collection with full pieces.

This paper includes three contributions: representation, application, and dataset. We propose a representation based on time–frequency scattering to model the inter-

class variability for fine discrimination of vibrato, tremolo, trill, and flutter-tongue. Rather than decomposing all the wavelet bands as the scattering transform, we calculate a time–frequency scattering around the wavelet subband of maximal acoustic energy, i.e. the transform is calculated adaptively on the predominant frequency. On the application side, to our knowledge this is the first attempt at creating a system for detecting and classifying periodic modulations in music signals. To evaluate our methodology, we create a dedicated dataset of the Chinese bamboo flute, also known as the *dizi* or *zhudi*, and thereafter abbreviated as CBF. This dataset, named CBF-periDB, contains full-length solo performances recorded by professional CBF players and has been thoroughly annotated by the players themselves.

The rest of this paper is organised as follows. The characteristics of each periodic modulation and how this information can be represented by an adaptive time–frequency scattering are described in Section 2. Section 3 shows details of the feature extraction process and the proposed recognition system. The dataset, evaluation methodology, and results are discussed in Section 4. Section 5 presents our conclusions and directions for future research.

2. SCATTERING REPRESENTATION OF PERIODIC MODULATIONS

Prior to discriminating between the four periodic modulations, we analyse characteristics of each modulation in Section 2.1. A short introduction of the scattering transform is provided in Section 2.2. Section 2.3 describes the proposed representation for modelling periodic modulations.

2.1 Characteristic Statistics of Periodic Modulations

The characteristic statistics of each modulation in discussion are shown in Table 1. As can be seen, flutter tonguing has a much higher modulation rate as compared to the other three modulations; thus, the modulation rate can be used as a main feature to distinguish it from others. For the other three techniques with similar modulation rate, the discriminative information lies in the modulation extent and shape of the modulation unit. The *modulation unit* refers to the unit pattern that repeats periodically within the modulation. It can be one-dimensional, either amplitude modulation (AM) or frequency modulation (FM), or two-dimensional as a spectro-temporal modulation. This can be intuitively observed from the partially enlarged spectrograms given in Fig. 1. Trills are note-level modulations, for which the frequency variations are larger than one semitone. This extent of modulation is much larger than vibratos and tremolos. The shape of the modulation unit for trill is more square-like rather than sinusoidal ones which vibratos and tremolos exhibit. The difference between vibrato and tremolo is that vibratos are FMs, while tremolos are AMs. We show later how this discriminative information is encoded into the proposed representation in Section 2.3.

Type	**Rate** (Hz)	**Extent**	**Shape**
Flutter-tongue	25-50	< 1 semitone	Sawtooth-like
Vibrato	3-10	< 1 semitone	Sinusoidal (FM)
Tremolo	3-8	$\approx$ 0 semitone	Sinusoidal (AM)
Trill	3-10	Note level	Square-like

Table 1. Characteristic statistics of four periodic modulations in music signals.

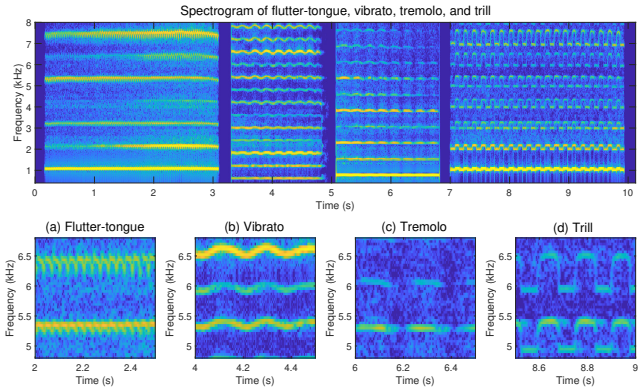


Figure 1. Visual comparison of four periodic modulations. Top: Spectrogram of flutter-tongue, vibrato, tremolo, and trill; bottom: partially enlarged spectrogram of each modulation for detailed comparison.

2.2 Scattering Transform

Proposed by [10], the scattering transform is a cascade of wavelet transforms and nonlinearities. Its structure is similar to a deep convolutional network. The difference is that its weights are not learnt but can be hand-crafted to encode prior knowledge about the task at hand. Since little energy is captured by the scattering transform with orders higher than two [2], we focus in this paper to the second order.

Let ψ_λ denote the wavelet filter bank obtained from a mother wavelet ψ, where λ is the centre frequency of each wavelet in the filter bank. Likewise, ψ_{λ_m} refers to the wavelet filter bank of the m^{th}-order scattering transform, for $m \geq 1$. The second-order temporal scattering transform of a time-domain signal x is defined as:

$$\left| \left| x \overset{\text{t}}{*} \psi_{\lambda_1} \right| \overset{\text{t}}{*} \psi_{\lambda_2} \right| \overset{\text{t}}{*} \phi_T, \tag{1}$$

where $\overset{\text{t}}{*}$ is the wavelet convolution along time. $|x \overset{\text{t}}{*} \psi_{\lambda_m}|$ is the modulus of the m^{th}-order wavelet transform, which hereafter we refer to as the m^{th}-order wavelet modulus transform. The temporal scattering coefficients are obtained by an averaging at a time scale T by means of a low-pass filter ϕ_T.

In addition to the invariance to time-shifts and time-warps provided by the scattering transform, time–frequency scattering provides frequency transposition invariance by adding a wavelet transform along log-frequency axis [7]. The specific time–frequency scattering we apply is a *separable* scattering proposed in [3]. Here,

separable refers to separate steps of temporal and frequential operations of wavelet scattering, arranged in a cascade. The separable scattering representation comprises a second-order temporal scattering and a first-order frequential scattering. The latter is calculated by another wavelet transform along the log-frequency dimension on top of the second-order temporal scattering:

$$\left|\left|\left|x \overset{t}{*} \psi_{\lambda_1}\right| \overset{t}{*} \psi_{\lambda_2}\right| \overset{t}{*} \phi_T \overset{fr}{*} \psi_{\gamma_1}\right| \overset{fr}{*} \phi_F. \quad (2)$$

where $\overset{fr}{*}$ is the wavelet convolution along log-frequency. ψ_{γ_1} is the wavelet filter bank applied in the first-order frequential scattering. The frequential scattering coefficients are obtained by an averaging of the frequential wavelet modulus transform with transposition invariance of F (in octave unit) using a low-pass filter ϕ_F. All scattering coefficients in this paper are normalised and have their logarithm calculated, to capture only the temporal structure and to motivate auditory perception [2]. Hereafter, we use Morlet wavelets throughout the whole scattering network for wavelet convolutions. This is because Morlet wavelets have an exactly null average while reaching a quasi-optimal tradeoff in time–frequency localisation [9]. Our source code is based on the ScatNet toolbox [1].

2.3 Scattering Representation of Periodic Modulations

Periodic modulation recognition, as suggested by the analysis above, is a pitch invariant task. The core discriminative information is based on the modulation itself, which is indicated by its modulation rate, extent, and shape. Fig. 2 shows respectively (a) the spectrogram, (b) the second-order temporal scattering representation, and (c) the first-order frequential scattering representation of a series of periodic modulation examples in CBF-periDB. The spectrogram used here is only for illustration purposes. The first four examples are regular cases (modulations based on stable pitch or with constant parameters): vibrato, tremolo, trill, and flutter-tongue. The last three are cases with time-varying parameters (modulations based on time-varying pitch or with time-varying rate, extent, or shape): rate-changing trill, rate- and extent-changing trill, and flutter-tongue with time-varying pitch. We use these examples to show how the characteristic information of each pattern is captured and discriminated by a separable scattering transform, which consists of a second-order temporal scattering and a first-order frequential scattering transform.

2.3.1 Second-order temporal scattering

Different from a standard two-order temporal scattering transform, we do not decompose all the frequency bands (exchangeable with wavelet bands) in the first-order wavelet modulus transform. As can be observed from Fig. 2 (a), the patterns of these modulations, either in the regular case or time-varying case are similar for each harmonic partial. This indicates that the decomposition of one partial is sufficient to capture modulation information. Fig. 2 (b) shows the second-order temporal scattering representation decomposed only from the frequency band with the highest energy. Flutter-tongue is the most discriminable one with the highest modulation rate. For the other three patterns with close modulation rate value, other characteristic information is considered. By dominant band decomposition, the trill can also be discriminated because of its large modulation extent. This can be interpreted by filters with bandwidth larger than one semitone, which blurs other subtle modulations.

To specifically detect vibrato or tremolo, frequency bands less than one semitone should be obtained. We then make use of their modulation shape information by introducing a *band-expanding* technique. Assume we have frequency bands of 1/16 octave bandwidth in the first-order wavelet modulus transform. Ideally for tremolo, the modulation information is contained only in the dominant frequency band since it is an AM. This is verified by the second example in Fig. 2 (b), which has almost only the fundamental modulation rate with no upper harmonics in the second-order temporal scattering representation. However, vibratos are FMs, which means the modulation information spreads over neighbouring frequency bands. Decomposing neighbouring frequency bands above or below the dominant band provides additional information to distinguish vibrato from tremolo. All this discriminative information can be visualised from the fundamental modulation rate and the richness of the harmonics in the second-order temporal scattering representation in Fig. 2 (b).

2.3.2 First-order frequential scattering

The temporal scattering transform is sensitive to attacks and amplitude modulations, which results in the high en-

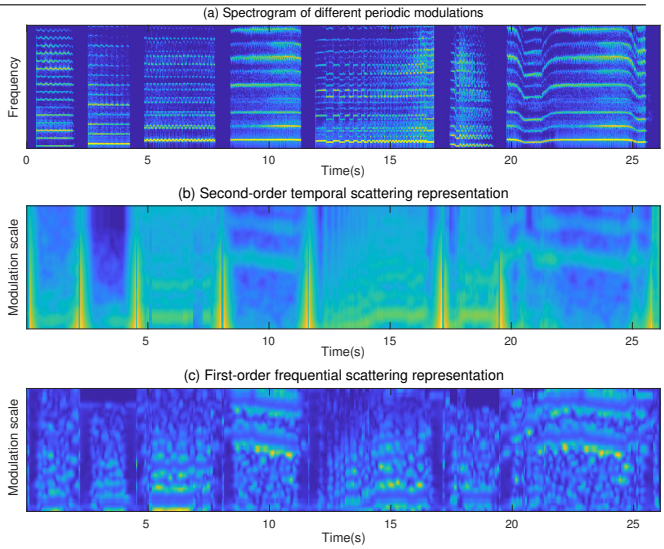


Figure 2. Separable scattering representation of different periodic modulations. From left to right, the first four are regular cases: vibrato, tremolo, trill, flutter-tongue based on stable pitches; the last three are time-varying cases: rate-changing trill, rate- and extent-changing trill, flutter-tongue with time-varying pitch.

[1] https://www.di.ens.fr/data/software/scatnet/

ergy part of the boundaries as clearly observed in Fig. 2 (b). To suppress this noisy information while retaining the frequential structure offered by the second-order temporal scattering, we use frequential scattering along the log-frequency axis on top of Fig. 2 (b). Frequential scattering has a similar framework as temporal scattering while the former captures regularity along log-frequency. As shown in Fig. 2 (c), we obtain a clearer representation without reducing the discriminative information necessary for the task. Although the last example is flutter-tongue bounded to time-varying pitch, its modulation rate is relatively stable. This verifies our method of using just the dominant frequency band or expanded frequency bands from the first-order wavelet modulus transform. The rate-changing and rate-extent changing cases show that the time-varying modulation rates are also captured.

3. PERIODIC MODULATION RECOGNITION

With the proposed representation prepared, we build a recognition system consisting of four binary classification schemes that each predicts one modulation type. Section 3.1 describes the feature extraction process. The calculated features are then fed to a machine learning classifier illustrated in Section 3.2.

3.1 Feature Input

As described in Section 2.3, we adapt the scattering transform by decomposing the dominant and its neighboring frequency bands in the first-order wavelet modulus transform. The feature extraction process of dominant band decomposition is shown in Fig. 3. Using a waveform as input, we first obtain the first-order wavelet modulus transform, where the frequency band with the highest energy for each time frame is localised. Decomposing these bands, we obtain a second-order temporal scattering. A first-order frequential scattering is then conducted on top of the second-order temporal scattering coefficients. Concatenating the two representations in a frame-wise manner, we obtain the feature input to the classifiers. For expanded band decomposition, additional features are calculated similarly by decomposing the neighbouring frequency bands around the dominant band.

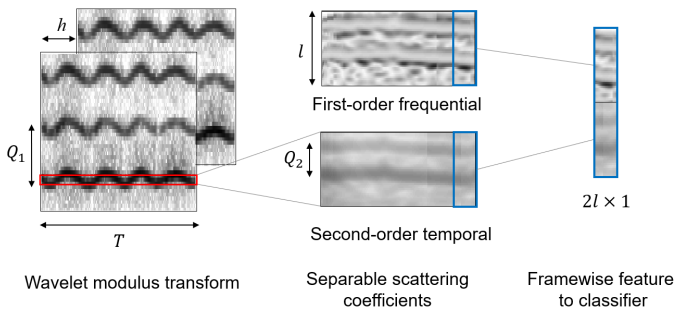


Figure 3. Feature extraction process.

Table 2 gives the parameters which encode the core discriminative information for the recognition. T is the averaging scale for the temporal scattering coefficients. This parameter is useful for discriminating modulations with large differences on modulation rate, for example, on distinguishing flutter-tongue from other low-rate periodic modulations. Averaging scales covering at least four unit patterns are recommended for reliable estimation of the modulation rate. Q_1 are the filters per octave in the first-order temporal scattering transform. Since the modulations discussed here are all oscillatory patterns, setting Q_1 should ensure that each of the modulations are not blurred in the first-order wavelet modulus transform. Here, we use $Q_1 > 12$ for the first-order temporal scattering to support subtly-modulated vibratos and tremolos, of which the modulation extent is less that one semitone. N is the number of neighbouring frequency bands besides the dominant band decomposed from the first-order wavelet modulus transform. $N = 0$ refers to dominant band decomposition only while $N > 0$ means expanded band decomposition. This is a key parameter to encode the unit shape information of subtle modulations. However, if the task at hand is only to detect modulations with high modulation rate or with large extent, this parameter is not necessary.

Parameter	Notation	Main information encoded
Averaging scale	T	Modulation rate
Filters per octave	Q_1	Modulation extent
Expanded bands	N	$= 0$, temporal shape
		> 0, spectro-temporal shape

Table 2. Parameters encoding modulation information in the adaptive time–frequency scattering framework.

Other parameters involved in the feature calculation include frame-size h, filters per octave Q_2 in the second-order scattering decomposition, frequency bands l extracted from the second-order temporal scattering. For frequential scattering, we apply a single scale wavelet transform. The frame size is inversely log-proportional to the oversampling parameter α by $h = T/2^{\alpha}$ (samples), which is designed to compensate for the low temporal resolution resulting from the large averaging scales. Since these parameters carry little discriminative information, we set them consistently for all classification schemes, with $\alpha = 4$, $Q_2 = 8$, and $l = 2Q_2$ based on experimental results. A general example with $T = 2^{15}$ corresponds to frame size of $h = T/2^{\alpha} = 2048$ samples (46ms, assuming the sampling rate is 44.1kHz). The dimensionality of the final representation at each time frame equals to $(N + 1) \times 2l = 4(N + 1)Q_2$.

3.2 Recognition System

Due to the existence of combined playing techniques, such as the combination of flutter-tongue and vibrato, and the combination of tremolo with trill, one frame of the input may have multiple labels. Multi-label classification is considered beyond the scope of this paper and is regarded as future work. Here, we conduct binary classifications for

each modulation, which enables us to explicitly encode the characteristic information specifically for the corresponding pattern. Four binary classifiers are constructed using support vector machines (SVMs) with Gaussian kernels. The model parameters to be optimized in the training process are the error penalty parameter and the width of the Gaussian kernel [6]. The best parameters selected in the validation stage are used for testing. The input feature to the classifiers is the proposed adaptive time–frequency transform of the current time frame.

Taking flutter-tongue as an example, its relatively high modulation rate (25-50Hz) can be emphasized by setting $T = 8192$ (sampling rate is 44.1kHz). The modulation extent which is less than one semitone is interpreted by setting $Q_1 = 16$. Fig. 4 shows the detection result of flutter-tongue from a piece in CBF-periDB using dominant band decomposition ($N = 0$), which can be clearly observed from the harmonic structure in Fig. 4 (a). This is then enforced by removing the noisy attacks using a frequential scattering transform as shown in Fig. 4 (b). Concatenating the two representations, we form frame-wise separable scattering feature vectors with $(N + 1) \times 2l = 32$ dimensions and frame size of $h = T/2^{\alpha} = 512$ samples (12ms). Fig. 4 (c) visualises the binary classification result of flutter-tongue compared with the ground truth for an example excerpt. Overall it can be seen that the proposed approach is successful at detecting flutter-tongue, even in short segments, although occasionally the output is over-fragmented. Similarly, binary classifiers can be implemented for detecting vibratos, tremolos, and trills, with parameters fine-tuned to the corresponding modulations.

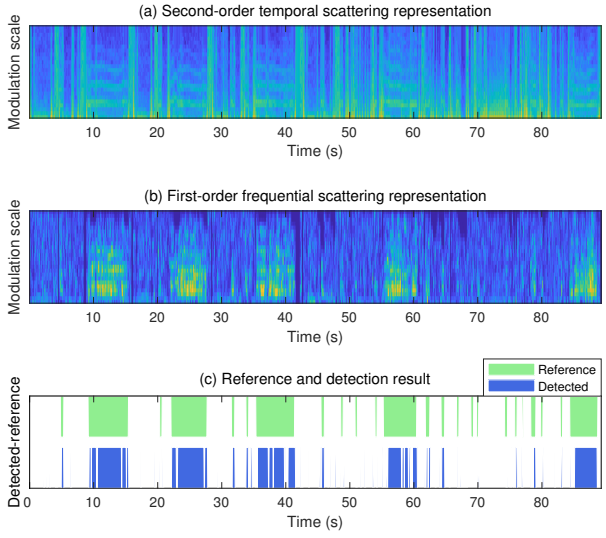


Figure 4. Binary classification result of flutter-tongue in an example excerpt of the CBF-periDB.

4. EVALUATION

4.1 Dataset

To verify the proposed system, we focus on folk music recordings which have more inter-performer variations than Western music. The proposed periodic modulation analysis dataset, CBF-periDB, comprises monophonic performances recorded by ten professional CBF players from the China Conservatory of Music. All data is recorded in a professional recording studio using a Zoom H6 recorder at 44.1kHz/24-bits. Each of the ten players performs both isolated periodic modulations covering all notes on the CBF and two full-length pieces selected from *Busy Delivering Harvest* «扬鞭催马运粮忙», *Jolly Meeting* «喜相逢», *Morning* «早晨», and *Flying Partridge* «鹧鸪飞». Players are grouped by flute type (C and G, the most representative types for Southern and Northern styles, respectively) and each player uses their own flute. This dataset is an extension of the CBF-glissDB dataset in [18], with ten pieces containing periodic modulations added. The playing techniques are thoroughly annotated by the players themselves. Details of both isolated techniques and full-piece (performed) recordings are shown in Table 3. The dataset and annotations can be downloaded from `c4dm.eecs.qmul.ac.uk/CBFdataset.html`.

Isolated		**Performed**	
Type	Length	Piece, number	Length
Flutter-tongue	4.9	Mo, 3	16.0
Vibrato	7.3	BH, 7	28.0
Tremolo	5.0	JM, 4	12.4
Trill	12.3	FP, 6	51.9

Table 3. Length of both isolated techniques and full-piece recordings in CBF-periDB (Mo=*Morning*; JM=*Jolly Meeting*; BH=*Busy Delivering Harvest*; FP=*Flying Partridge*; all numbers for length are measured in minutes).

In the recognition implementation process, the dataset is split into a 6:2:2 ratio according to players (players are randomly initialised) and a 5-fold cross-validation is conducted. This way of data splitting ensures a non-overlap between players in the train, validation, and test sets. Since each player uses their own flute during recording, there is no overlap across flutes. Due to the limited piece types, non-overlap of pieces is not feasible at the current stage. We regard this as future work with a dataset expansion plan to address piece diversity.

4.2 Metrics

Due to the short duration and periodic nature of vibratos, tremolos, trills, and flutter-tongue, feature dependencies over sequential frames are slightly required. The precision $\mathcal{P} = \frac{TP}{TP+FP}$, recall $\mathcal{R} = \frac{TP}{TP+FN}$, and F-measure $\mathcal{F} = \frac{2\mathcal{P}\mathcal{R}}{\mathcal{P}+\mathcal{R}}$ are used here for frame-based evaluation, where TP, FP, FN are true positives, false positives, and false negatives respectively [12]. Assigned labels by SVMs are then compared to the ground truth annotations in a frame-wise manner.

4.3 Baseline

According to our knowledge, there is not yet any previous work on discriminating between these four periodic mod-

Type	Dominant band						Expanded band					
	Temporal scattering			Separable scattering			Temporal scattering			Separable scattering		
	$\mathcal{P}(\%)$	$\mathcal{R}(\%)$	$\mathcal{F}(\%)$	$\mathcal{P}(\%)$	$\mathcal{R}(\%)$	$\mathcal{F}(\%)$	$\mathcal{P}(\%)$	$\mathcal{R}(\%)$	$\mathcal{F}(\%)$	$\mathcal{P}(\%)$	$\mathcal{R}(\%)$	$\mathcal{F}(\%)$
Flutter-tongue	96.1	99.8	97.9	96.3	99.8	98.0	96.7	99.6	98.1	97.8	99.5	**98.7**
Trills	87.1	66.7	75.1	87.4	68.2	76.2	89.5	73.3	80.4	89.8	76.3	**82.3**
Vibrato	75.2	17.5	26.4	72.2	33.1	45.3	75.9	59.4	66.5	75.1	64.7	**69.3**
Tremolo	92.5	1.21	2.2	80.9	6.7	10.6	70.8	38.5	49.1	67.6	41.4	**50.7**

Table 4. Performance comparison of binary classification for flutter-tongue, vibratos, tremolos, and trills in CBF-periDB using separable scattering and temporal scattering representations based on the dominant frequency band and expanded frequency band decomposition ($\mathcal{P}$=precision; $\mathcal{R}$=recall; $\mathcal{F}$=F-measure).

ulations; thus we compare the proposed systems against a state-of-art detection method for vibrato. The filter diagonalisation method (FDM), which efficiently extracts high resolution spectral information for short time signals, was first applied to vibrato detection in erhu performance [20]. Based on the high similarity of the music style between erhu and CBF, both being traditional Chinese instruments, we use FDM as a baseline method for vibrato detection. Using automatically estimated fundamental frequency by pYIN [11] as input for FDM, we try different parameter ranges for vibrato rate and extent based on the vibrato characteristics of the CBF. The best result we obtain based on a 256ms-frame-wise evaluation is $\mathcal{P}$=36.5%, $\mathcal{R}$=58.7%, $\mathcal{F}$=45.0%. The rate range and extent range we use are 3-10 Hz and 5-20 cent, respectively.

4.4 Results

In order to explicitly show information captured in each classification task, a small set of parameter settings for T, Q_1, and N is used. Besides the different meta-parameter settings specifically designed for each modulation classification, we run further experiments by using the same parameters for all four binary classification processes: $T = 2^{15}$, $Q_1 = 16$, and $N = 6$ frequency bands symmetrically expanded around the dominant band. This corresponds to frame size of 46ms and feature dimension of 224. Note that for specific modulation detection, meta parameters of the scattering transform can be fine-tuned to have a much lower feature dimension, as the flutter-tongue detection example demonstrated in Section 3.2. The binary classification results for each pattern are given in both the dominant band decomposition and expanded band decomposition, as shown in Table 4. The detection results using the second-order temporal scattering coefficients only as feature are also provided. The comparison between temporal scattering and separable scattering verifies our analysis in Section 2.3 that for periodic modulation recognition, frequential scattering in the separable scattering representations provides additional discriminative information by removing noisy information. Better performance for flutter-tongue and trill detection shows that for modulations with high modulation rate and large extent, decomposition of the dominant band is sufficient. For discriminating between temporal modulation and spectro-temporal modulation, expanded band decomposition works much better.

Generally, detection performance on vibrato and tremolo is worse than that for flutter-tongue and trill detection. Identifying the errors in the original audio, we find that in most cases these are combined techniques, i.e. subtle frequency variations are accompanied with amplitude modulations or vice versa. In the case of CBF, such combinations are common because of the instrumental gestures of vibrato and tremolo. Vibrato can be generated by fingering or tonguing, while tremolos are commonly produced by breathing variations. Performers are also expressively intended to add tremolo effects on top of other playing techniques.

5. CONCLUSIONS AND FUTURE WORK

Periodic modulation recognition is a pitch invariant task and should only capture modulation information. This is realised by calculating a time–frequency scattering around the wavelet subband of the maximum acoustic energy. We found that the proposed representation decomposed from the dominant band is sufficient to detect modulations with high modulation rate (flutter-tongue) and large modulation extent (trill), while expanded band decomposition captures subtle frequency modulations (vibrato and tremolo). We introduce the ecologically valid dataset, CBF-periDB, to evalate the recognition system. Results show that the proposed representation captures discriminative information between vibratos, tremolos, trills, and flutter-tongue in real-world pieces.

The current work only considers continuous periodic modulations; other periodic patterns, such as tonguing, will be considered as future work due to their noncontinuous nature and more complicated parameter variations. Inspired by research on polyphonic music transcription, current work may be expanded to polyphonic periodic modulation detection. Additional comparison of the current result with other equivalent representations such as the modulation spectra, will be conducted. We conclude that the scattering transform presents a versatile and compact representation for analysing periodic modulations in performed music and opens up new avenues for computational research on playing techniques.

6. ACKNOWLEDGEMENTS

Changhong Wang is funded by the China Scholarship Council (CSC). Emmanouil Benetos is supported by a RAEng Research Fellowship RF/128. Elaine Chew is supported by the European Union's Horizon 2020 research and innovation program (grant agreement No 788960) under the European Research Council (ERC) Advanced Grant (ADG) project COSMOS.

7. REFERENCES

[1] J. Andén and S. Mallat. Scattering representation of modulated sounds. *15th International Conference on Digital Audio Effects (DAFx)*, 2012.

[2] J. Andén and S. Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, 2014.

[3] C. Baugé, M. Lagrange, J. Andén, and S. Mallat. Representing environmental sounds using the separable scattering transform. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8667–8671, 2013.

[4] Y. P. Chen, L. Su, and Y. H. Yang. Electric guitar playing technique detection in real-world recording based on F0 sequence pattern recognition. In *Proc. Conf. International Society for Music Information Retrieval (ISMIR)*, pages 708–714, 2015.

[5] J. Driedger, S. Balke, S. Ewert, and M. Müller. Template-based vibrato analysis in music signals. In *Proc. Conf. International Society for Music Information Retrieval (ISMIR)*, pages 239–245, 2016.

[6] T. Hastie, R. Tibshirani, and J. Friedman. The elements of statistical learning: data mining, inference, and prediction, 2009.

[7] V. Lostanlen. *Convolutional operators in the time-frequency domain*. PhD thesis, PSL Research University, 2017.

[8] V. Lostanlen, J. Andén, and M. Lagrange. Extended playing techniques: the next milestone in musical instrument recognition. In *5th International Conference on Digital Libraries for Musicology (DLfM)*, 2018.

[9] S. Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way.* Academic Press, 2008.

[10] S. Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

[11] M. Mauch and S. Dixon. pYIN: A fundamental frequency estimator using probabilistic threshold distributions. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 659–663, 2014.

[12] M. Müller. *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015.

[13] Y. Panagakis, C. Kotropoulos, and G. R. Arce. Music genre classification via sparse representations of auditory temporal modulations. In *17th IEEE European Signal Processing Conference (Eusipco)*, pages 1–5, 2009.

[14] R. Panda, R. Malheiro, and R. Paiva. Novel audio features for music emotion recognition. *IEEE Transactions on Affective Computing*, (1):1–1, 2018.

[15] B. L. Sturm and P. Noorzad. On automatic music genre recognition by sparse representation classification using auditory temporal modulations. *Proc. Computer Music Modeling and Retrieval (CMMR)*, 2012.

[16] L. Su, H. M. Lin, and Y. H. Yang. Sparse modeling of magnitude and phase-derived spectra for playing technique classification. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 22(12):2122–2132, 2014.

[17] S. Sukittanon, L. E. Atlas, and J. W. Pitton. Modulation-scale analysis for content identification. *IEEE Transactions on Signal Processing*, 52(10):3023–3035, 2004.

[18] C. Wang, E. Benetos, X. Meng, and E. Chew. Hmm-based glissando detection for recordings of chinese bamboo flute. In *Proc. Conf. Sound and Music Computing (SMC)*, pages 545–550, May 2019.

[19] J. Wilkins, P. Seetharaman, A. Wahl, and B. Pardo. Vocalset: A singing voice dataset. In *Proc. Conf. International Society for Music Information Retrieval (ISMIR)*, pages 468–474, 2018.

[20] L. Yang, K. Rajab, and E. Chew. The filter diagonalisation method for music signal analysis: frame-wise vibrato detection and estimation. *Journal of Mathematics and Music*, 11(1):42–60, 2017.

CONTROLLING SYMBOLIC MUSIC GENERATION BASED ON CONCEPT LEARNING FROM DOMAIN KNOWLEDGE

Taketo Akama
Sony Computer Science Laboratories, Tokyo, Japan
taketo.akama@sony.com

ABSTRACT

Machine learning allows automatic construction of generative models for music. However, they are learned from only the succession of notes itself without explicitly employing domain knowledge of musical concepts such as rhythm, contour, and fragmentation & consolidation. We approximate such musical domain knowledge as a function, and feed it into our model. Then, two decoupled spaces are learned: the *extraction space* that captures the target concept, and the *residual space* that captures the remainder. For monophonic symbolic music, our model exhibits high decoupling/modeling performance. Controllability in generation is improved: (i) our *interpolation* enables concept-aware flexible control over blending two musical fragments, and (ii) our *variation generation* enables users to make concept-aware adjustable variations.

1. INTRODUCTION

Listeners not only perceive the succession of notes itself, but also respond to higher-level concepts in music. Two critical components in melodic perception and memory are scale and contour [5]. It is said that similarity between musical fragments is important in listeners' emotional arousal responses to music [18]. Listeners sense those similarities through perceiving patterns of music constructs or transformations such as rhythm, interval, and fragmentation & consolidation (F&C) [20].

Music data processing, especially music generation and analysis have attracted much attention. One of the major methods exploits models that learn the *latent space* [1, 6, 9, 16, 27–29]. These models learn compressed but informative feature vectors of data samples and distribute them in the multi-dimensional latent space. The spacial arrangement represents the relation of data samples. Also, numerous intermediate features—corresponding to samples hopefully not in the dataset—are yielded to fill in the "holes" in the latent space. Then generation and analysis are performed by bidirectional mapping of the latent space and the data space. The latent space is, however, learned from raw musical data without supervision. Therefore, the musical notions or concepts that are important for people are not sufficiently organized on the latent space. Hereinafter, we refer to such notions or concepts as *musical concepts*, examples of which include rhythm, contour, and F&C, as mentioned above.

How do we organize those musical concepts on the latent space? People possess domain knowledge about musical concepts, although even the major concepts are not necessarily defined clearly. In fact, various musical concepts can be approximated as a *function* of raw musical sequences. We input such domain knowledge to our model in the form of a function, and then our model learns latent spaces that capture the corresponding musical concepts.

Our model is called ExtRes (Extraction-Residual Latent Space Decoupling Model), and it aims at learning decoupled latent spaces, each of which is associated with a musical concept. In other words, each musical concept is learned as a *latent-space concept* that occupies a part of the dimensions in the multi-dimensional latent space. The concept-wise decoupled latent spaces allow us to measure similarity between musical fragments in terms of each concept. The similarity is then used for e.g., pattern discovery in a musical piece [17]. In generation, the control over concept-wise latent features helps us to create musical phrases as imagined or to compose patterns/structures in a musical piece [23].

Our ExtRes has the following characteristics. (I) **Knowledge based**: musical concepts can be incorporated as function approximations on the basis of domain knowledge. For monophonic musical sequences, various important musical concepts can be incorporated such as rhythm, chromatic/diatonic interval or pitch, step-leap/signed contour, and F&C. This is possible because given an input sequence, these concepts can be approximated as other sequences with rules or algorithms [2, 19] to construct the functions. (II) **Complex concept and active learning**: complex concepts are actively learned without obtaining attributes first [3, 4, 12, 14], where a set of attributes is a concept. (III) **Extraction-residual**: aiming at comprehending complex data by repeatedly analyzing "one concept versus the rest." The rest (residual) may capture the useful concepts (e.g., scale for contour and "pitch order" for rhythm). (IV) **Coexisting latent spaces**: depending on the input domain knowledge, corresponding concepts are captured in latent spaces. Users can exploit multiple models of ours—wherein latent spaces with different concepts *coexist*—for handling many concepts.

We apply our ExtRes to improving controllability in interactive music generation. (i) **Concept-axes interpolation**: this mechanism helps users to create musical phrases as imagined. Users first input two musical fragments into the system. Prior methods allow users to adjust the *blending* ratio of the two fragments uniformly regardless of concepts [27, 28], whose blending is more musically meaningful than the naive data space blending. Our concept-axes interpolation offers more flexibility: enabling users to blend only the factor of the desired concept in musical fragments and to also adjust the blending ratio for each target/residual concept. (ii) **Concept-aware variation generation**: given a musical fragment, this mechanism allows users to obtain variations, where the amount of variation for each concept is adjustable. When generating a long structured piece of music, this mechanism helps to faithfully realize either of the instructions of a song template [23] or the user intention.

2. METHODOLOGY

2.1 Outline

We propose ExtRes, a generative model that allows learning reusable representation (as latent features and embedding vectors) for a user-specified concept, given a function based on domain knowledge on the concept (Sec.2.2). We then instantiate our proposed ExtRes model for sequence datasets (Sec.2.4). In Sec.2.5, musical concepts are approximated as functions on the basis of domain knowledge. For applications of our ExtRes, we focus on meaning-level controllability in generation. We consider the following kinds of control: (i) altering in relation to other samples (e.g., interpolation), (ii) altering a sample to another sample with similar but not the same meaning (e.g., variation), and (iii) altering individual concepts. ExtRes not only puts (iii) into practice by free explorations in concept spaces, but also allows hybridizing the meaning-level controls (i-iii) for more intended controllability: Sec.2.3 Concept-Axes Interpolation is for bridging (i) and (iii), Sec.2.3 Concept-Aware Variation Generation is for bridging (ii) and (iii).

2.2 Extraction-Residual Latent Space Decoupling Model (ExtRes)

Let us consider a dataset $\mathcal{D} = \{x^{(n)}\}_{n=1}^{N}$, consisting of N i.i.d. samples of stochastic variable $x \in \mathcal{X}$. Given $f_{ext}\colon \mathcal{X} \to \mathcal{Y}$ that extracts information of a target concept as $y \in \mathcal{Y}$ from x, ExtRes is for learning latent spaces $\mathcal{Z}_e$ and $\mathcal{Z}_r$ that correspond to f_{ext}. $\mathcal{Z}_e$ is called an *extraction space*, which captures the extracted target concept, and $\mathcal{Z}_r$ is called a *residual space*, which is expected to be decoupled from $\mathcal{Z}_e$, aiming at capturing a concept corresponding to all the rest. Figure 1 summarizes our model.

f_{ext} can be obtained e.g., by constructing rules or algorithms on the basis of the data domain knowledge (for musical sequences, see [2, 19]). Specific examples based on musical domain knowledge are shown in Sec.2.5.

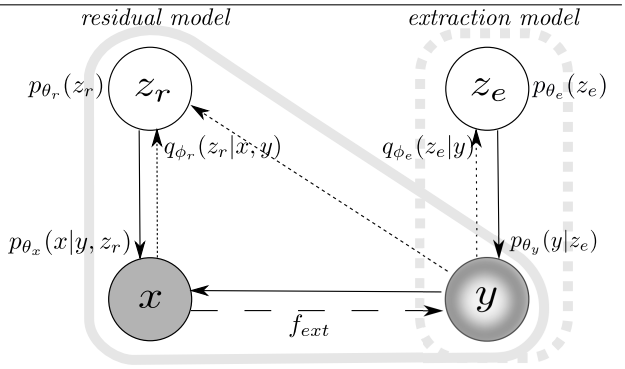


Figure 1: Graphical model of ExtRes.

First, we conduct *data derivation*: augmenting the dataset $\mathcal{D}$ to obtain $\mathcal{D}_{drv} = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$, through the mapping $y = f_{ext}(x)$. Then, our approach is to learn a generative model involving two latent variables: $z_e \in \mathcal{Z}_e$ for capturing variability in y and $z_r \in \mathcal{Z}_r$ for variability in x given y. We assume the dataset is generated from the following process: (i) $z_e^{(n)} \sim p_{\theta_e^*}(z_e)$, $z_r^{(n)} \sim p_{\theta_r^*}(z_r)$, (ii) $y^{(n)} \sim p_{\theta_y^*}(y|z_e^{(n)})$, and (iii) $x^{(n)} \sim p_{\theta_x^*}(x|y^{(n)}, z_r^{(n)})$. Then we model this generative process by maximizing marginal log likelihood $\log p_\theta(\mathcal{D}_{drv}) = \sum_{n=1}^{N} \log p_\theta(x^{(n)}, y^{(n)})$ with each term rewritten as:

$$\log p_\theta(x,y) = \log \int p_{\theta_x}(x|y,z_r) p_{\theta_r}(z_r) dz_r \,, \\ + \log \int p_{\theta_y}(y|z_e) p_{\theta_e}(z_e) dz_e. \quad (1)$$

Since this is computationally intractable in general, we derive an evidence lower bound (ELBO) [16]:

$$\log p_\theta(x,y) \geq \mathcal{L}_{res} + \mathcal{L}_{ext}, \quad (2)$$

$$\text{where } \mathcal{L}_{res} = \mathbb{E}_{q_{\phi_r}(z_r|x,y)}\left[\log p_{\theta_x}(x|y,z_r)\right] \\ - D_{KL}\left(q_{\phi_r}(z_r|x,y)||p_{\theta_r}(z_r)\right), \quad (3)$$

$$\text{and } \mathcal{L}_{ext} = \mathbb{E}_{q_{\phi_e}(z_e|y)}\left[\log p_{\theta_y}(y|z_e)\right] \\ - D_{KL}\left(q_{\phi_e}(z_e|y)||p_{\theta_e}(z_e)\right). \quad (4)$$

Here, D_{KL} denotes the Kullback-Leibler (KL) divergence.

We maximize the data likelihood of right hand side of Eq.(2) for optimizing parameters $\{\theta_x, \theta_y, \theta_r, \theta_e, \phi_r, \phi_e\}$. We refer to the models corresponding to Eq.(3) and Eq.(4) as the *residual model* and *extraction model*, respectively.

2.3 Controlling Generation

Concept-Axes Interpolation. With our decoupled latent spaces, interpolation can be done for each latent space $\mathcal{Z}_e$ and $\mathcal{Z}_r$. In the simplest linear case, interpolation between two latent vectors $[z_e^{(i)}; z_r^{(i)}]$ and $[z_e^{(j)}; z_r^{(j)}]$ produces samples $x(\alpha_e, \alpha_r) \sim p_{\theta_x}(x|y(\alpha_e), z_r(\alpha_r))$ with $y(\alpha_e) \sim p_{\theta_y}(y|z_e(\alpha_e))$, where $z_r(\alpha_r) = z_r^{(i)} + \alpha_r(z_r^{(j)} - z_r^{(i)})$ and $z_e(\alpha_e) = z_e^{(i)} + \alpha_e(z_e^{(j)} - z_e^{(i)})$ with $(\alpha_r, \alpha_e) \in [0,1] \times [0,1]$.

Variation Generation Approach. Finding the *boundary*, in a latent space, between *variations* and *non-variations* is semi-automatically learned by defining that *variations* of a given data sample are the samples that contain enough information in terms of reconstruction error ϵ,

in expectation. For our ExtRes, the error ϵ can be adjusted by introducing trade-off parameters in Eq.(3) and Eq.(4) (see Sec.4.2). Intuitively, the latent vectors capture *high-level* features for the corresponding data samples, and given a data sample $\hat{x}$, the learned inference distributions $q_{\phi_r}(z_r|\hat{x}, f_{ext}(\hat{x}))$ and $q_{\phi_e}(z_e|f_{ext}(\hat{x}))$ specify feature-wise similar/dissimilar (i.e. *variations*/*non-variations*) *boundaries* of the given sample. Therefore, our approach is to generate variations $x^{(i)}$ of $\hat{x}$ in accordance with the *boundaries* as follows: $x^{(i)} \sim p_{\theta_x}(x|y^{(i)}, z_r^{(i)})$ with $y^{(i)} \sim p_{\theta_y}(y|z_e^{(i)})$ and $z_r^{(i)} \sim q_{\phi_r}(z_r|\hat{x}, f_{ext}(\hat{x}))$, where $z_e^{(i)} \sim q_{\phi_e}(z_e|f_{ext}(\hat{x}))$.

Concept-Aware Variation Generation. We also propose how to generate variations $x^{(i)}$ in a concept-wise manner when the inference models are normal distributions. The amount of variation is controlled for each concept in either of the following ways: simply changing the ratio of the covariance scales of inference distributions, or ordering the samples $z_e^{(i)}$ or $z_r^{(i)}$ on the basis of Mahalanobis distances:

$$
\begin{aligned}
&D_M(z^{(i)}, \hat{z}) = \sqrt{(z^{(i)} - \hat{z})^{\mathrm{T}} \Sigma^{-1} (z^{(i)} - \hat{z})}, \qquad (5)\\
&\text{for } (z^{(i)}, \hat{z}, \Sigma) \in \{(z_e^{(i)}, \hat{z}_e, \Sigma_e), (z_r^{(i)}, \hat{z}_r, \Sigma_r)\},\\
&\quad \text{where} \quad q_{\phi_e}(z_e|f_{ext}(\hat{x})) = \mathcal{N}(z_e|\hat{z}_e, \Sigma_e),\\
&\quad \text{and} \quad q_{\phi_r}(z_r|\hat{x}, f_{ext}(\hat{x})) = \mathcal{N}(z_r|\hat{z}_r, \Sigma_r).
\end{aligned}
$$

Here, $\mathcal{N}$ denotes normal distribution.

2.4 Instantiation of ExtRes for Sequences

Throughout the rest of this paper, we consider the case where x consists of a sequence of discrete variables s_t i.e. $x = (s_1, ..., s_T)$. Here, each s_t has a distribution over the elements of a finite alphabet set $\mathcal{A}$. Let f_{ext} be a function that maps a sequence of length T to that of length T, i.e. $f_{ext}\colon \mathcal{A}^T \rightarrow \mathcal{B}^T$, where $\mathcal{B}$ is another alphabet set. First, we conduct *data derivation* using $f_{ext}(x^{(n)}) = y^{(n)} = (a_1^{(n)}, ..., a_T^{(n)})$. The given dataset $\mathcal{D} = \{x^{(n)}\}_{n=1}^N$ can be augmented to become $\mathcal{D}_{drv} = \{(x^{(n)}, y^{(n)})\}_{n=1}^N$, where $x^{(n)} = (s_1^{(n)}, ..., s_T^{(n)})$ and $y^{(n)} = (a_1^{(n)}, ..., a_T^{(n)})$. We refer to $(a_1^{(n)}, ..., a_T^{(n)})$ as an *abstract sequence* of $(s_1^{(n)}, ..., s_T^{(n)})$. Now, the inference models are

$$
\begin{aligned}
&(h_{z_e,t}, c_{z_e,t}) = \mathrm{LSTM}(\mathrm{E}(a_t), h_{z_e,t-1}, c_{z_e,t-1}), \qquad (6)\\
&q_{\phi_e}(z_e|a_{1:T})\\
&\quad = \mathcal{N}(z_e|\mathrm{MLP}(h_{z_e,T}), \mathrm{diag}(\exp(\mathrm{MLP}(h_{z_e,T})))), \qquad (7)\\
&(h_{z_r,t}, c_{z_r,t})\\
&\quad = \mathrm{LSTM}([\mathrm{E}(s_t); \mathrm{E}(a_t)], h_{z_r,t-1}, c_{z_r,t-1}), \qquad (8)\\
&q_{\phi_r}(z_r|s_{1:T}, a_{1:T})\\
&\quad = \mathcal{N}(z_r|\mathrm{MLP}(h_{z_r,T}), \mathrm{diag}(\exp(\mathrm{MLP}(h_{z_r,T})))), \qquad (9)
\end{aligned}
$$

where LSTM, E, and MLP denote a long short-term memory RNN (first, second, and third arguments of LSTM are input, hidden state, and cell state, respectively) [13], embedding layer, and multi-layer perceptron, respectively. The generative model for the *abstract sequence* is

$$
\begin{aligned}
&p_{\theta_e}(z_e) = \mathcal{N}(z_e|\mathbf{0}, \mathcal{I}), \qquad (10)\\
&(h_{a,1}, c_{a,1}) = \mathrm{LSTM}([Ea_0; z_e], \mathrm{MLP}(z_e), c_{a,0}), \qquad (11)\\
&(h_{a,t}, c_{a,t}) = \mathrm{LSTM}([\mathrm{E}(a_{t-1}); z_e], h_{a,t-1}, c_{a,t-1}), \qquad (12)\\
&p_{\theta_y}(a_t|a_{1:t-1}, z_e) = \mathrm{Cat}(a_t|\sigma(\mathrm{MLP}(h_{a,t}))), \qquad (13)\\
&p_{\theta_y}(a_{1:T}|z_e) = \textstyle\prod_{t=1}^T p_{\theta_y}(a_t|a_{1:t-1}, z_e), \qquad (14)
\end{aligned}
$$

where Cat and σ denote the categorical distribution and softmax function, respectively. Note that we use notation $p_{\theta_y}(a_1|a_{1:0}, z_e) = p_{\theta_y}(a_1|z_e)$ for brevity. The generative model for the original sequence is

$$
\begin{aligned}
&p_{\theta_r}(z_r) = \mathcal{N}(z_r|\mathbf{0}, \mathcal{I}), \qquad (15)\\
&(h_1, c_1) = \mathrm{LSTM}([Es_0; \mathrm{E}(a_1); z_r], \mathrm{MLP}(z_r), c_0), \qquad (16)\\
&(h_t, c_t) = \mathrm{LSTM}([\mathrm{E}(s_{t-1}); \mathrm{E}(a_t); z_r], h_{t-1}, c_{t-1}), \qquad (17)\\
&p_{\theta_x}(s_t|s_{1:t-1}, a_{1:t}, z_r) = \mathrm{Cat}(s_t|\sigma(\mathrm{MLP}(h_t))), \qquad (18)\\
&p_{\theta_x}(s_{1:T}|a_{1:T}, z_r) = \textstyle\prod_{t=1}^T p_{\theta_x}(s_t|s_{1:t-1}, a_{1:t}, z_r), \qquad (19)
\end{aligned}
$$

where we use the following notation for brevity: $p_{\theta_x}(s_1|s_{1:0}, a_{1:1}, z_r) = p_{\theta_x}(s_1|a_1, z_r)$.

2.5 Formulating Musical Domain Knowledge

On the basis of musical domain knowledge, we approximate musical concepts as $f_{ext}(x)$ for monophonic sequences. As mentioned in Sec.1, given an input musical sequence, many *abstract sequences* expressing important musical concepts can be derived [2, 19]. Among these, we demonstrate formulating two examples: *rhythm* and *contour*. Our finding is that a concept of musical transformations—fragmentation & consolidation, which is the main characteristic of similarity in musical sequences [20]—can also be approximated as a function, and we demonstrate it. We use the real name of musical notes as symbols ('C3', 'D#4', etc.) and use 'R' to represent a rest symbol. We add an extra symbol '__' representing that a note is held and not replayed [11]. Then the alphabet set $\mathcal{A}$ becomes a set of symbols listed above. Let $\mathcal{O}$ be a set of symbols that has no pitch, i.e., $\mathcal{O} = \{\text{'__'}, \text{'R'}\}$. We use the notation $\mathcal{S}_{1:t-1} = \{s_1^{(n)}, ..., s_{t-1}^{(n)}\}$.

Rhythm. Let 'N' denote a symbol that represents any note. Then the *rhythm* sequence for $(s_1^{(n)}, ..., s_T^{(n)})$ is defined as $(a_1^{(n)}, ..., a_T^{(n)})$, where

$$
a_t^{(n)} = \begin{cases} s_t^{(n)} & (s_t^{(n)} \in \mathcal{O}) \\ \text{'N'} & (\text{otherwise}) \end{cases}
$$

Thus, $\mathcal{B} = \mathcal{O} \cup \{\text{'N'}\}$. Now we define f_{ext} for *rhythm* as: $f_{ext}(x) = (a_1, ..., a_T)$.

Contour. In this paper, we refer to a chromatic signed contour with rhythm information as *contour*. Formally, the *contour* sequence for $(s_1^{(n)}, ..., s_T^{(n)})$ can be defined as $(a_1^{(n)}, ..., a_T^{(n)})$, where

$$
a_t^{(n)} = \begin{cases} s_t^{(n)} & (s_t^{(n)} \in \mathcal{O}) \\ \text{'SP'} & (s_t^{(n)} \notin \mathcal{O} \wedge \mathcal{S}_{1:t-1} \subset \mathcal{O}) \\ \mathrm{sgn}(\mathrm{Interval}(s_t^{(n)})) & (\text{otherwise}) \end{cases}
$$

Here 'SP' stands for Starting Pitch, sgn is a real sign function, and Interval is a function that calculates the chromatic pitch difference between the symbol of its argument and the previous symbol that has pitch. Then $\mathrm{sgn}(\mathrm{Interval}(s_t^{(n)}))$ outputs whether the pitch of $s_t^{(n)}$ is higher than ('1'), lower than ('-1'), or the same as ('0') the last symbol that has pitch. Thus, the alphabet set $\mathcal{B}$={'__', 'R', 'SP', '1', '-1', '0'}. Now we define f_{ext}

for *contour* as: $f_{ext}(x) = (a_1, ..., a_T)$.

Fragmentation and Consolidation (F&C). Fragmentation involves replacing one long note with several shorter notes, whereas consolidation conversely involves replacing several shorter notes with a single long note [20]. Formally, the *F&C*-invariant sequence for $(s_1^{(n)}, ..., s_T^{(n)})$ can be defined as $(a_1^{(n)}, ..., a_T^{(n)})$, where

$$a_t^{(n)} = \begin{cases} \text{'FC'} & (s_t^{(n)} \in \mathcal{O} \vee s_t^{(n)} = \text{LSP}(s_t^{(n)})) \\ s_t^{(n)} & (\text{otherwise}) \end{cases}$$

Here LSP stands for Last Symbol with Pitch, and $\text{LSP}(s_t^{(n)}) = s_{t_l}^{(n)}$, where $t_l = \max\{u\colon s_u^{(n)} \in \mathcal{S}_{1:t-1} \wedge s_u^{(n)} \notin \mathcal{O}\}$. Thus, the alphabet set $\mathcal{B} = \{\text{'FC'}\} \cup \mathcal{A} \setminus \mathcal{O}$. Now we define f_{ext} for *F&C* as: $f_{ext}(x) = (a_1, ..., a_T)$.

3. RELATED WORK

Conditional Models. Conditional deep generative models are widely studied especially in the image domain [15,22]. Although our *residual model* is similar to this family of models, their methods are different from ours in that (i) the problem setting itself is different: data for conditioning variable y is given in their method, (ii) condition y is not as structured as ours (usually labels), and (iii) the latent space of condition $\mathcal{Z}_e$ is not learned (i.e., their method has no *extraction model* of ours). In the music domain, conditions of notes or chords are used for factoring out those information from latent variables [7,28,29].

Disentangled Latent Spaces. In the image domain, some approaches successfully disentangle latent space [3,4,14]. The popular approach is regularizing each latent dimension to be independent, hoping to obtain interpretable factors as attributes. Meanwhile, an approach that permits disentangling a latent space applicable to symbolic music has been proposed [10], although this method assumes an attribute has order.

Exploring Latent Space in Music. After learning the latent space, some methods attempt to discover a meaningful direction in a latent space [8, 27]. These methods are also useful for exploring within our individual concept spaces.

Variation Generation in Music. The differences between our *concept-aware variation generation* and the variation mechanism proposed by Pachet et al. [23] are (i) their method is based on a Markov model; (ii) their method controls variation generation in terms of edit operations, while our method controls it in terms of musical concepts; (iii) our ExtRes tries to learn the notion of *variation* (see Sec.2.3). Difference (iii) also differentiates our method from the previous VAE method for generating melody variation, wherein simply Gaussian noises are added to the latent vector to create perturbed latent vectors [29], although their method could produce more diverse variations. One can also use their method within our concept spaces.

Regularizing Latent Space in Music. Human dissimilarity ratings on timbre are utilized to regularize the latent space for bridging audio analysis, perception, and generation [9].

4. EXPERIMENTS

4.1 Dataset

We conduct experiments using a leadsheet dataset introduced by Pachet et al. [24] with more than 12,000 songs, by hundreds of famous songwriters, covering several genres of popular music: jazz, blues, pop, and rock. The dataset has been used in music generation studies [21, 23, 25,26]. We extract all monophonic melody parts with time signature 4/4, which are transposed in all possible keys if the transposition remains within the midi pitch range of $[55, 84]$. We choose to discretize time with 24 symbols in a bar, where every beat has six symbols whose note-on timings in one beat are $\{0, 1/4, 1/3, 1/2, 2/3, 3/4\}$. We then extract all consecutive subsequences of length $T = 24$ (1 bar) or $T = 96$ (4 bars). The total dataset is split into proportion of $\{0.85, 0.1, 0.05\}$ for train, validation, and test data respectively.

4.2 Implementation Details

The numbers of dimensions for z_e or z_r are chosen to be $(16, 32)$ for the $T = (24, 96)$ model. 2-layer stacked LSTMs are employed. We introduce trade-off parameters β_1 for Eq.(3) and β_2 for Eq.(4) to weight the second terms [9, 27, 28]. Intuitively, the amount of information required for each latent variable depends on the target concepts: *rhythm*, *contour*, and *F&C*. Therefore, we conduct a hyper-parameter search using the validation dataset such that β_1 and β_2 are the maximum subject to reconstruction accuracies being sufficiently high. Then, β_1 and β_2 for *rhythm*, *contour*, and *F&C* are chosen to be $(\beta_1, \beta_2) = (0.7, 0.7)$, $(1.0, 0.7)$, and $(0.9, 0.7)$, respectively. The number of training epochs is set to 20, KL-annealing is used [1,9], and teacher forcing is not used.

4.3 Model Performance

Decoupling Performance. We first sample sequences $\{x^{(i)}\colon i \in \{1, ..., I\}\}$ and $\{x^{(i,j)}\colon (i, j) \in \{1, ..., I\} \times \{1, ..., J\}\}$, following the sampling procedure:

$$x^{(i)} \sim p_{\theta_x}(x|y^{(i)}, z_r^{(i)}),\ x^{(i,j)} \sim p_{\theta_x}(x|y^{(i)}, z_r^{(j)}),$$
$$\text{with } z_r^{(i)} \sim p_{\theta_r}(z_r),\ z_r^{(j)} \sim p_{\theta_r}(z_r), \text{ and}$$
$$y^{(i)} \sim p_{\theta_y}(y|z_e^{(i)}), \text{ where } z_e^{(i)} \sim p_{\theta_e}(z_e).$$

For $x^{(i)}$ and $x^{(i,j)}$, we count

$$N_{i,j} = |\{t \in \{1, ..., T\}\colon f_{ext}(x^{(i)})_t = f_{ext}(x^{(i,j)})_t\}|, \quad (20)$$

which is how many elements of the same index are the identical symbol between *abstract sequences* of $x^{(i)}$ and $x^{(i,j)}$. Then, we define the accuracy for a pair $(x^{(i)}, x^{(i,j)})$ as $N_{i,j}/T$.

Figure 2 illustrates the results of cumulative decoupling accuracies for $I = 1000$ and $J = 100$. *F&C* accuracies are almost perfect. Even for *rhythm* and *contour*, $(T = 24, T = 96) = (99.6, 97.5)\%$ and $(85.0, 37.6)\%$ of the samples have perfect accuracies, respectively. Interestingly, for *rhythm* and *contour*, the cumulative percentage of samples grows sharply if one symbol mistake

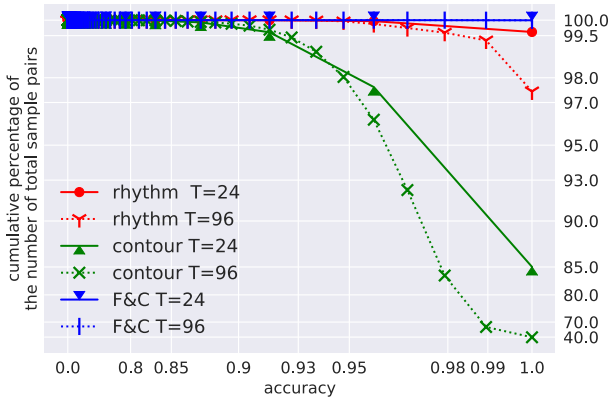


Figure 2: Decoupling accuracy.

	NLL		Accuracy	
	T=24	T=96	T=24	T=96
VAE (β=1.0)	0.714	0.577	0.880	0.880
VAE (β=0.8)	0.730	0.586	0.935	0.913
VAE (β=0.7)	0.753	0.606	0.952	0.923
VAE (β=0.5)	0.792	0.660	0.974	0.956
VAE (β=0.3)	0.856	0.748	0.988	0.960
Ours, *Rhythm*	0.412	0.322	0.967	0.969
	0.335	0.274	0.982	0.976
Ours, *Contour*	0.297	0.212	0.968	0.968
	0.468	0.405	0.973	0.938
Ours, *F&C*	0.141	0.130	0.978	0.975
	0.630	0.497	0.963	0.950

Table 1: Negative log-likelihood and reconstruction accuracy. For our models, upper and lower rows denote the *residual model* and *extraction model*, respectively.

is allowed. For instance, the percentage for *contour* grows from $(85.0, 37.6)\%$ to $(97.6, 66.3)\%$.

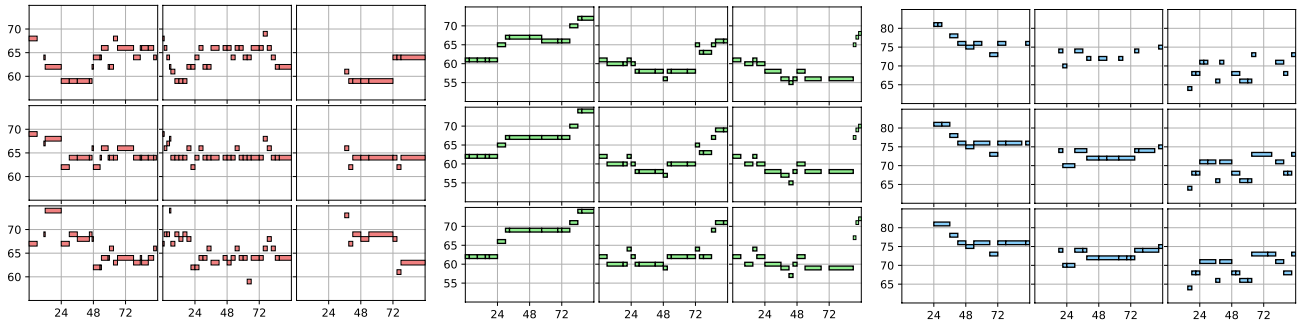
Modeling Performance. Table 1 shows negative log-likelihoods (NLLs; per symbol and lower bound) and reconstruction accuracies (accuracies) for the test dataset. We compare baseline variational auto-encoders (VAEs) [16] (its model implementation and the training algorithms are the same as our *residual model* without the condition y except that the number of dimensions for the latent variable is doubled for fair comparisons) with our three proposed models. Each of our models is divided into two models (*residual/extraction model* (see Sec.2.2)), whose individual NLLs and accuracies are shown in the table. The additions of two NLLs for the *residual/extraction model* are comparable to the NLLs for the baseline VAE. The multiplications of two accuracies for the *residual/extraction model* are also comparable to the accuracies for the baseline VAE.

4.4 Concept-Axes Interpolation

As depicted in Fig.4, given two musical fragments (top-left and bottom-right) in each subfigure with 5×5 fragments, the other 23 fragments "in between" are yielded using *concept-axes interpolation*, whereas the interpolation in a traditional latent space [27, 28] would produce only the three diagonal fragments. In these figures, horizontal axes are the *extraction space* $\mathcal{Z}_e$ axes, and moving towards the axes smoothly changes the extracted target concepts (i.e., *rhythm*, *contour*, and *F&C*-invariant), while generally not changing the *residual* concepts. On the other hand, vertical axes are the *residual space* $\mathcal{Z}_r$ axes, showing smooth change in *residual* concepts and little change in target concepts. Note that here we only show 5×5, but interpolation of the two fragments could be arbitrarily finer/coarser (at any positions of the subfigure) upon users' demand. In Fig.4a, *rhythm* direction preserves "pitch appearing order," showing that the $\mathcal{Z}_r$ successfully captures concepts that are important but might be difficult to learn in $\mathcal{Z}_e$. In Fig.4b, in non-*contour* direction, fragments tend to transpose to match the "pitch set" of the bottom-right fragment without changing *contour*, which captures the scale-like concept. In *contour* direction, the fragments gradually adopt the descending-like contour. In other words, only the descending-like feature is retrieved from the bottom-right fragment to generate fragments in the first row. For Fig.4c, in *F&C*-invariant direction, the gradual altering of fragmenting or consolidating notes is observed. Similar analyses can also be done in longer sequences: Figure 3 shows results for $T = 96$ in piano roll representation with each subfigure consisting of 3×3 musical fragments.

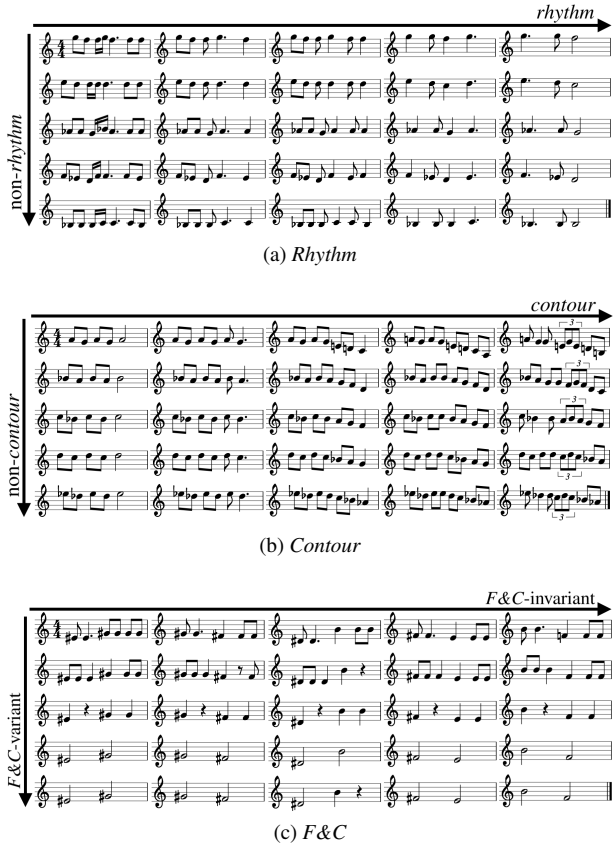
4.5 Concept-Aware Variation Generation

In Fig.5, our variation generation approach is applied to ExtRes/VAE, which are for concept-aware/-unaware variation generations, respectively. In each column of the figure, the generated variations are sorted from top to bottom in the ascending order of learned Mahalanobis distance (see Sec.2.3). Figure 5a depicts the variations for *rhythm*. The second column shows the *extraction space* $\mathcal{Z}_e$ variations, where various rhythms are produced without changing the other factors. In contrast, *residual space* $\mathcal{Z}_r$ variations (the third column) all keep the rhythm unchanged, whereas the other factors such as the "order of used pitches" change. Variations by VAE (the fourth column) mix factors of rhythm/non-rhythm, without drastic change in rhythm. Figure 5b depicts the variations for *contour*. $\mathcal{Z}_e$ variations have various contours without changing other factors. In contrast, $\mathcal{Z}_r$ variations all keeps the contour unchanged, whereas the scale-like concept "set of used pitches" changes. VAE yields relatively conservative variations with mixed contour/scale-like factors. Lastly, variations for *F&C* are in Fig.5c. For the $\mathcal{Z}_e$ variations, melodies with different pitches are generated, while fixing the concept of "the consecutive notes with the same pitch" except the third row from the bottom. As for the $\mathcal{Z}_r$ variations, F&C of notes are observed, which is not the case in the fourth column except the second row from the bottom, indicating that our ExtRes successfully captures the notion of F&C. Note that the variations by ExtRes are generated with simply $(0, 1)$ or $(1, 0)$ variance scales for two spaces to clearly explain the capabilities of ExtRes, but one could interactively change the scale ratio to obtain



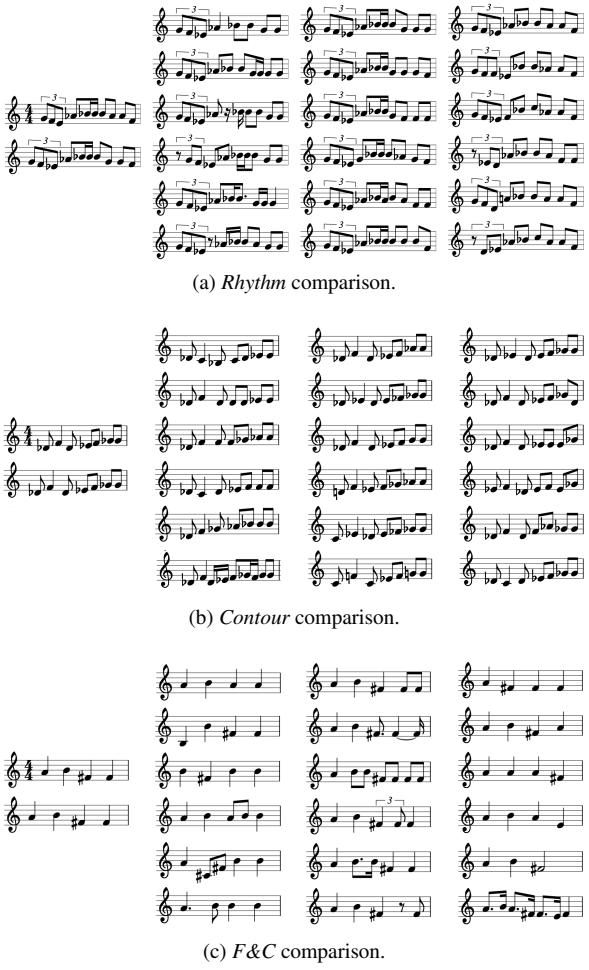
(a) *Rhythm* (b) *Contour* (c) *F&C*

Figure 3: *Concept-axes interpolation (T=96, 0.5 stride).* Vertical axis denotes MIDI note number, and horizontal axis denotes $t \in \{1, ..., T\}$.



(a) *Rhythm* (b) *Contour* (c) *F&C*

Figure 4: *Concept-axes interpolation (T=24, 0.25 stride).*



(a) *Rhythm* comparison. (b) *Contour* comparison. (c) *F&C* comparison.

Figure 5: Our variation generation approach is applied to ExtRes and VAE. For each subfigure (a,b,c), left (first column): top is an original fragment and bottom is its reconstruction; center left (second column): ExtRes *extraction space* $\mathcal{Z}_e$ variations; center right (third column): ExtRes *residual space* $\mathcal{Z}_r$ variations; right (fourth column): VAE variations.

variations with desired mixing proportions of the concepts.

5. CONCLUSION AND FUTURE WORK

We presented a latent space decoupling model for learning concept spaces using domain knowledge. For monophonic symbolic music, we experimented on three musical concepts. Controllability in generation was improved by *concept-axes interpolation* and *concept-aware variation generation*. In future, other musical concepts mentioned in Sec.1 should also be tested on ExtRes. We believe that this paper opens up possibilities for learning models with concept-aware inference/generative processes to be used for different information retrieval tasks or more controlled and flexible generations.

6. ACKNOWLEDGMENTS

I would like to thank Gaëtan Hadjeres for data encoding codes and reviewing my manuscript. I also greatly appreciate Frank Nielsen for helping me with writing the manuscript.

7. REFERENCES

[1] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proc. of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016.

[2] Emilios Cambouropoulos, Tim Crawford, and Costas S. Iliopoulos. Pattern processing in melodic sequences: challenges, caveats and prospects. *Computers and the Humanities*, 35(1):9–21, 2001.

[3] Tian Qi Chen, Xuechen Li, Roger B. Grosse, and David K. Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Annual Conference on Neural Information Processing Systems*, 2018.

[4] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Annual Conference on Neural Information Processing Systems*, 2016.

[5] Walter Dowling. Scale and contour: Two components of a theory of memory for melodies. *Psychological Review*, 85:341–354, 07 1978.

[6] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martín Arjovsky, Olivier Mastropietro, and Aaron C. Courville. Adversarially learned inference. In *International Conference on Learning Representations*, 2017.

[7] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. GANSynth: Adversarial neural audio synthesis. In *International Conference on Learning Representations*, 2019.

[8] Jesse Engel, Matthew Hoffman, and Adam Roberts. Latent constraints: Learning to generate conditionally from unconditional generative models. In *International Conference on Learning Representations*, 2018.

[9] Philippe Esling, Axel Chemla-Romeu-Santos, and Adrien Bitton. Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces. In *Proc. of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*.

[10] Gaëtan Hadjeres, Frank Nielsen, and François Pachet. GLSR-VAE: Geodesic latent space regularization for variational autoencoder architectures. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI 2017)*, pages 1–7. IEEE, 2017.

[11] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. DeepBach: a steerable model for Bach chorales generation. In *Proc. of the 34th International Conference on Machine Learning*, 2017.

[12] Irina Higgins, Nicolas Sonnerat, Loic Matthey, Arka Pal, Christopher P Burgess, Matko Bošnjak, Murray Shanahan, Matthew Botvinick, Demis Hassabis, and Alexander Lerchner. SCAN: Learning hierarchical compositional visual concepts. In *International Conference on Learning Representations*, 2018.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[14] Arka Pal Christopher Burgess Xavier Glorot Matthew Botvinick Shakir Mohamed Alexander Lerchner Irina Higgins, Loic Matthey. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

[15] Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Annual Conference on Neural Information Processing Systems*, 2014.

[16] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[17] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. Learning transposition-invariant interval features from symbolic music and audio. In *Proc. of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*.

[18] Steven Livingstone, Caroline Palmer, and Emery Schubert. Emotional response to musical repetition. *Emotion (Washington, D.C.)*, 12:552–67, 06 2011.

[19] David Meredith. The ps13 pitch spelling algorithm. *Journal of New Music Research*, 35, 06 2006.

[20] Marcel Mongeau and David Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24(3):161–175, 1990.

[21] Simon Moulieras and François Pachet. Maximum entropy models for generation of expressive music. *CoRR*, abs/1610.03606, 2016.

[22] Siddharth Narayanaswamy, T. Brooks Paige, Jan-Willem van de Meent, Alban Desmaison, Noah Goodman, Pushmeet Kohli, Frank Wood, and Philip Torr. Learning disentangled representations with semi-supervised deep generative models. In *Annual Conference on Neural Information Processing Systems*, 2017.

[23] François Pachet, Alexandre Papadopoulos, and Pierre Roy. Sampling variations of sequences for structured music generation. In *Proc. of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017*.

[24] François Pachet, Jeff Suzda, and Dani Martínez. A comprehensive online database of machine-readable lead-sheets for jazz standards. In *In Proc. of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013*.

[25] François Pachet. A joyful ode to automatic orchestration. *ACM TIST*, 8:18:1–18:13, 2016.

[26] Alexandre Papadopoulos, Pierre Roy, and François Pachet. Assisted lead sheet composition using flowcomposer. In *CP*, 2016.

[27] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. In *Proc. of the 35th International Conference on Machine Learning*, 2018.

[28] Ian Simon, Adam Roberts, Colin Raffel, Jesse Engel, Curtis Hawthorne, and Douglas Eck. Learning a latent space of multitrack measures. *CoRR*, abs/1806.00195, 2018.

[29] Yifei Teng, Anny Zhao, and Camille Goudeseune. Generating nontrivial melodies for music as a service. In *Proc. of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017*.

UNMIXER: AN INTERFACE FOR EXTRACTING AND REMIXING LOOPS

Jordan B. L. Smith **Yuta Kawasaki** **Masataka Goto**
National Institute of Advanced Industrial Science and Technology (AIST), Japan

ABSTRACT

To create their art, remix artists would like to have segmented stem tracks at their disposal; that is, isolated instances of the loops and sounds that the original composer used to create a track. We present Unmixer, a web service that will analyze and extract loops from any audio uploaded by a user. The loops are presented in an interface that allows users to immediately remix the loops; if users upload multiple tracks, they can easily create mashups with the loops, which are automatically matched in tempo. To analyze the audio, we use a recently-proposed method of source separation based on the nonnegative Tucker decomposition of the spectrum. To reduce interference among the extracted loops, we propose an extra factorization step with a sparseness constraint and demonstrate that it improves the source separation result. We also propose a method for selecting the best instances of the extracted loops and demonstrate its effectiveness in an evaluation. Both of these improvements are incorporated into the backend of the interface. Finally, we discuss the feedback collected in a set of user evaluations.

1. INTRODUCTION

Professional and amateur composers across the world enjoy creating remixes and mashups. Remixes are pieces of music that are composed, in whole or in part, using snippets of another audio recording, whereas mashups juxtapose snippets of two or more recordings [8]. Creators of official remixes usually have access to the stem tracks for a recording, but these resources are not typically available to amateur remixers. Unofficial remixes, sometimes called 'bootlegs', can still be created using clips of the down-mixed recording of the song [8], but this presents two challenges: first, it is time-consuming to manually segment an audio track to select the most prominent or interesting bars or sounds. Second, because the sources in the original audio recording are down-mixed, the artist may have to use equalization (i.e., filtering out certain frequencies) to achieve a simple separation of sources.

We present Unmixer [1], an interface that accomplishes both of these tasks for the user, using a source separation technique instead of equalization (see Fig. 1). The interface allows a user to upload any song; it then processes the audio and returns a set of loops. The user can then play with the loops on the spot, re-combining the loops live. If the user uploads more songs, they can also juxtapose loops from different songs, which are automatically tempo-matched, to create live mash-ups. Finally, users can download the loops to remix offline.

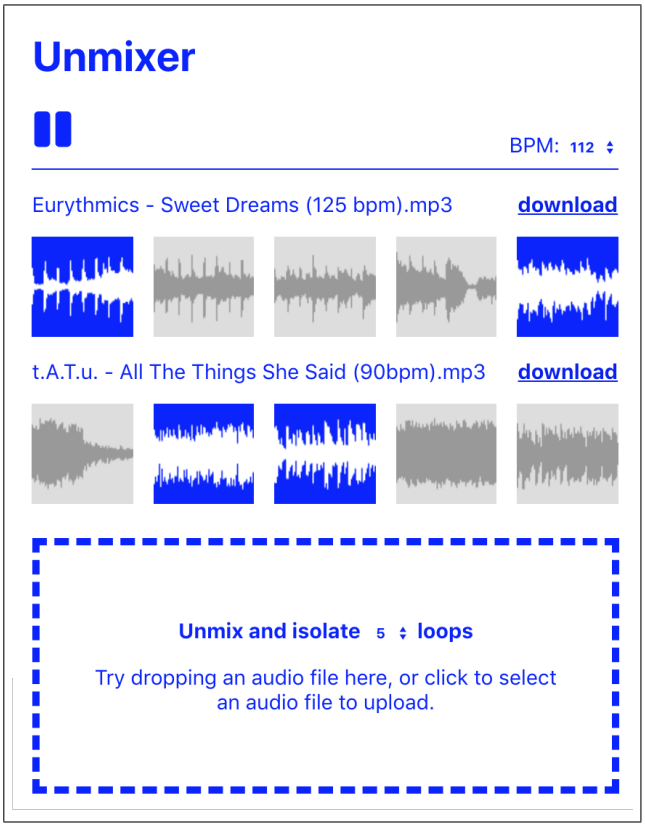


Figure 1. Screenshot of Unmixer interface with two songs loaded. In its current state, two loops from each song are playing.

The website was inspired in part by Adventure Machine [2], a Webby-nominated site designed to promote an album by the musician Madeon. That site allowed visitors to remix stem samples from a Madeon track, and it attracted significant traffic, according to the designers [3]. The thought that inspired us was: what if visitors could populate a remixing interface with samples from any track

[1] Available at: `https://unmixer.ongaaccel.jp`

[2] `https://www.madeon.fr/adventuremachine`

[3] `https://developers.google.com/web/showcase/2015/adventuremachine`

they own? With Unmixer, we aim to achieve this vision.

In the next subsection, we discuss alternative methods of extracting loops. In Section 2, we explain the features of the interface and some design considerations. In Section 3, we explain the algorithm which supports it [22] and propose an extension and improvement to it. Both contributions are evaluated. In Section 4, we present a usability study, and we end with a discussion (Section 5).

1.1 Tools for extracting loops and creating remixes

Existing interfaces for creating live remixes from a library of samples include AdventureMachine and Beat-Sync-Mash-Coder [7], but these do not allow you to populate the interface with automatically extracted loops. The web application *Girl Talk in a Box*[4] cuts any song into chunks for a user and offers novel resequencing options, but does not separate sources or allow the users to play multiple chunks at once. Advanced users can always use a Digital Audio Workstation (DAW), which is the most powerful and flexible way to compose a remix, but they will need to do the work of extracting loops on their own. That said, software exists to support this tedious task, such as [3] and [18], both of which require users to guide the algorithm by indicating regions of the spectrum to ignore or focus on. In sum, we are not aware of another interface that, given an input song, extracts source-separated loops and presents them to the user for remixing.

To extract repeating patterns, two broad approaches are popular: kernel-additive modeling [10], including harmonic-percussive source separation (HPSS) [4] and REPET, a method of foreground-background separation that models a looping background, and its variants [17]. However, these are binary separations: only two sources are obtained. The family of non-negative matrix factorization (NMF) approaches includes NMF [21], NMF deconvolution [19], and non-negative tensor factorization (NTF) [5]. All take advantage of redundancies in the signal—repeating spectral templates, transpositions, stereo dependence, etc. By applying NMF iteratively, [20] separated layers of electronic music that built up progressively, but did not model the extracted sources in terms of loops. This is a drawback shared by all of these algorithms: they produce full-length separated tracks. To obtain short, isolated loops, some extra step is required. However, a recent NTF system [22] models the periodic dependencies in the signal and is thus suitable for extracting loops directly; it forms the basis for our system and is described in Section 3. An orthogonal approach to extracting loops is that of [12], which seeks only to extract drum breaks (short drum solos that are desirable for remixes) by devising a percussion-only classifier.

2. WEB SERVICE AND INTERFACE

The purpose of the interface is to allow users to remix and mash-up the songs they love, and to perform automatically the work of isolating loops, normally done through editing and equalization or source separation.

The interface has just a few, simple features, all visible in the screenshot (see Fig. 1). To begin, a user must upload a song from their hard drive, using the box at the bottom of the page. They may adjust the number of loops to extract using the drop-down list, with possible values between 3 and 10. The audio is processed on the web server using the algorithm outlined in Section 3; once the audio has been analyzed, the interface is populated with a set of loop 'tiles', each tile bearing a waveform sketch. The possible actions are then:

1. click on loop tiles to activate or deactivate them;
2. change the global tempo (drop-down list at top-right of the interface);
3. pause playback (button at top-left);
4. download a zip archive containing all the loops for a given song;
5. choose an additional song (and number of loops to extract) using the box at the bottom of the page.

Uploading and processing a new file can take a while (currently between 5 and 10 minutes), but users can still use the other functions (playing tiles and changing the tempo) while waiting for the next batch of tiles. Users can add any number of songs to a single workspace.

All the loops have the same duration (equivalent to two bars of audio), and the playback of the loops is synchronized so that the downbeats align, no matter when the user activates them. When a user uploads the first song, the global tempo is set to the detected tempo. New songs added to the workspace are tempo-shifted to match the global tempo.

The interface was built as a web application using React[5], making it available on any web-accessible device through a browser. To handle audio playback, we use the Web Audio API, making it easy to synchronize playback of all the tiles and control the playback rate. The audio analysis is run on our server, using the algorithm described in the next section. The web server keeps a log of uploaded audio files and the analysis computed for each. To save time, if the system recognizes an uploaded audio file, it re-uses the old analysis. However, it does not re-use the old audio; it re-extracts the component loops from the new audio to return to the user, thereby avoiding any copyright issues related to redistributing audio.

3. LOOP EXTRACTION

Our approach to extracting loops is based on [22], but we make two contributions: first, we suggest and evaluate methods for selecting individual loops; second, we propose and evaluate an extra "core purification" step.

3.1 Review of NTF for source separation

The pipeline for our system, based on the work of [22], is shown in Fig. 2. First, the mono spectrogram X is di-

[4] http://girltalkinabox.playlistmachinery.com

[5] https://reactjs.org/

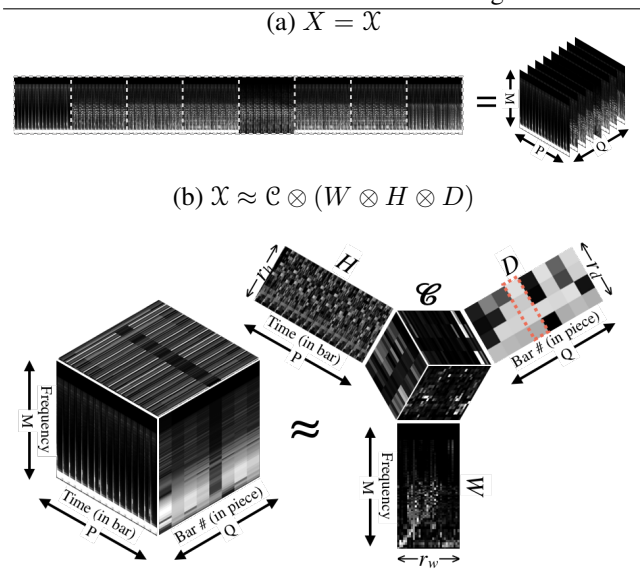


Figure 2. Overview of system, using example '125_acid' [11]. (a) Spectrum of 8-bar song reshaped into tensor. (b) Tucker decomposition expresses song as product of frequency templates W, rhythm templates H, repetition templates D, and a core tensor $\mathcal{C}$. In this example, $(M, P, Q) = (1025, 661, 8)$ and (r_w, r_h, r_d) = (32,20,4).

vided into downbeat-sized windows using a beat-tracker. (Unmixer uses the madmom system [1], but the illustrated example uses the known downbeats.) The $M \times P$-shaped spectrogram windows (one for each bar) are stacked into a new dimension, creating an $M \times P \times Q$ tensor $\mathcal{X}$ (Fig. 2a). Then, using Tensorly [9], we compute the non-negative Tucker decomposition (NTD) with ranks (r_w, r_h, r_d), approximating the tensor as the outer product $\mathcal{X} \approx \mathcal{C} \otimes (W \otimes H \otimes D)$ (Fig. 2b). In plain terms, the NTD models the spectrum with three meaningful components: a set of spectral templates W (the *sounds*), a set of within-bar time-activation templates H (the *rhythms*), and a set of loop-activation templates D (the *layout*, i.e., the arrangement of loops in the song). In Fig. 2b, a decomposition of an artificial 8-bar ($W = 8$) stimulus [11] has been approximated using $r_d = 4$ loop templates, giving a good estimate of the layout of the piece (shown in Fig. 6). The templates are diverse: some sounds are monophonic, others polyphonic; some rhythms are percussive, others sustained.

The sparse core tensor is $\mathcal{C}$; a non-zero element $\mathcal{C}_{[i,j,k]}$ indicates that sound i is played with rhythm j, and this pattern is repeated according to layout template k. To separate the contribution of the k^{th} loop, use the k^{th} row of D to take the outer product $\mathcal{C} \otimes (W \otimes H \otimes D_k) \approx \mathcal{X}_k$, and unfold the tensor into X_k. The reconstructed real-valued spectrum X_k is not sufficient to recreate the signal; we must apply softmasking (as outlined in [17] and implemented in librosa [13]) and use the original phase:

$$y_k = \text{ISTFT}(\text{phase}(X) \cdot \text{mag}(X) \cdot \frac{X_k^p}{X_k^p + X^p}) \quad (1)$$

where p is the power of the softmask operation. As noted

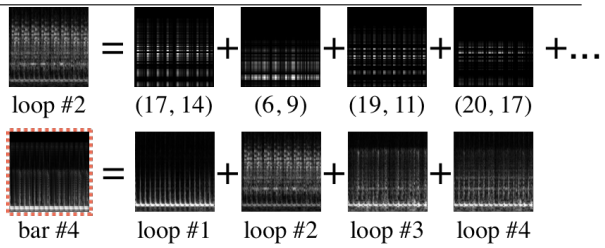


Figure 3. Example reconstruction of spectrum for a single loop (#2) from multiple (freq, rhythm) combinations (above), and for a single bar (#4, also indicated in Fig. 2) from multiple loops (below).

by [6], using softmask filters is convenient, although not necessarily optimal, for non-negative approaches like ours.

The core tensor can be interpreted as a set of "recipes" for building the loops, the recipe for the k^{th} loop being the $r_w \times r_h$ slice of the core tensor $\mathcal{C}_{[:,:,k]}$. To see how, note that $W_i \otimes H_j$, the outer product of the i^{th} sound with the j^{th} rhythm, is an $M \times P$ spectrum of a single bar. The outer product $\mathcal{C}_{[:,:,k]} \otimes (W \otimes H)$ thus represents a sum of such one-bar spectra, leading to the k^{th} loop template. Fig. 3 shows how the 2^{nd} loop template is the sum (in descending order of magnitude) of individual $W_i \otimes H_j$ components. Each bar in the piece will be a superposition of several loops: the bottom part of the figure shows how the 4^{th} bar consists of copies of each loop.

3.2 Loop selection

The output of the algorithm in [22] is a set of full-length tracks, each corresponding to the contribution of one loop. For remixing purposes, we only want a bar-length version of each loop, as cleanly separated as possible. The plain approach is to extract the full-length track, then excerpt a single bar, but the question is then: which bar to select?

There are at least two factors to consider: how loud a given loop instance is, and how strongly that instance stands out from the other parts. Loudness can be maximized by choosing the bar that takes the maximum value in the loop activation matrix: i.e., to select the best bar for the k^{th} loop, choose argmax($D_{[k,:]}$). To minimize cross-talk, we consider two approaches. First, we may normalize the columns of D by choosing argmax($D_{[k,:]} - \bar{D}$), where $\bar{D}$ is a vector of the column means of D.

Alternatively, the coefficients from the softmask operation could estimate how prominently a given loop stands out from the background. In this approach, we compute the softmask coefficients for each bar (i.e., the fractional part of Equation 1), and then select the bar which maximizes the total value of the mask. That is, if $M_{k,i}$ gives the softmask matrix for the i^{th} bar for the k^{th} loop, we select argmax($\sum M_{k,:}$). Finally, we may combine any or all of these decision criteria.

Evaluation To determine the most reliable selection method, we tested them on a set of stimuli assembled by [11]. The test set contains 7 compositions, each containing 4 loops arranged in the same 8-bar layout (shown

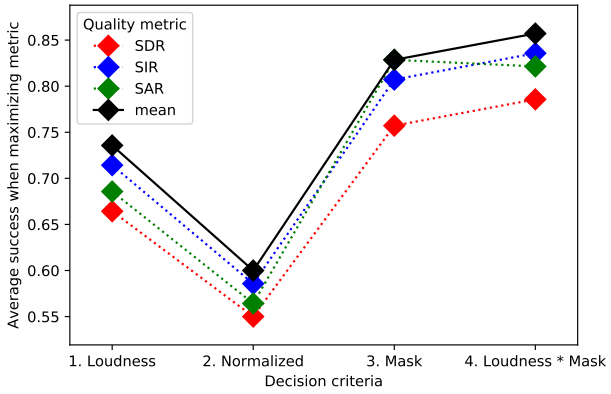


Figure 4. Main effect of bar selection strategies.

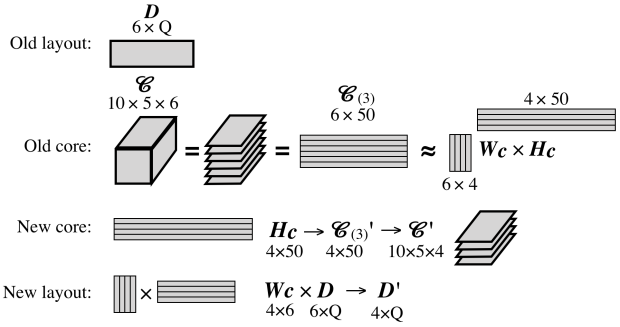


Figure 5. Illustration of core tensor purification. W_C and H_C are estimated using NMF with H_C constrained to be sparse.

in the bottom part of Fig. 6). We ran the NTD algorithm on all the stimuli (with $(r_w, r_h, r_d) = (32, 40, 4)$), and then measured the reconstruction quality of each loop in each bar. We measured reconstruction quality using SDR, SIR and SAR, the source-to-distortion, -interference, and -artefact ratios, respectively, calculated using mir_eval [16]. (For each metric, higher is better.) Then, we tested how frequently the optimal bars were selected by maximizing each criterion. The four tested criteria were: loudness only ($D_{[k,:]}$); normalized loudness ($D_{[k,:]} - \bar{D}$); mask only ($M_{[k,:]}$); and loudness times mask ($D_{[k,:]} \cdot M_{[k,:]}$).

The main effect of the choice of criterion is shown in Fig. 4, which shows how often a given strategy (e.g., "select the loudest bar") correctly found the bar that maximized a given metric. Using loudness alone (1), the optimal bar was selected at least two-thirds of the time. Normalizing the matrix D to diminish cross-talk (2) was too coarse, with the best bar selected less than 60% of the time. However, multiplying the loudness by the mask (4) to diminish cross-talk led to the best overall result, with the optimal bar selected around 80% of the time. The choice of criterion depends slightly on the quality metric (SDR, SIR or SAR) being maximized: if the priority is to maximize SAR, using the mask alone (3) may be advised, but using the loudness-and-mask criterion worked best overall, and it is the criterion used in the Unmixer system.

3.3 Loop purification

As reported by [22], a problem with the algorithm is that the extracted loops can be redundant. For example, suppose a song contains a drum pattern A and a synth pattern B, which are independent, but where B never occurs without A. Instead of modeling the independent patterns A and B, the algorithm is likely to learn one pattern for A and another for $A + B$. This error is not fixed by changing the number of loop templates that the model should learn.

The problem would be avoided if the core tensor were estimated with a sparsity constraint. Note that sparsity is only desired in the 3^{rd} dimension, to prevent the $r_w \times r_h$ slices (the loop 'recipes') from being too similar. Denseness is still desirable in the other dimensions; indeed, allowing different sources (frequency templates) to share rhythms (time activation templates), and vice versa, was a motivation to use NTD to begin with, since it allows an accurate and meaningful reconstruction without the model size becoming infeasibly large.

Algorithms for sparsity-constrained tensor factorizations exist [14], but we are not aware of any existing tensor decomposition packages that implement them. [6] Therefore, we propose to follow regular NTD with a "core-purification" step, using sparsity-constrained NMF to simplify the core tensor.

The process is illustrated in Fig. 5, supposing a decomposition of initial rank $(10, 5, 6)$. First, we take $\mathcal{C}_{(3)}$, the third-dimension unfolding of the core tensor $\mathcal{C}$, so that each horizontal slice of $\mathcal{C}$ is reshaped into a row in $\mathcal{C}_{(3)}$. Then, we apply sparse NMF (using the SNMF function in Nimfa [23]) to model $\mathcal{C}_{(3)} \approx W_C \times H_C$, imposing sparsity on H_C only. (It is also possible to use simple NMF, without the sparsity constraint; this is tested later as a baseline.) The rows of H_C give a new set of maximally independent vectors; when the matrix H_C is refolded, it gives $\mathcal{C}\prime$, a set of maximally independent recipes. The matrix W_C tells us in what proportion to add the previous 6 templates to each other to obtain the new set of 4 templates; hence, we use it to transform the original layout D into $W_C \times D = D'$.

An example showing the promise of this method is shown in Fig. 6. First, we computed the NTD of a song using ranks $(10, 5, 4)$; second, we computed a new NTD with ranks $(10, 5, 6)$, and purified the core so that the final shape was $(10, 5, 4)$. The 4 slices of each core are shown in parts (a) and (b) of the figure, along with the corresponding layouts: first, the loop activation matrix D; second, the revised matrix D'. Each is an estimate of the ground truth layout at bottom. Whereas there is a clear redundancy among loops 2 and 3 in the first set (plain NTD), the redundancy has been reduced in the second (NTD with purification). The reconstruction error for the second set is actually a little higher (0.186 compared with 0.185), but the separated audio will be of higher quality. In the actual example (where we used ranks $(32, 40, 4)$ and $(32, 40, 6)$), the mean SDR and SIR for the 4 estimated loops climbed from 1.98 and 9.96 to 6.85 and 16.16, respectively, while

[6] NB: factorization of 'sparse' tensors is often a supported feature, but not the estimation of sparse outputs.

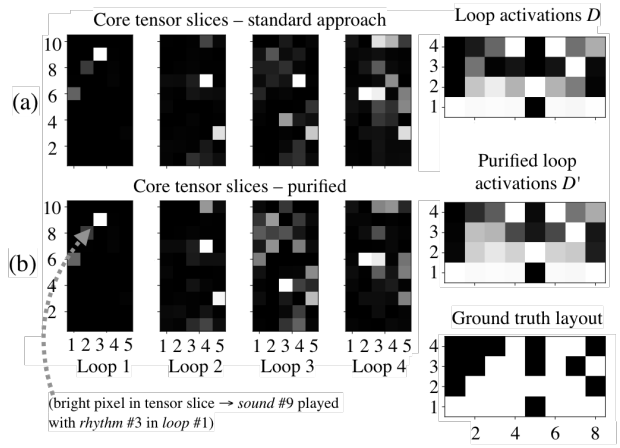


Figure 6. Comparison of (a) core estimated with plain NTD vs. (b) NTD with purification (core purified from 6 to 4), using stimulus '125_acid' [11]. The pixel at $(4, 7)$ which appears redundantly in loops 2–4 in (a) has been subdued in (b). Each subfigure's pixels are linearly scaled between its min and max.

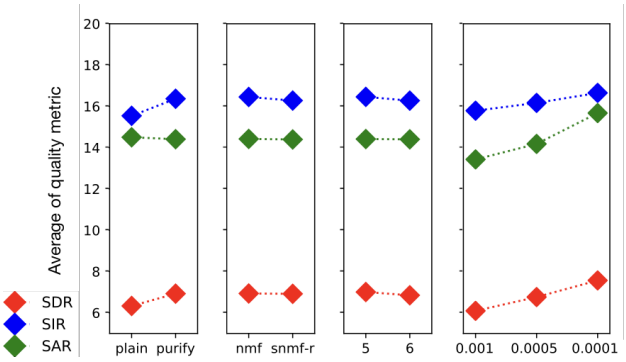


Figure 7. Main effect of loop purification and runtime strategies.

SAR only decreased from 17.97 to 17.38. (These metrics are explained in the next section.)

Evaluation We evaluated the effectiveness of the proposed purification approach on the same dataset used in Section 3.2. We ran a fully-factorial design, exploring the following parameters and settings:

1. Plain NTD vs. purification;
2. Purification method: unconstrained NMF vs. the proposed SNMF approach;
3. Initial rank: 5 or 6 for purification methods only;
4. Tolerance: stopping criterion for computation of initial tensor (0.001, 0.0005 or 0.0001).

We use the known downbeat locations, and the final rank was always set to 4, the true number of loops in the examples. Although these must be estimated in a real-life scenario, we can still evaluate the impact of the purification method, initial rank, and tolerance. The reported metrics are SDR, SIR and SAR, as before.

Fig. 7 shows the main effects. We see that the purification step increased SDR and SIR, with only a minor decrease in SAR. This was the hoped-for result: less interference among the extracted signals (SIR), even if some reconstruction quality is sacrificed (more artefacts, SAR). However, we found that there was little difference between the proposed SNMF-R approach and a simpler NMF approach. We also see that purifying an estimate to rank 4 from rank 5 gave slightly better results than from rank 6. To put these effects in context, we also show the effect of reducing the tolerance, i.e., allowing the tensor factorization to run for longer. The purification step increased SIR nearly as much as did reducing the tolerance by 90%.

4. USER EVALUATIONS

To assess the usability of the system and solicit feedback about the audio quality and enjoyability of app, we conducted a user evaluation. We solicited 8 participants (4 men and 4 women), all between 25 and 40 years old. The study had three components:

1. A background questionnaire covering: their musical training (using the standard Goldsmiths MSI short test [15]); their experience with audio editing interfaces and audio production; and their familiarity with the songs used in the study.
2. A 10-minute test of the interface. Participants were asked to upload a song, familiarize themselves with the interface, then upload 2–5 more and explore the combinations of sounds.
3. A feedback questionnaire with Likert-scale and free-text questions focused on usability, audio quality, enjoyment, and potential new features. Usability questions were adapted from the standard Systems Usability Scale [2]).

We limited participants to a set of 11 audio files that the system had already seen, which greatly sped up the processing time: users did not have to wait the typical 5–10 minutes for the Tucker decomposition to converge; they only needed to wait for the system to receive the audio, use the pre-existing decomposition to extract new audio files, and load said audio files into the interface, all of which takes roughly 30 seconds per file. Otherwise, the system they used is the same that is available live, now at `unmixer.ongaaccel.jp`. The ranks of the analysis are $(r_w, r_h, r_d) = (50, 40, 30)$ (with r_d purified to 25 using SNMF-R), and the NTD is solved with tolerance 0.0001.

User background: The 8 user testers included musical experts and novices: 5 played musical instruments, and 6 had experience editing audio files. Of those 6, 4 also had some experience either using a DAW or creating a remix or mashup. According to the MSI test, the musical sophistication of 4 users was within a standard deviation of average [15], with 1 user above this range and 3 below.

Usability: There was unanimous agreement that the system was "easy to use" and that users thought "most people would learn to use this system very quickly." In fact, in the free response, ease of use was cited by 7 users as one of the best things about Unmixer, especially for "musical novices" or "a beginner [like] myself". One aspect of the

interface that users found inconvenient, though, was the inability to anticipate what a loop would sound like. One wrote that "you need to guess what's in each loop based on the waveform and sometimes it's not what you expected or wanted", and suggested a short text label (e.g., 'vox', 'drum', 'synth') to indicate the content; another suggested visual hints. One user also noted that they did not know what the impact on the loop content or quality would be if they changed the number of loops to extract.

Sound quality: Asked whether "the sound quality of the loops was poor", users were divided, with 4 each agreeing and disagreeing. However, one disagreer later explained that while "audio quality of some samples was not great, ...it was possible to find good quality ones." The quality of source separation was appreciated: 6 agreed that "most loops isolated a single source (e.g., drums, vocals, synth, bass)", and 5 agreed that "loops within one song had a nice variety" (with 2 disagreeing). No one agreed that the "loops from different songs were too similar".

Enjoyment: 6 agreed that "the combination of loops was often interesting", and 4 agreed that they would "like to use Unmixer frequently" (with 2 disagreeing). The interface struck users as novel: none agreed that the "interface was similar to others I've used before". Four agreed that "a remix artist could build a good song from these loops" (with 2 disagreeing); however, among users with experience using DAWs or creating mashups, opinion on this was split 2-against-2. Users saw different reasons to enjoy the interface: one "enjoyed the experience of doing something new"; another wrote that "it was very easy to try out new ideas very quickly". Opinon also varied on what users would use Unmixer for, with responses including: generating mashups "for interludes or as backing music"; "using it to prototype [remix] ideas quickly"; "having fun at a house party—home-DJ style"; creating "nice effects for a video"; and "taking some of my own music" and generating "new ideas from it."

Potential features: To understand what future developments for the interface would be most desirable, we polled users' opinions on a list of 7 suggested features: controls for (1) loudness, (2) pitch-shifting and (3) equalization (e.g., to boost the bass or mid-range); (4) keyboard shortcuts for activating loops; (5) allowing more than 10 loops; and colour-coding loops by (6) type (e.g, vocals, synth) or (7) tonality. Users expressed broad approval for all of these except for (5), although if loops were colour-coded, it might become more desirable to have more loops available. Almost all users indicated that they had already thought of feature (1). When asked what change in Unmixer would make it more usable or useful, two users suggested a timeline functionality so that repeating sequences of loops could be made; two also wanted to be able to 'save' or 'download' the combinations they had created.

5. DISCUSSION

Our user feedback confirmed for us the usability of the system: without supervision, all participants completed the steps of the user study. The positive comments assure us that the app holds promise as a tool for exploring interesting remix possibilities. However, to be most useful and engaging, we need to improve the sound quality of the extracted audio. The interface as it is may even be *too* simple: we expect that with a few changes, like adding keyboard shortcuts and having the tiles give some visual hint as to their content, we can increase user enjoyment and satisfaction. On the other hand, we should not implement all of the features discussed; to do so would be to program an in-browser, fully-functional DAW, whereas our focus is to provide users with loops extracted from songs and allow them to experiment with combinations.

The user study differs from the live experience in a few ways: (1) It allowed a choice among 11 pre-selected tracks, although real users are free to choose any music from their library; and (2) it featured a streamlined experience with minimal waiting for the system to analyze the files. We must collect more feedback from realistic scenarios to understand the system usability. If we cannot speed up the algorithm, we may need to adjust the interface to maintain the feeling of interactivity. For instance, we could provide the user with a quick-and-dirty set of extracted loops, and refine them in the background while the user experiments. Also, the current version kept fixed the ranks of the analysis and the amount of purification. We hope to make these tuneable, or have the system predict the optimal values.

There remain several ways to improve the source separation quality. We have reconstructed the signals using softmask filtering, but we could realize further improvements, and even speed up the algorithm, by using newer masking methods, such as the divergence-based masks proposed by [6]. To improve the system's output, we could use equalization to refine the separated loops. As noted in the introduction, creators of unofficial remixes often use equalization to separate sources in lieu of source separation algorithms. We could blindly apply equalization or HPSS to create, say, bass, treble and percussive versions of each loop, or use a set of instrumentation-detection functions (similar to [12]) to select the best ones.

To evaluate these proposed improvements, it is important to collect more expansive test sets. The small test set of [11] was sufficient to assess the best bar-picking strategy and whether the purification step was useful. However, future work on this task ought to treat more diverse stimuli—more genres, loops per song, and loop layouts—to gauge how the system copes with realistically complex pieces.

6. CONCLUSION

We developed Unmixer, a web-app where users can upload music, extract loops, remix them, and mash-up loops from different songs. Expert and novice users found it easy to use; many also found it a novel way to develop remix ideas, although higher-quality audio output may be required for polished remixes. The backend uses the NTD-based source separation algorithm from [22]; we proposed and tested techniques to select the best reconstructed loop excerpts, and techniques to refine the loop layout, measurably improving on the baseline system output.

7. ACKNOWLEDGMENTS

This work was supported in part by JST ACCEL Grant Number JPMJAC1602, Japan.

8. REFERENCES

[1] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: a new Python Audio and Music Signal Processing Library. In *Proc. of the ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, November 2016.

[2] John Brooke. SUS: A 'quick and dirty' usability scale. In Patrick W. Jordan, Bruce Thomas, Bernard A. Weerdmeester, and Ian L. McClelland, editors, *Usability Evaluation in Industry*, pages 189–194. Taylor and Francis, London, UK, 1996.

[3] Nicholas J Bryan, Gautham J Mysore, and Ge Wang. ISSE: An interactive source separation editor. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 257–266. ACM, 2014.

[4] Derry FitzGerald. Harmonic/percussive separation using median filtering and amplitude discrimination. In *Proc. of the International Conference on Digital Audio Effects*, Graz, Austria, September 2010.

[5] Derry FitzGerald, Matt Cranitch, and Eugene Coyle. Sound source separation using shifted non-negative tensor factorisation. In *Proc. of the IEEE ICASSP*, volume 5, Toulouse, France, 2006.

[6] Derry Fitzgerald and Rajesh Jaiswal. On the use of masking filters in sound source separation. In *Proc. of the International Conference on Digital Audio Effects*, York, UK, 2012. Dublin Institute of Technology.

[7] Garth Griffin, Youngmoo E. Kim, and Douglas Turnbull. Beat-sync-mash-coder: A web application for real-time creation of beat-synchronous music mashups. In *Proc. of the IEEE ICASSP*, pages 437–440, Dallas, TX, USA, 2010.

[8] Sheena D Hyndman. No money, mo' problems: The role of the remix in restructuring compensation for producers of electronic dance music. *MUSICultures*, 41(1):57–72, 2014.

[9] Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. Tensorly: Tensor learning in python. *The Journal of Machine Learning Research*, 20(1):925–930, 2019.

[10] Antoine Liutkus, Derry Fitzgerald, Zafar Rafii, Bryan Pardo, and Laurent Daudet. Kernel additive models for source separation. *IEEE Trans. on Signal Processing*, 62(16):4298–4310, 2014.

[11] Patricio López-Serrano, Christian Dittmar, Jonathan Driedger, and Meinard Müller. Towards modeling and decomposing loop-based electronic music. In *Proc. of the ISMIR*, pages 502–508, New York, NY, USA, 2016.

[12] Patricio López-Serrano, Christian Dittmar, and Meinard Müller. Finding drum breaks in digital music recordings. In *Proc. of the International Symposium on Computer Music Multidisciplinary Research*, pages 68–79, Matosinhos, Portugal, 2017.

[13] Brian McFee, Colin Raffel, Dawen Liang, Daniel Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proc. of the Python in Science Conference*, 2015.

[14] Morten Mørup, Lars Kai Hansen, and Sidse M Arnfred. Algorithms for sparse nonnegative tucker decompositions. *Neural Computation*, 20(8):2112–2131, 2008.

[15] Daniel Müllensiefen, Bruno Gingras, Jason Jiří Musil, and Lauren Stewart. The musicality of non-musicians: An index for assessing musical sophistication in the general population. *PLOS One*, 9(2), 2014.

[16] Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel PW Ellis. mir_eval: A transparent implementation of common MIR metrics. In *Proc. of the ISMIR*, pages 367—372, Curitiba, Brazil, 2014. Citeseer.

[17] Zafar Rafii, Antoine Liutkus, and Bryan Pardo. REPET for background/foreground separation in audio. In G. R. Naik and W. Wang, editors, *Blind Source Separation*, Signals and Communication Technology, pages 395–411. Springer-Verlag, 2014.

[18] Zafar Rafii, Antoine Liutkus, and Bryan Pardo. A simple user interface system for recovering patterns repeating in time and frequency in mixtures of sounds. In *2015 IEEE ICASSP*, pages 271–275. IEEE, 2015.

[19] Mikkel N. Schmidt and Morten Mørup. Nonnegative matrix factor 2-d deconvolution for blind single channel source separation. In *International Conference on Independent Component Analysis and Signal Separation*, pages 700–707. Springer, 2006.

[20] Prem Seetharaman and Bryan Pardo. Simultaneous separation and segmentation in layered music. In *Proc. of the ISMIR*, pages 495–501, New York, NY, USA, 2016.

[21] Paris Smaragdis. Non-negative matrix factor deconvolution: Extraction of multiple sound sources from monophonic inputs. In *Independent Component Analysis and Blind Signal Separation*, volume 3195 of *Lecture Notes in Computer Science*, pages 494–499. Springer-Verlag, Berlin, Heidelberg, 2004.

[22] Jordan B. L. Smith and Masataka Goto. Nonnegative tensor factorization for source separation of loops in audio. In *Proc. of the IEEE ICASSP*, pages 171–175, Calgary, AB, Canada, 2018.

[23] Marinka Zitnik and Blaz Zupan. Nimfa: A python library for nonnegative matrix factorization. *Journal of Machine Learning Research*, 13:849–853, 2012.

QUANTIFYING DISRUPTIVE INFLUENCE IN THE ALLMUSIC GUIDE

Flavio Figueiredo
Universidade Federal de Minas Gerais
flaviovdf@dcc.ufmg.br

Nazareno Andrade
Universidade Federal de Campina Grande
nazareno@computacao.ufcg.edu.br

ABSTRACT

Understanding how influences shape musical creation provides rich insight into cultural trends. As such, there have been several efforts to create quantitative complex network methods that support the analysis of influence networks among artists in a music corpus. We contribute to this body of work by examining how disruption happens in a corpus about music influence from the All Music Guide. A disruptive artist is one that creates a new stream of influences; this artist builds on prior efforts but influences subsequent artists that do not build on the same prior efforts. We leverage methods devised to study disruption in Science and Technology and apply them to the context of music creation. Our results point that such methods identify innovative artists and that disruption is mostly uncorrelated with network centrality.

1. INTRODUCTION

What is disruption? To understand the concept, let us consider the careers of two famous Jazz musicians whose careers started in the 1940's: Bud Powell and Sun Ra. According to the *All Music Guide* [1], both artists have been highly influential. The AllMusic biography of Bud Powell states that: "One of the giants of the jazz piano, Bud Powell changed the way that virtually all post-swing pianists play their instruments.". Similarly, the guide describes Sun Ra as a major innovator, both in his music and in his style: "[Sun Ra] surrounded his adventurous music with costumes and mythology that both looked backward toward ancient Egypt and forward into science fiction.".

Disruption, as explored in this paper, is focused on differentiating artists like these even though they share similar backgrounds. While both artists may seem alike in terms of influence, the network of future artists citing them as a past influence sets these two great Jazz musicians apart by their network structure. Again according to the AllMusic guide, both artists have Thelonious Monk and Art Tatum as former influences. However, while future musicians influenced by Bud Powell also cite these two artists as influences, those following Sun Ra do not cite Thelonious Monk and Art Tatum; they are mostly influenced by Sun Ra in isolation when compared to Sun Ra's past influences. In this sense, Sun Ra is primarily self-sufficient, and as a consequence, disruptive. In contrast, Bud Powell's contribution is more related to developing and consolidating an ongoing field of work. Figure 1 depicts this concept.

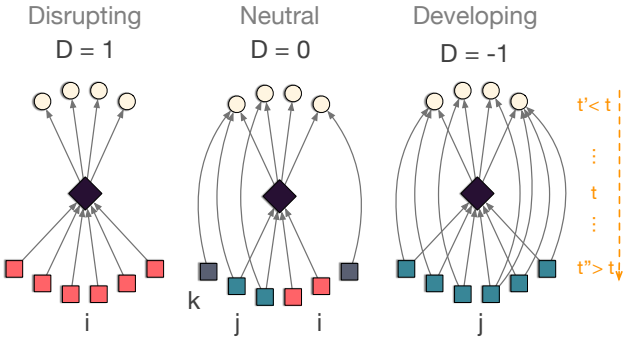


Figure 1: Reference networks of different focal works (diamonds), with their preceding references (circles) and posterior work (squares). i nodes (in red) reference the focal work but none of its predecessors; j nodes (in blue) reference both the focal node and its predecessors, and k nodes (in grey) link only to the focal node's references.

The figure shows an influence network and exemplifies disruption from the standpoint of the central, diamond-shaped, focal work. Links represent influence or citations. In this network, an overall influential innovator will be a node with a high in-degree, and both Bud Powell and Sun Ra fit this definition. In contrast, a disruptive node ($D = 1$) is singled out and thus self-sufficient compared to it's past. In our example, the focal node is cited by other nodes that tend to refer only to this single node as an influence.

To formalize the metric, let us call the focal, diamond-shaped node, as a (for artist). There are n_i nodes that reference a's work and at least one of its predecessors, while n_j nodes reference a but none of its predecessors. There are also n_k nodes do not reference a but reference at least one of its predecessors. Funk and Owen-Smith's [5] disruption index, here called D, is measured as:

$$D = \frac{n_i - n_j}{n_i + n_j + n_k} \quad (1)$$

D ranges from -1 to 1. The negative extreme captures a developing work, one that is mostly cited in conjunction with its own influences (i.e., $n_i = 0$ and $n_k = 0$). The positive extreme captures disruption (i.e., $n_j = 0$ and $n_k = 0$).

[1] http://www.allmusic.org. Quotations from June 16th 2019.

Motivated by recent demonstrations of the utility of D in other fields [5, 18], our work uses this index to examine disruptiveness of music artists using the AllMusic Guide. While several authors have tackled the task of understanding innovative artists, songs, and lyrics [2–4, 12, 15, 16], to the best of our knowledge, no prior effort exists that explores the disruptiveness of artists.

We opt to explore the AllMusic Guide as such a dataset has been used as a *gold standard* to other methods focused on influence [12, 16]. The guide contains a human-curated network of artists that influenced one another. This network provides us the contrasting example of Bud Powell and Sun Ra as a motivator to the importance of considering disruption. While both artists are influential, analyzing disruption unveils that one of them consolidates a stream of influences, while the other destabilizes this stream, shifting attention towards a different direction.

Our main contributions are (i) providing evidence that disruption measures provide insight when analyzing music corpora based on artist metadata, and (ii) describing disruption topologies that characterize how disruption happens in different contexts included in our corpus.

2. BACKGROUND AND RELATED WORK

Before presenting our dataset and results, we discuss previous work that explored different corpora to understand musical influence (Section 2.1). Next, we describe the metric that captures disruption (Section 2.2).

2.1 Influence Analysis in Music Corpora

Understanding musical influence is not a new endeavor [2–4, 12, 13, 15–17]. Several works have leveraged large datasets of audio and metadata to investigate historical trends in music creation quantitatively. Notably, Serrà and collaborators [15] use the Million Song Dataset to unveil historical regularities and changes in pitch transition, timbre usage, and loudness in pop music.

Several researchers have used the Billboard charts of songs most played on radios and streaming as a corpus representing western pop music. The audio and metadata about songs and artists in these charts have been used to characterize trends related to innovation, for example in lyrics [2], songs [16], and artists [16]. In particular, we point out the work of Mauch et al. [11] used timbral, tone, and harmonic information to analyze the sonic dynamics in the Billboard charts from 1960 to 2010. Their results point to three historical inflection points in the evolution of this corpus: 1964, 1983 and 1991.

A complementary approach to the analysis of aggregate trends is to examine individual artists or songs who have innovated in their context. Shalit et al. [16] use a dynamic topic model learned from audio and metadata to evaluate influence and innovation in songs from the Billboard charts. Their model identifies innovative songs and periods and suggests that overall, there is no correlation between how innovative and how influential a song is, with exceptions during the early 70s and mid-90s. For this analysis, Shalit operationalizes innovative songs as songs hard to account for by a model trained with data from the past. On the other hand, a song is influential if its language is used by subsequent work. Authors have also studied influence for particular settings such as Electronic Dance Music [4].

2.2 Measuring Disruptive Influence/Innovation

Our work is inspired by the network measure proposed by Funk and Owen-Smith [5] to study technological change. Funk and Owen-Smith propose a network model and the D index, which "quantifies the extent to which an invention consolidates or destabilizes the subsequent use of the components on which it builds" [5]. The index (detailed in Eq (1)) is built on the notion that disruption should be measured by more than the number of references an invention has in subsequent work. Besides that, a measure of disruption must consider the structure of previous and subsequent work that form the context of the invention.

In their original work, the authors leverage a comprehensive database of patents to validate that their index quantifies how consolidating or destabilizing inventions are and that this information is uncorrelated with the sheer impact of innovations. More recently, Wu et al. [18] validate this same index in datasets of scientific papers and software products. Moreover, Wu et al. show that in the context of papers, software, and patents, disruptive innovation is more associated with smaller teams and work that cites prior efforts further in the past.

3. METHODOLOGY

To describe how we measure disruption in the AllMusic guide, we first detail how we identified artist pages and influence edges from the AllMusic website (Section 3.1).

Different from other datasets where disruption has been measured [5, 18], the AllMusic guide presents a human-curated graph of influences. While from one perspective this is an advantage (e.g., provides explicit opinions of music editors), it is apparent that the information about influences is not complete, and it is likely less complete than in datasets created from of patents and scientific papers. One other particularity of our network is that it widely suffers from biases towards modern occidental musicians who achieved considerable success. To tackle the disadvantages related to sparse counts, we employ a Bayesian approach to measure disruption (Section 3.2).

3.1 Crawling the AllMusic Guide

AllMusic is a comprehensive catalog of artists, albums and songs. The AllMusic website contains Artist Profile pages that detail an artist's biography, discography, genres, styles and links to other related artists, among other information. The list of related artists details those that are similar, have influenced, followed, or have worked with the owner of the profile page. Artists said to influence a given artist according to the site are those "that have had a direct musical influence on, or were an inspiration to, the selected artist,

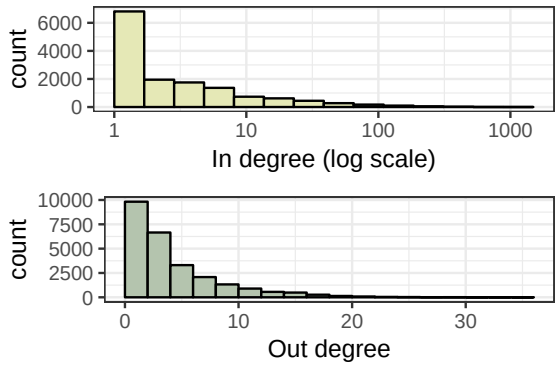


Figure 2: Distribution of in and out node degree in the AllMusic influences dataset. For in degrees, 18,281 nodes with zero inbounding edges are ommitted.

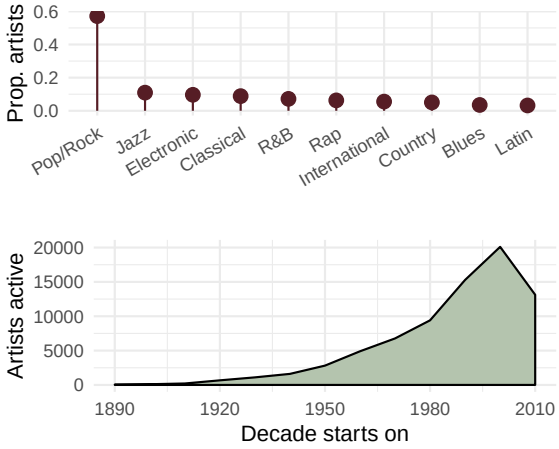


Figure 3: Proportion of artists with either in or out degree different from zero tagged with the 10 most popular genres, and number of artists active per decade.

as determined by our music editors"[2]. Being a human-curated graph, there are several situations an artist influences a contemporary musician, band or singer.

We use AllMusic to create a network of influences among artists. Data were obtained by exhaustively crawling the website. The crawler started with a list of approximately 73,000 thousand AllMusic URLs present in the open MusicBrainz[3] database. We note that not all of these URLs were valid artist page addresses (e.g., some were wrong or deleted links). Nevertheless, we crawled the correct ones and followed their links in a snowball approach [7]. In particular, we followed links to every related artist until crawling we found no new artists. Even though we only use the influence edges in our analysis, to gather as much artists as possible, we crawled all of the similar, influenced, followed, member of egdes.

For each visited artist page, the crawler saves the artist's name, decades of activity, genre, style, and list of influencers. The resulting set of artists from the crawler has 162,971 members, connected by 119,961 directed edges.

[2] https://www.allmusic.com/faq/topic/influencedby
[3] https://musicbrainz.org/

	Cited a	Did **not** cite a
Cited a's influences	n_j	n_k
Did **not** cite a's influences	n_i	everything else

Table 1: 2x2 Contigency Table used for Computing D

After filtering out artists with no influencers cataloged in AllMusic, our dataset comprises of 32,568 artists connected by 119,961 directed links, where a link from artist a to b denotes that a has been influenced by b. When we consider the weakly connected components of the graph, 96% (3,1279) of the nodes are in the giant component. This indicates that there is an undirected path of influence between most nodes. This is expected as major hubs, such as The Beatles, will lead to a mostly connected graph.

To characterize our graph, the in and out-degree distributions of nodes are shown in Figures 2. Complementary, in Figure 3 we show the genre distributions and active decades of the artists. Both the distribution of in and out degrees are skewed, with the distribution of in degrees being considerably more skewed and spanning a more extensive range. It is likely that influences (outgoing edges) of an artist are entered manually by editors and are kept to customary size. Bands or musicians influencing most artists are The Beatles (indegree 1,492), Bob Dylan (784), and The Rolling Stones (636), and there are 18,281 artists with no incoming edges. On the other hand, those with most extensive lists of influencers (outdegree) are Grateful Dead (36), Sonic Youth (35) and Jimi Hendrix (35). Concerning genre and epoch, our sample is biased towards Pop/Rock and artists active from the 80s to the present.

3.2 Measuring Disruption in Sparse Data

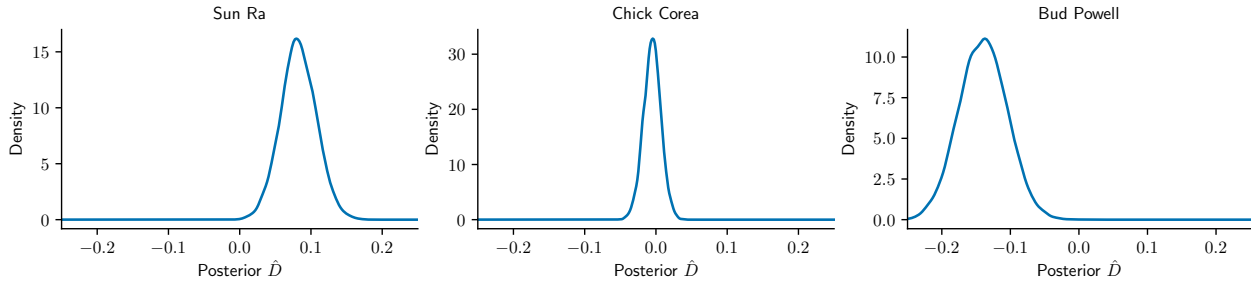
One challenge we tackle while computing disruption is how discuss results regarding D with statistical significance. Considering a focal artist a, recall that other artists may either: (1) cite a only, thus increasing n_i; (2) cite a and a's past influences, thus increasing n_j; (3) cite a's past influences only, increasing n_k; or, (4) do cite a nor past influences. These choices are shown Table 1.

In some of our initial exploratory analysis, we computed disruption in the AllMusic influence graph *as is* (i.e., with no filters nor priors), and found that Eq. (1) would lead to either very high ($D \approx 1$) or very low ($D \approx -1$) scores when n_i, n_j and n_k were very small. One example is the extreme case where $n_i = 1$, $n_j = n_k = 0$. Here, the metric will unveil a biased D due to the small numbers.

To explain our Bayesian approach, initially note that Eq. (1) captures the difference between two proportions:

$$D = p_i - p_j = \frac{n_i}{n_i + n_j + n_k} - \frac{n_j}{n_i + n_j + n_k} \qquad (2)$$

Here p_i, p_j and p_k (unused) are proportions. Furthermore, the counts may be captured by a Multinomial distribution $I, J, K \sim Multinomial(p_i, p_k, p_k, n)$, where $n = n_i + n_k + n_j$. Here, I, J and K are random variables modelling the respective counts n_i, n_j and n_k.



(a) Over 95% Credible and Disruptive (b) Neutral (c) Over 95% Credible and Developing

Figure 4: Posterior $\hat{D}$ for three Jazz Artists. On the left we have a disruptive were 95% of posterior samples are above zero; in the middle a neutral artist; on the left a developing artist where 95% of posterior samples are below 0.

To model D, one approach would be to use the closed form for the probability mass function for $D = I - J$. One negative aspect of this approach is that it also requires some assumption on the joint distribution of I and J or that both are independent. Another approach would be use statistical tests (see [1] for details). Here, classical approaches like the Binomial test for proportions have issues with small samples or assume independence in the choices that lead to the contingency table. Other options such Fisher's or McNemar's test focus on comparing either rows/columns or the off-diagonal of the 2x2 table. In our setting, n_j shares a column with n_i (see Table 1). Our Bayesian approach, discussed next, has the advantage that it that does not require such closed forms or assumptions.

Given that proportions captured by a Multinomial distribution, for each node of the graph, we can apply a conjugate Dirichlet [6] prior on such a Multinomial distribution. Being a conjugate prior, the posterior will also be a Dirichlet distribution from which we can sample proportions: $\hat{p}_i, \hat{p}_j, \hat{p}_k \sim Dirichlet(n_i + \alpha_i, n_j + \alpha_j, n_k + \alpha_k)$. Here, α_i, α_k, and α_j are prior hyper-parameters. These can be fine tuned by an analyst to capture prior beliefs. By sampling from this distribution, we are left with a posterior estimate of disruption that is defined as: $\hat{D} = \hat{p}_i - \hat{p}_j$.

Suppose we perform 10,000 of such samples. Let, $\hat{\boldsymbol{D}}$ be the vector os estimates. The average score of this vector will lead to similar results as the original one ($mean(\hat{\boldsymbol{D}}) \approx D$). However, using these samples, we are able to measure the *credibility* of our estimates [6]. This credibility comes from what is called the *credible-interval*, a Bayesian analogous of the confidence-interval. While a confidence-interval measures the probability that some true population statistic will fall into the range of the interval, the credible-interval is determined by the posterior samples.

To explain how we capture credibility, consider the case where $n_i = 1$ and $n_j = n_k = 0$. Moreover, consider the particular choice priors (discussed later), $\alpha_i = \alpha_k = \alpha_j = 10$. If we measure the fraction of posterior samples greater than zero $P(\hat{D} > 0)$, this value is only of 0.58, even if $p_i = 1$ and $p_k = 0$. Thus, our estimate is 58% credible. Credibility is thus captured by: (1) $P(\hat{D} > 0)$ when $D > 0$; and, (2) $P(\hat{D} < 0)$ when $D < 0$. In other words, simply the fraction of posterior samples when D that are either positive of negative depending on the value of D. If this fraction is above 0.95, we are over 95% credible for either the positive (disruptive) or negative (developing) case. Figure 4 shows three examples of posterior samples of $\hat{D}$ for three artists, illustrating the cases when of disruption and development, and neutrality.

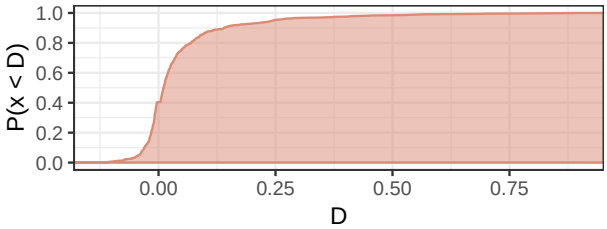


Figure 5: CDF of the disruption D for artists where our estimation has a minimum confidence of 0.95.

We set our hyper-parameters to the non-informative case of $\alpha_i = \alpha_j = \alpha_k = 10$. This choice is based on synthetic samples, where we estimated our credibility scores for different values of $n_i - n_j$ and $n_j - n_i$. Via simulations using different values of n_i, n_j and n_k, we found that with these our choices credibility is over 95% only when $|n_i - n_j| \approx 10$. Moreover, one can notice that these priors do not bias results towards positive nor negative values of $\hat{D}$ (i.e., $\alpha_i = \alpha_j$). We argue that this is adequate as it imposes a minimum difference between n_i and n_j to have some credibility in our estimates. Furthermore, we present disruption values D only in cases where our credibility is over 95%, using 10,000 samples per node.

Finally, being a human-curated guide, some artists will suffer from missing given the limited knowledge of the AllMusic editors. To avoid discussing such cases, we limit the analysis to nodes with at least three incoming and outgoing edges *after disruption is computed*.

4. DISRUPTIVE ARTISTS

We now explore the most and least disruptive artists in our data. Figure 5 shows the distribution of disruptiveness D. This distribution is concentrated around a median value close to zero (0.01), with a longer right tail – there are more highly destabilizing artists than highly consolidating ones.

Artist	D	AM Genre	n_i	$n_j + n_k$
King Sunny Ade	0.90	International	9	1
Édith Piaf	0.86	Vocal	51	6
Frankie Knuckles	0.70	Electronic	30	13
The Clark Sisters	0.68	Religious	13	6
Mstislav Rostropovich	0.68	Classical	13	6
Los Tigres del Norte	0.57	Latin	14	9
John Cage	0.56	Classical	141	64
Tiësto	0.45	Electronic	10	12
Alfred Brendel	0.45	Classical	13	16
Scott Asheton	0.41	Pop/Rock	16	18
Bernard Herrmann	0.41	Stage & Screen	23	28
Converge	0.40	Pop/Rock	32	45
K.M.D.	0.38	Rap	9	15
Too $hort	0.36	Rap	70	104
Darkthrone	0.35	Pop/Rock	15	28

Table 2: Most disruptive artists in AllMusic with at least three influences catalogued.

Artist	D	AM Genre	n_i	$n_j + n_k$
Teddy Wilson	-0.13	Jazz	7	163
John Coltrane	-0.11	Jazz	103	1307
Bud Powell	-0.11	Jazz	23	358
Philly Joe Jones	-0.10	Jazz	8	121
Geto Boys	-0.10	Rap	33	474
Sarah Vaughan	-0.09	Jazz	23	495
Pete Seeger	-0.09	Folk	22	335
Sonny Rollins	-0.08	Jazz	31	641
The Stanley Brothers	-0.08	Country	11	308
Augustus Pablo	-0.08	Reggae	1	212
Buddy Guy	-0.07	Blues	8	611
Roy Acuff	-0.07	Country	25	204
Jimmy Reed	-0.07	Blues	32	515
Oscar Peterson	-0.07	Jazz	15	411
Master P	-0.07	Rap	12	530

Table 3: Most consolidating artists in AllMusic with at least three influences catalogued.

Overall, we also find disruption does not correlate with the influence of an artist. This measure is captured here by the number of other artists influenced by a certain artist (in-degree). This was measured using the linear correlation coefficient is $\rho = -.001$. Correlations between disruption and centrality are further detailed in the next section.

Next, in Table 2 show the 15 most disruptive in our dataset. None of the most disruptive artists are among the most influential of AllMusic, and their communities in the network are very diverse. For example, King Sunny Ade is a Nigerian musician credited for a significant contribution in the popularization of juju music worldwide. According to AllMusic, he has been influenced by other juju and highlife artists such as I. K. Dairo & His Blue Spots and Rex Lawson, while he has influenced artists from a diverse stream including Talking Heads and Trey Anastasio, both from the US. Édith Piaf is a famous French singer with a similar network structure, bridging influences from earlier French singers and an assorted group of followers spanning several decades and countries.

The remaining artists in Table 2 illustrate multiple other types of destabilizing innovation in various genres. For example, Frankie Knuckles and Too $hort are acknowledged as pioneers of house music and gangsta rap, respectively.

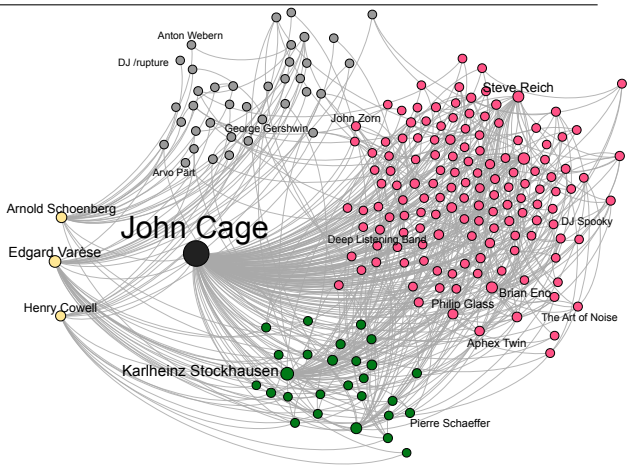


Figure 6: Network with John Cage as the focal node. Preceding artists are in yellow, while i, j and k nodes are pink, green and grey, resp.

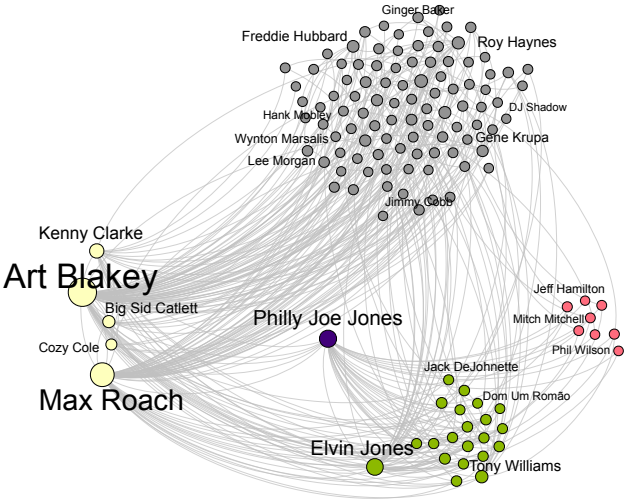
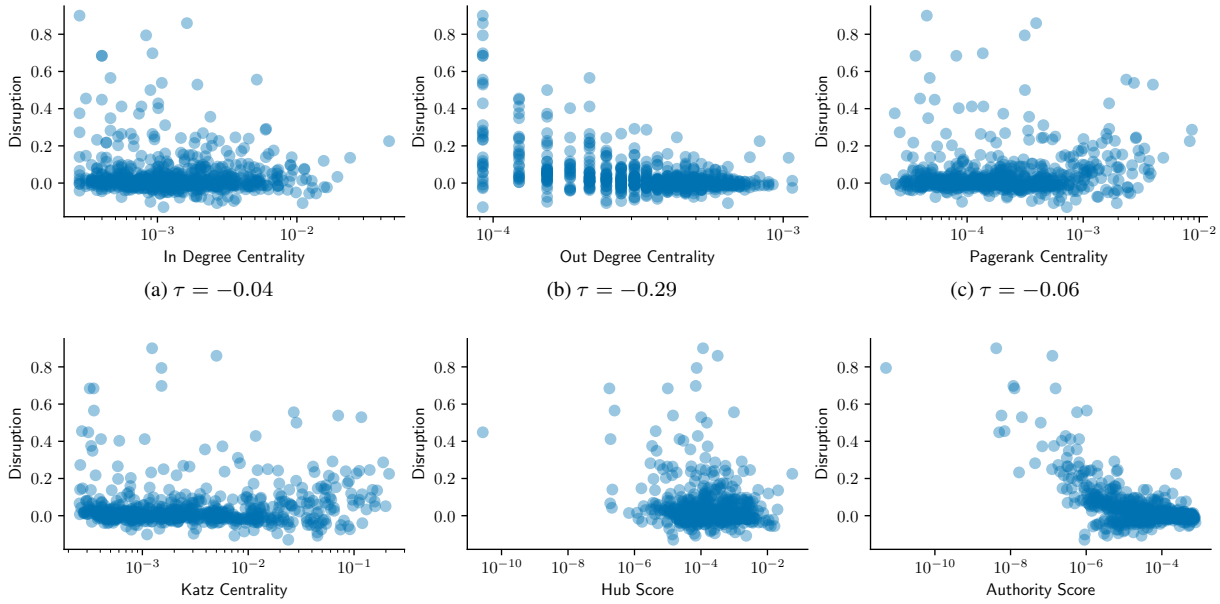


Figure 7: Network with Philly Joe Jones as the focal node. Preceding artists are in yellow, while i, j and k nodes are pink, green and grey, resp.

The most influential artist in this list is John Cage, a highly inventive composer who, according to AllMusic, has influenced generations of composers, writers, dancers, and visual artists. Figure 6 shows how the three classical composers cataloged as influences of John Cage are not influences of many of his followers.

On the opposite side of the spectrum of disruption, Table 3 lists the 15 most consolidating artists, according to D. There is a predominance of jazz artists, who are 6 of the ten most consolidating artists. Jazz instrumentalists and singers whose career started after the 1930s are often characterized by our method as consolidators building on a stream of shared influences. These artists share a set of influences with many others. Figure 7 illustrates one such case for the jazz drummer Philly Joe Jones, who shares the influences of Art Blakey and Max Roach with a large number of other jazz drummers, including most subsequent work. Geto Boys and Augustus Pablo have a similar neighborhood structure in the Rap and Reggae genres.



(a) $\tau = -0.04$ (b) $\tau = -0.29$ (c) $\tau = -0.06$

(d) $\tau = -0.01$ (e) $\tau = -0.11$ (f) $\tau = -0.36$

Figure 8: Correlation of Disruption with six node importance measures.

Overall, the examination of artists identified as most and least destabilizing points to the expressiveness of this method. This approach highlights artists who have not necessarily influenced a large number of other artists, so this information is not readily available based on a direct quantification of impact. Moreover, our face validity analysis promptly links high valuations of disruptiveness with widely known stories of innovation. Most interestingly, these stories are diverse in their geography, genre, and epoch, even if mined from a considerably biased dataset.

5. DISRUPTION VS CENTRALITY

We now investigate to what degree the disruption scores provides information that is not already available through other metrics of network topology. To do so, we measure six different node importance scores using our full graph: the Indegree Centrality (normalized indegree), Outdegree Centrality (normalized outdegree), Pagerank [14], Katz [8] Centrality, Hub Scores [10], and Authority Scores [10].

We correlated each score with the disruption index of artists. Figure 8 presents the relation of each score with D together with Kendall's rank correlation (τ) for each case. We resort to Kendall's coefficient, τ, as it addresses ties (e.g., nodes with the same in degree) and is able to uncover both linear as well as non-linear relationships [9].

The patterns in the figure and τ scores point that most metrics do not correlate with disruptiveness. The only cases were moderate negative correlations were uncovered were: Out degree centrality ($\tau = -0.29$) and Authority score ($\tau = -0.36$). A small negative relation also exists for Hub scores ($\tau = -0.11$). Nevertheless, these scores are moderate at best. Such a result is relevant, as it shows that disruptiveness values are not easy to recover using standard node scores from complex networks.

In summary, our results in this section combined with our discussion in the previous section, point to the relevance of measuring disruptiveness. The D index unveils insightful patterns on the AllMusic corpus that are not trivially explained by other network centrality scores.

6. DISCUSSION AND FUTURE WORK

In this paper, we present the first in-depth analysis of disruption in a music corpus. More importantly, we argue in favor of a Bayesian disruption index. While our analysis is limited to a dataset, the approach we here discuss is general enough to be used in other settings. In particular, we note data analysts may tune choices of hyper-parameters to their prior-beliefs for different datasets.

Our contributions bring two main implications. First, our examination of the validity of the disruption index suggests it can be applied to music corpora. Second, our analysis of disruptive artists in the AllMusic guide shows new information about artists in this guide that may be taken into account in musicological analyses.

At the same time, both of these directions call for relevant future work. In particular, further validation of the disruption index with other contexts seems very relevant, to understand its applicability to other musical traditions or to networks formed by albums or songs, for example. In complement to this direction, musicological analyses that use disruption to provide deeper insight leveraging this data is necessary to further validate the usefulness of the approach we here presented.

Reproducibility: We point out that our source code for data collection, analysis and for the figures and tables in this paper, as well as the dataset used, is available at: `http://github.com/flaviovdf/allmusic-disruption`.

Acknowledgements: This work has been partially supported by the project ATMOSPHERE (atmosphere-eubrazil.eu), funded by the Brazilian Ministry of Science, Technology and Innovation (Project 51119 - MCTI/RNP 4th Coordinated Call) and by the European Commission under the Cooperation Programme, Horizon 2020 grant agreement no 777154. Funding was also provided by the authors' individual grants from CAPEs and CNPq. In particular, CNPQ's Universal 2018 Grant: 421884/2018-5.

7. REFERENCES

[1] Alan Agresti. *An introduction to categorical data analysis*. Wiley, 2018.

[2] Jack Atherton and Blair Kaneshiro. I said it first: Topological analysis of lyrical influence networks. In *ISMIR*, 2016.

[3] Nicholas J Bryan and Gen Wang. Musical influence network analysis and rank of sample-based music. In *ISMIR*, 2011.

[4] Nick Collins. Influence in early electronic dance music: An audio content analysis investigation. In *ISMIR*, 2012.

[5] Russell J Funk and Jason Owen-Smith. A dynamic network measure of technological change. *Management Science*, 63(3):791–817, 2016.

[6] Andrew Gelman, Hal S Stern, John B Carlin, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

[7] Leo A Goodman. Snowball sampling. *The annals of mathematical statistics*, pages 148–170, 1961.

[8] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[9] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[10] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5), 1999.

[11] Matthias Mauch, Robert M MacCallum, Mark Levy, and Armand M Leroi. The evolution of popular music: Usa 1960–2010. *Royal Society open science*, 2(5), 2015.

[12] Brandon G Morton and Youngmoo E Kim. Acoustic features for recognizing musical artist influence. In *ICMLA*, 2015.

[13] Erik Noyes, I Elaine Allen, and Salvatore Parise. Artistic influences and innovation in the popular music industry. *Frontiers of Entrepreneurship Research*, 30(15), 2010.

[14] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[15] Joan Serrà, Alvaro Corral, Marián Boguñá, Martin Haro, and Josep Lluís Arcos. Measuring the evolution of contemporary western popular music. *Scientific reports*, 2, 05 2012.

[16] Uri Shalit, Daphna Weinshall, and Gal Chechik. Modeling musical influence with topic models. In *ICML*, 2013.

[17] Jay Summach. The structure, function, and genesis of the prechorus. *Music Theory Online*, 17(3), 2011.

[18] Lingfei Wu, Dashun Wang, and James A Evans. Large teams develop and small teams disrupt science and technology. *Nature*, 566(7744):378, 2019.

LEVERAGING KNOWLEDGE BASES AND PARALLEL ANNOTATIONS FOR MUSIC GENRE TRANSLATION

Elena V. Epure
Deezer R&D

Anis Khlif
Deezer R&D

Romain Hennequin
Deezer R&D

research@deezer.com

ABSTRACT

Prevalent efforts have been put in automatically inferring genres of musical items. Yet, the propose solutions often rely on simplifications and fail to address the diversity and subjectivity of music genres. Accounting for these has, though, many benefits for aligning knowledge sources, integrating data and enriching musical items with tags. Here, we choose a new angle for the genre study by seeking to predict what would be the genres of musical items in a target tag system, knowing the genres assigned to them within source tag systems. We call this a translation task and identify three cases: 1) no common annotated corpus between source and target tag systems exists, 2) such a large corpus exists, 3) only few common annotations exist. We propose the related solutions: a knowledge-based translation modeled as taxonomy mapping, a statistical translation modeled with maximum likelihood logistic regression; a hybrid translation modeled with maximum a posteriori logistic regression with priors given by the knowledge-based translation. During evaluation, the solutions fit well the identified cases and the hybrid translation is systematically the most effective w.r.t. multilabel classification metrics. This is a first attempt to unify genre tag systems by leveraging both representation and interpretation diversity.

1. INTRODUCTION

Music genres have been long studied as semantic dimensions of artists and tracks [8]. Rooted in musicology, music experts have mainly undertaken this endeavour. With digitization of music and prevalence of Internet music consumption, online communities have also shown increasing interest in annotating musical items with genres (e.g. creating folksonomies such as Lastfm). In addition, crowdsourced, web-based encyclopedias that describe and structure music-related knowledge including genres, have been created and openly disseminated [4, 37, 41].

Apart from ontologically describing musical items, genres are also among the most common attributes of tracks, albums and artists to which the users of music streaming services relate [21]. Users resort to genres to discover music, create playlists, define their profiles, foster interactions with other users, etc. Hence, being able to correctly infer music genres as metadata is central to such tasks.

Music genre is a challenging concept to model and highly subjective. Past studies [11, 18, 33, 36] convey how difficult it is to agree upon shared definitions and interpretations, even for popular genres. People interpret genres differently, influenced by their culture, personal preferences or acquired musicological knowledge [11, 18, 33]. Genre representations within tag systems vary [31] with respect to: the level of detail (how specialized genres can get); the coverage (which genres are considered); the genre interpretation (*pop/rock* could be distinctly defined and interpreted across sources); how genres are related (*blues rock* is a subgenre of *rock*, but not of *blues* in the MuMu dataset [16, 22]). Divergences also result from the spelling variability (e.g. *alternative rock* vs. *alt. rock*).

The research question we address in this work is: given annotations with genre tag systems of multiple sources, how to infer the equivalent annotations within a target tag system? We refer to this as a translation task, but we do not necessarily seek to translate tags between languages.

When relying only on the definition of the sources and target tag systems, this task could be solved using taxonomy mapping [27, 29]. A taxonomy is a classification schema with concepts organized from general to specialized. The goal of taxonomy mapping is to align the concepts of the source and target taxonomies. Related works integrate commercial catalogues [24,29], align multi-lingual taxonomies [34,35,43] or restructure existing taxonomies [26,27,38] in supervised or unsupervised manners. Ontology mapping [23] is a similar task, in which additional relation properties and axioms can be exploited.

A solution focused on taxonomy mapping is nonetheless incomplete as it does not consider the application of the taxonomies in practice, which could reveal divergences in genre interpretation. Thus, we hypothesize that a robust translation is built not only on the definitions of genre tag systems, but also on their use for annotations. In accordance with the terminology of the Automatic Machine Translation domain [9], we call a corpus of items jointly annotated by multiple sources a parallel corpus.

The contribution of the current work is a translation system that effectively leverages knowledge-based and statistical methods for genre translation in three cases:

1. A cold-start case, when genre tag systems of the target

and sources are known, but there is no parallel corpus. We address this case with a Knowledge-Based (KB) system based on taxonomy mapping (Section 3).

2. Many parallel annotations are available allowing to learn mappings between genre interpretations (e.g. when some sources use *alternative rock* the target tends to use *alt. rock* and *indie rock*). To deal with this case, we use a simple linear multilabel classifier, namely a logistic regression model trained with Maximum Likelihood (ML) (Section 4.1).
3. The case in-between when less annotations are available and some target tags may be missing in the parallel corpus. We tackle this scenario with an hybrid Bayesian approach that leverages the KB translation as a prior for the logistic regression model trained with Maximum A Posteriori (MAP). This case, presented in Section 4.2, is the most general. Finding an effective solution for it has multiple positive implications for practice.

We release the code of these methods for reproducibility [1] .

The Music Information Retrieval (MIR) community has extensively studied the automatic genre annotation of musical items by exploiting the content (e.g. audio, lyrics) [10, 16, 22]. Other genre representations, tackled in [2, 20, 31, 41], create genre graphs from multiple knowledge sources. Yet, to our knowledge, there is no past work translating music genres from one tag system to another (e.g. from Discogs to Wikipedia) by leveraging the diversity of both genre representations and interpretations.

We resort to item annotation to assess the proposed translation methods. To reflect a real-life context [7], we consider a musical item annotated with multiple source tag systems; having multiple labels and not only broad genres such as *rock*, but also very detailed subgenres, which results in predicting among hundreds of possibilities. Lastly, combining multiple tag predictors in a Bayesian framework was done before [10, 39]. However, these works aggregate information from different predictors in the same tag system while we consider several tag systems.

2. NOTATIONS AND PROBLEM FORMULATION

In this work, we denote matrices by bold capital letters, $\mathbf{M}$; vectors by bold lower case letters, $\mathbf{v}$; the n-th row vector of matrix $\mathbf{M}$ by $\mathbf{m}_n$; scalars by italic lower case letters, x; the coefficient at row i and column j of matrix $\mathbf{M}$ by m_{ij}; the i-th element of vector $\mathbf{v}$ by v_i. Calligraphic font is used for sets of sets (e.g. $\mathcal{S}$) and capital letters for sets (e.g. S).

Let $\mathcal{D}$ be a set of tag systems, $\mathcal{S}$ a subset of $\mathcal{D}$, henceforth referred to as source tag systems, and $T \in \mathcal{D}, T \notin \mathcal{S}$ henceforth referred to as a target tag system. Further, we refer to a tag system as a tag set, but we stress that it may contain broader information such as relations between genres (e.g. taxonomies or ontologies). The research problem we address is: given $\mathcal{S}, T$ and a set of tag annotations (e.g. associated with a given musical item) taken from $\mathcal{S}$, what would have been the corresponding tag annotations if the tags had been taken from T. We note $S = \cup_{E \in \mathcal{S}} E$ the union of the source tag systems, and $|S|$ its cardinality.

The approach we adopt consists in defining a translation scoring function $f : \mathcal{P}(S) \rightarrow \mathbb{R}^{|T|}$, where $\mathcal{P}$ denotes partitions over S, that predicts translation scores for every target tag from a set of source tags. Estimating such a scoring function is a standard setting for multilabel classification.

3. KNOWLEDGE-BASED GENRE TRANSLATION

We propose a translation method based on multiple genre taxonomies brought together under a genre graph. Section 3.1 introduces the graph types of concepts and relations and presents the genre taxonomies. In Section 3.2, we show how we create the links between the genre taxonomies using advanced normalization and tokenization. In Section 3.3, we define the translation scoring function f by exploiting the genre graph structure and its relations.

3.1 Building a knowledge-based genre graph

We automatically derive an undirected genre graph by aggregating multiple genre tag systems (e.g. taxonomies, ontologies or social tags), created by either experts or non-experts as in [2, 20, 31, 34]. Its modular design allows to easily integrate new sources through a normalization pipeline that addresses much more variability of genre strings than the existing works [30, 41] (presented in Section 3.2). The knowledge sources used to build the current version of the genre graph are: DBpedia (English, 12443 genres), and Lastfm (327 genres), Tagtraum (296 genres) and Discogs (296 genres)–the taxonomies released in the 2018 MediaVal AcousticBrainz Genre Task [7]. The Discogs genre taxonomy is pre-defined by experts. The Lastfm and Tagtraum genre taxonomies are automatically inferred from social tags with the approach proposed by Schreiber [30], followed by a manual processing [7].

The types of relations between genres vary across sources. In DBpedia, the retrieved types for each genre are: subgenres, origins, aliases–various spellings of the same genre, and derivatives–genres which are influenced by this genre, but could not be considered subgenres. The other knowledge sources contain only subgenre relations.

Each genre tag system becomes a graph by adding a source node that connects all the genre tags as in Figure 1. Then, to connect these decentralized graphs, a normalized graph is produced from all available tags. Each original tag is connected to its normalized form in the normalized graph. The description of how we normalize genres and create the normalized graph is continued in Section 3.2.

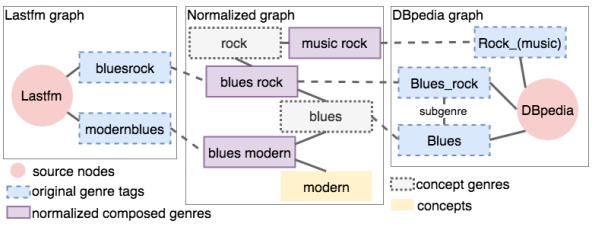


Figure 1: Genre graph extract. Dashed edges link the original genre tags to their normalized versions.

[1] https://github.com/deezer/MusicGenreTranslation

3.2 Normalizing genre tags

We create a more robust normalization pipeline compared to the related works [29,30,35,41] that, apart from basic tokenization and normalization, also separates words written together (e.g. *poprock* in *pop* and *rock*). The basic tokenization splits tags by non-alphanumeric characters (e.g. "-", "_"). The basic normalization converts tags to lower case and brings tags containing "&", "+" and "'n'" to the same form (e.g. *d+b*, *drum'n'bass* and *drum and bass*).

For the advanced tokenization, we use a modified trie [12] and a probabilistic tokenization built on Wikipedia unigrams [3]. A trie is a tree data structure that efficiently stores and retrieves strings. Each node has a char and a flag to mark if the path from the root to it forms a word. We modify the way we populate the trie as follows. At first, we sort the tokens obtained from the basic tokenization and normalization, ascendingly by length. Then, we add the tokens of DBpedia with less than l letters directly to the trie [2] . For the others, we attempt to split them using the trie and only the unknown words are added to the trie.

The tokenization using the trie is a recursive greedy algorithm that aims at matching the longest possible words in the trie. If a recursion fails, we explore the path with the next best previous word instead. If we assess the split output as incorrect, meaning that it results in too many short words, in short suffixes, or fails to split a large tag, then we use the probabilistic tokenization.

The probabilistic tokenization uses dynamic programming to find the words best maximizing their probability product. The frequency of each word, assuming that they are independently distributed, is approximated using the Zipf's law [44] to $\frac{1}{nlog(N)}$, where n is the word rank [42] and N is the total number of Wikipedia unigrams. We again assess the split output. Some extra conditions are added besides those presented in the previous paragraph: a Wikipedia split is incorrect if there are single letters as middle words and if no word is already contained in the trie [3] . If this tokenzation fails, we add the token as it is.

Finally, we transform the obtained tokens in nodes in the normalized graph (see Figure 1). There are three types of nodes: 1) normalized composed genres (e.g. *altern rock*, *deep house*), 2) concepts which are words that do not represent genres but are part of the name of multiple genres (e.g. *nu* in *nu jazz* and *nu metal*); 3) concept genres which are standalone genres but can be also part of composed genres (e.g. *punk* in *post punk*). If a genre is tokenized, its tokens are sorted and concatenated becoming a composed genre node as in [30] (e.g. *music rock* in Figure 1). This node is then connected to its concept and concept genre nodes.

3.3 Translating Genres through DBpedia Mapping

Using intermediate mapping spaces such as taxonomies or pivot languages has been explored in past works to match multi-lingual [35, 43], multi-cultural [27] or e-commerce [26, 29] taxonomies. Similar to [27], we use DBpedia, an ontology derived from Wikipedia infoboxes [19] as it has the highest genre coverage and quite high quality. However, to map a genre to DBpedia genres, we avoid using string similarity as it can be very noisy (e.g. *pop* vs. *bop*). Instead, we leverage genre knowledge to create a mapping strategy as we further present. Most related works rely on the structure of taxonomies for mapping the source and target concepts [24,27,29,35,43]. Our solution uses structural information too, but differently. Specifically, we use the neighbours of the source and target concepts and the structure of the directed DBpedia graph.

We map each genre of the source and target tag system, to the genres of the DBpedia ontology: $B \in \mathcal{D}$. We assume $B \notin \mathcal{S}$ and $B \neq T$. For each input tag system D, with $D \in \mathcal{S}$ or $D = T$, the output of the mapping is a matrix $\mathbf{Z}^D \in \mathbb{R}^{|D|\times|B|}$, where each row represents the relatedness of a genre tag from D to the DBpedia genres. We compute the mapping matrix $\mathbf{Z}^D$ by applying the following steps for each tag $s \in D$:

1. Normalize s with the process described in Section 3.2 (*e.g. Rock/Pop* becomes *pop rock*).
2. Check if the normalized s equals any normalized genre of B. If true, all entries in $\mathbf{Z}_D$ linked to the DBpedia aliases of the found genres are set to 1 and all others to 0 (e.g. *acid house* is mapped to *Acid_house*, with aliases *Acid_(electronic_music)*, *Warehouse_music*, etc.).
3. If the normalized s is not in B, then map it using its context genres in D: compound s with each parent tag in D and check if the normalized compounded tag equals any normalized genre of B (inspired from [43]). If true, proceed as in Step 1. (e.g. *stoner* has parent *rock* in Lastfm; search by *rock stoner* and map it to *Stoner_rock*).
4. If Steps 2 and 3 are unsuccessful, consider two cases:
   (a) s is a concept genre as defined in Section 3.2. First, retrieve the DBpedia directed subgraph composed of the nodes which contain the normalized s as a substring in their normalized form. Second, map s to the nodes with the highest in-degree centrality [5] in this subgraph. The intuition is that concept genre nodes are more likely fundamental music genres; hence they tend to have many subgenres or related genres. Third, assign to the selected DBpedia genres and their aliases a score of 1 divided by the number of selected nodes, and to the others 0 (e.g. *rock* does not exist as is in DBpedia. To map it, we retrieve all tags that contain it such as *Punk_rock*, *Art_rock*, *Rock_music*, etc. We observe that *Rock_music* is the most connected node in the subgraph with the genres containing *rock*. As only one node is selected, we assign to it and its aliases a score of 1).
   (b) s is a composed genre as defined in Section 3.2. First, select from the normalized genres in B those that share the greatest number of words with s. Second, select from this list, the genres with the highest number of shared concept genres–if it is 0, then the initial selection is kept unchanged. Third, assign scores as in Step 4(a).

[2] DBpedia seeds the trie as it has the highest coverage and we set l=7.

[3] As we already added to the trie short genre and concept tags from multiple sources, we assume the probability of all words to be new is low.

5. For each genre in B associated to s in Steps 1–4, propagate half of the value of its score to its neighbors in B. The intuition is that parent genres or subgenres could be relevant and sometimes specified by other sources.

For each s not mapped in the previous process, we compute its scores by averaging the rows in $\mathbf{Z}^D$ of its related genres in the input taxonomy D (e.g. for *aor* which is not found in DBpedia, we compute the scores by assigning it the scores obtained for *rock*, its parent genre in Discogs). Finally, the relatedness of a source genre $s \in S$ and a target $t \in T$ is computed using cosine similarity between their corresponding rows $\mathbf{s} = \mathbf{z}_s^S$ and $\mathbf{t} = \mathbf{z}_t^T$ in the mapping matrices, $\mathbf{Z}^S$ and $\mathbf{Z}^T$. We define $\mathbf{W}^{KB} \in \mathbb{R}^{|T|\times|S|}$ such that $w_{ts}^{KB} = \frac{\mathbf{s}^T\mathbf{t}}{||\mathbf{s}||_2||\mathbf{t}||_2}$. The translation scoring function is:

$$f_t(\{s_1, s_2, \ldots, s_K\}) = \sum_{k=1}^{K} w_{ts_k}^{KB} = \mathbf{x}^T \mathbf{w}_t^{KB}, \quad (1)$$

where $\mathbf{x}$ is the binary encoded vector of $\{s_1, \ldots, s_K\}$.

4. DATA-INFORMED GENRE TRANSLATION

In this section, we consider that a parallel corpus is available and present two statistical approaches: ML that relies only on annotations (Section 4.1), and MAP that leverages the KB results as a prior knowledge (Section 4.2).

4.1 Maximum Likelihood logistic regression

In statistical approaches to the tag translation task, we seek to train a parametric mapping to model the probability $P(\mathbf{y}|\mathbf{x})$ of having a collection of target tags (encoded as a binary vector $\mathbf{y} \in \{0,1\}^{|T|}$) given the source tags (encoded as a binary vector $\mathbf{x} \in \{0,1\}^{|S|}$). We assume the independence of the target tags, and only seek to model the conditional probabilities $P(y_t|\mathbf{x})$. This comes down to training $|T|$ binary classifiers, also known as binary relevance. There are more elaborated settings for doing multilabel classification without the target tag independence assumption. We notably also tested classifiers chain [28], but it did not result in significant improvement over the results presented in Section 5.3, while increasing the system complexity. We propose to implement binary relevance with logistic regression [40]. Logistic regression models the probability of having the t-th target tag y_t given the source tags $\mathbf{x}$ and the parameters of the logistic regression $\theta = \{\mathbf{W}, \mathbf{b}\}$, $\mathbf{W} \in \mathbb{R}^{|T|\times|S|}$ $\mathbf{b} \in \mathbb{R}^{|T|}$; as:

$$P(y_t = 1|\mathbf{x}, \theta) = \sigma(\mathbf{w}_t^T\mathbf{x} + b_t) \quad (2)$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$. $\mathbf{W}$ is called the weights matrix and $\mathbf{b}$ the bias. Note that, for the statistical approaches, the scoring function f introduced in Section 2 is defined here as $f(\{s_1, s_2, \ldots, s_K\}) = \big(P(y_1 = 1|\mathbf{x}, \theta), \ldots, P(y_{|T|} = 1|\mathbf{x}, \theta)\big)$. To train a logistic regression model we maximize the log-likelihood of the targets, given the source tags, w.r.t. the parameters θ:

$$\mathcal{L} = \log P(\mathbf{Y}|\mathbf{X}; \theta) = \sum_{n=1}^{N} \mathbf{y}_n^T \log(\sigma(\mathbf{W}\mathbf{x}_n + \mathbf{b})) + (\mathbf{1} - \mathbf{y}_n)^T \log(\mathbf{1} - \sigma(\mathbf{W}\mathbf{x}_n + \mathbf{b})) \quad (3)$$

where N is the size of the parallel corpus; $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^T \in \{0,1\}^{N\times|S|}$; and $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_N]^T \in \{0,1\}^{N\times|T|}$. In practice the regularization term $\frac{1}{2}||\mathbf{W}||_F^2$ is added to $\mathcal{L}$ in the objective, where $||\cdot||_F$ denotes the Frobenius norm on matrices, to limit overfitting.

4.2 A unified translation model

While ML logistic regression can be expected to work well with large amounts of parallel annotations, they will not adapt well to settings where no or little parallel data is available. In a real-life scenario, the size of the parallel corpus can range from zero to tens of thousands of samples, which precludes systematically favoring one or the other. Defining a criterion for when to switch from KB to statistical translation is arduous since this criterion would depend on the number of source and target tags as well as on their distribution. Ideally, we would like to have knowledge-based performances when no parallel data is available, and a smooth way to transition towards more data-abundant settings. This leads us to consider the translation table $\mathbf{W}^{KB}$ given by the KB system as a prior in a Bayesian framework, using the MAP [6] objective. Instead of maximizing the likelihood of the target tags, given source tags and parameters, we maximize the posterior probability of the parameters given the source and target tags:

$$P(\theta|\mathbf{x}, \mathbf{y}) \propto P(\mathbf{y}|\mathbf{x}; \theta)P(\theta|\mathbf{x}) = P(\mathbf{y}|\mathbf{x}; \theta)P(\theta). \quad (4)$$

By assuming, for each target tag t a normal distribution for $\mathbf{w}_t$ centered around $\mathbf{w}_t^{KB}$ with a precision matrix $\mathbf{\Lambda} = \lambda^2\mathbf{I}$ (λ is independent of t), we can write the logarithm of the prior distribution as:

$$\log(P(\theta)) = \frac{\lambda^2}{2}||\mathbf{W} - \mathbf{W}^{KB}||^2 + \text{cte}. \quad (5)$$

We also consider a centered Gaussian prior on the bias (corresponding to a l^2 regularization). We then define:

$$\mathcal{L}_{\text{prior}} = \frac{\lambda^2}{2}||\mathbf{W} - \mathbf{W}^{KB}||^2 + \nu||\mathbf{b}||^2. \quad (6)$$

Using (3), (4) and (6), the final MAP objective becomes $\mathcal{L}_{\text{MAP}} = \mathcal{L} + \mathcal{L}_{\text{prior}}$, where the first term is the loss of Eqn (3), and the second can be seen as a regularization term on the weight matrix $\mathbf{W}$ that penalizes its straying away from the priors. $\mathcal{L}$ depends on the number of training samples, while $\mathcal{L}_{\text{prior}}$ does not. Therefore, $\mathcal{L}$ becomes the predominant term in the loss as the size of the training data grows, leading to an objective function very close to the one of the logistic regression of Section 4.1. Conversely, when little data is available, we can expect the performances to be close or better than those of the KB system.

When a large parallel corpus is available, we can choose λ with grid search on a validation set. This is computationally expensive, and does not adapt well when the parallel corpus is small. For the sake of adaptability, we hereby propose a principled way inspired by [15] to choose λ, that does not require a lot of data while achieving top results. The rationale builds on the limited effective range of the logistic regression parameters. A shift of 5 of w_{ts} in the logit scale can move the probability associated with the target tag t from 0.5 to 0.99 or from 0.01 to 0.5. Hence, we would tend to choose λ in such a way that bigger shifts in the predicted probability of the target tag, which is the result of the added shifts for each annotated source tag, are unlikely. If we note $\bar{N}_S$ the average number of source tags per sample (which can be estimated with a few samples), this would mean restricting the coefficients w_{ts} from shifting by more than $\frac{5}{\bar{N}_S}$. For a normal variable $X \sim \mathcal{N}(\mu, \frac{1}{\lambda})$ we have $P(X \in [\mu - \frac{2}{\sqrt{\lambda}}, \mu + \frac{2}{\sqrt{\lambda}}]) \approx 95\%$, we therefore propose to choose precision λ such that:

$$\frac{2}{\sqrt{\lambda}} \approx \frac{5}{\bar{N}_S} \tag{7}$$

5. EXPERIMENTS

We report the performances of the proposed models on a recording-based tag translation task. This also serves as an indirect evaluation of the DBpedia mapping, which, in a work dedicated to taxonomy mapping, could have been assessed by experts. Due to its novelty, we do not benchmark our work against other genre-related research from MIR.

5.1 Dataset

The dataset used in the experiments was created from the dataset used in the 2018 AcousticBrainz Genre Task, part of the MediaEval benchmarking initiative [7]. The dataset in its original form was aimed at testing the automatic genre annotation from content-based features of musical items in a more challenging setup compared to past works. For each item, annotations from different sources were available, each source taxonomy was much more detailed with hundreds of genres-subgenres, and the overall task was modeled as a multi-label classification. The sources were already introduced in Section 3.1. We further describe how the provided dataset was created. In Discogs, the release annotation was propagated to tracks. In Lastfm and Tagtraum, each track was annotated with music genres and subgenres from the derived taxonomies [7]. We present an overview of the dataset in Table 1.

Dataset	Discogs	Lastfm	Tagtraum
Annotation type	Expert	User	User
Items	1,098,337	686,979	589,584
Number genres	315	327	296

Table 1: Description of the dataset [7].

Although the data was already split between development, validation and test [7], we brought several modifications to accommodate the translation task. We created a large dataset comprising the original development and validation data. In order to assess a notion of confidence on the computed metrics, we resorted to K-Fold cross validation [14]. For each possible target, we splitted the data in 4 folds using stratified sampling. First, we filtered out the items which were not annotated in the target tag system. Then, we used an altered version of the iterative stratification algorithm in [32] in order to ensure that the proportion of items for each target label was roughly the same across folds. Following [13], we added the constraint that items belonging to the same artist had to be assigned to the same fold. For that, we used MusicBrainz artist ids retrieved from the recording ids provided in the MediaEval data.

5.2 Evaluation setup

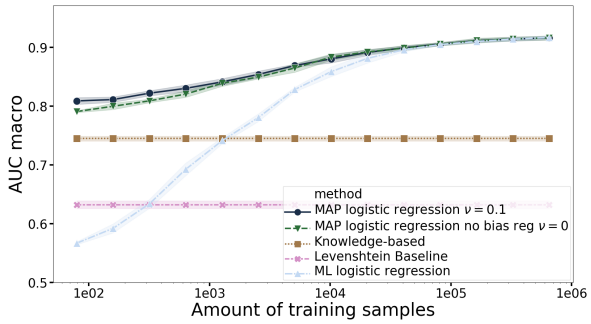
The presented models output a score for each target tag that relates to the confidence of this tag being used in the target annotation. We evaluated these outputs with a ranking metric called Area Under the receiver operating characteristic Curve (AUC), as commonly done in multilabel classification [22]. The (*macro*) averaging is over target tags and measures the ability of the system to rank higher a positive tag than a negative one. Specifically, shifting the values in a column by the same factor (or changing the values of $\mathbf{b}$ in the logistic regression) does not change the AUC macro score, being in that sense, unaffected by item popularity.

We evaluated the logistic regression models on each fold and trained on the three others. We uniformly subsampled the training data to simulate low data availability and chose subsampling factors as powers of 2 between 2^{-13} and 1. Consequently, for the smallest subsampling factors, some source and target tags may not be present in the training data. We used scikit-learn [25] implementation for ML logistic regression, with L-BFGS as the solver. We wrote a Tensorflow [1] implementation of MAP logistic regression. The Adam optimizer [17] was used, with a learning rate of 0.5. We trained the model for 500 epochs with batches of size 100000 or with the full training set if there were less samples. λ was chosen using Eqn 7.

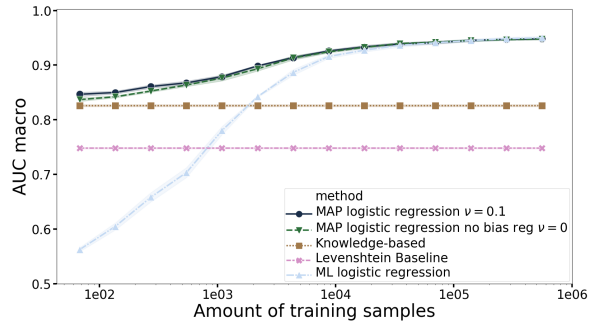
5.3 Results

Figure 2 illustrates how the ML translation eventually outperforms the KB model when enough data is available, while the latter performs much better when little data is available. The MAP translation successfully builds on the KB translation to yield the best results across the whole data availability spectrum. A simple baseline based on tag Levenshtein distance is also shown. Using only a source instead of two (e.g. only Lastfm) led to the same kind of behavior. While we currently proposed one method to obtain the KB translation table, we could also imagine it replaced by an expert-created one, if desired.

The fact that the MAP logistic regression performs consistently well on all the translation tasks is favorable evidence towards the choice of λ given in Eqn 7 as a good default, which we also confirmed using a grid search. Furthermore, we see that the MAP logistic regression lever-



(a) Translation from Lastfm and Tagtraum to Discogs



(b) Translation from Discogs and Tagtraum to Lastfm

Figure 2: AUC scores for the three models per amount of training samples (log-scale). The width of the area around the lines marks the standard deviation computed on folds. KB yields a constant value as it is data-independent.

ages even low amounts of training data to improve over the KB model, and even more so when applying regularization on the bias. We further explain this effect by analyzing how the AUC scores compare on a per-tag basis.

Figure 2 shows that MAP logistic regression with bias regularization can achieve better AUC scores than the KB translation, even on target tags absent during training. We argue that this is due to the regularization term on the bias that enables to learn negative correlations between tags. When no bias regularization is used, the optimal set of parameters for a tag t missing from the training data is: $(\mathbf{w}_t^\star, b_t^\star) = (\mathbf{w}_t^{\text{KB}}, -\infty)$. Indeed, we see in Figure 3 that the MAP results are very close to those of the KB model. This is not true anymore with bias regularization. The gradient of the cost function w.r.t. $\mathbf{w}_t$ can be written as:

$$\frac{\partial \mathcal{L}_{\text{MAP}}}{\partial \mathbf{w}_t} = [\hat{s}\sigma(w_{ts} + b_t) + \lambda^2(w_{ts} - w_{ts}^{\text{KB}})]_{1 \leq s \leq |\mathcal{S}|} \quad (8)$$

where $\hat{s}$ is the number of times (possibly 0) the source tag s appears in the training set. We therefore see that as w_{ts} gets closer to w_{ts}^{KB}, the term $\hat{s}\sigma(w_{ts} + b_t)$ will start to outweigh the second. Applying a gradient step will tend to decrease w_{ts}, away from w_{ts}^{KB}, and even more so when $\hat{s}$ is large (popular tags) and b_t is close to 0 (controlled by the regularization term), hence the negative correlation.

Finally, it is worthwhile to mention that the AUC metric relies on occurrences and is thus arguably biased towards statistical methods. We end this section by taking a qualitative look at how statistics modified the similarities between source and target tags, in particular for those with

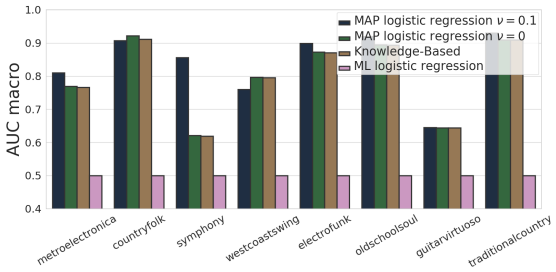


Figure 3: AUC scores for tags that were not in the training set, subsampled by a factor of 2^{-12} with Lastfm as target.

very different KB and ML AUC scores. These differences fall under four explanations:

- Annotation noise: Statistical models learn a very high similarity between the Discogs tag *italo-disco* and the Lastfm tag *classicalbritishheavymetal*. Both indeed often co-occur in the data, but are ontologically unrelated.
- The target tag does not have a suitable equivalent in the source taxonomies. Some Latin and Caribbean music genres like *cumbia*, *fado*, *rocksteady* or *forró* are present in Discogs but are not in Lastfm or Tagtraum. Thus, the mapping to DBpedia, described in Section 3.3, fails.
- The considered tag is highly ambiguous. Take the example of the tag *classical*. Besides the identical counterparts, knowledge-based translation tables also indicates relatedness to some subgenres of *jazz*. However, the specific translation task on which we evaluate appears to be more biased towards an understanding of *classical* that relates to subgenres of *metal* and *electronic* music (*symphonicmetal*, *germanmetal*, *postmodernelectronicpop*).
- The existing genre representations are incomplete or noisy. For instance, *baroque* has a counterpart in each taxonomy, but no direct link with *classical* in DBpedia. Statistical models find high correlation in the data between those two tags and so achieve better AUC scores.

6. CONCLUSION

In this work, we investigated the translation of tags from various source tag systems to a common target tag system. We show that the availability of large amounts of data advantages statistical methods over the knowledge-based one in terms of multilabel classification metrics. Moreover, the proposed hybrid method consistently outperforms both other methods on the whole range of data availability.

Although we did not address multi-language tag systems, both the knowledge-based approach that uses a mapping through the multilingual DBpedia, and the data-informed approach that only takes advantages of parallel annotations and is then insensitive to language, should be able to handle it. As future work, we aim to gather multilingual music genre datasets in order to confirm this claim. We also aim to exploit more thoroughly the genre graph we created by adding more knowledge sources and generating genre representations as node embeddings. We also consider modelling the tag annotation noise, such as missing or spurious tags, or tag bombing, in order to filter it out.

7. REFERENCES

[1] Martín Abadi and et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Manel Achichi, Pasquale Lisena, Konstantin Todorov, Raphaël Troncy, and Jean Delahousse. Doremus: A graph of linked musical works. In *International Semantic Web Conference*, pages 3–19, 2018.

[3] Derek Anderson. Word ninja, 2019. Software available at https://github.com/keredson/wordninja.

[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[5] Jrgen Bang-Jensen and Gregory Z. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer Publishing Company, Incorporated, 2nd edition, 2008.

[6] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006. pages 30–31.

[7] Dmitry Bogdanov, Alastair Porter, Julián Urbano, and Hendrik Schreiber. The mediaeval 2017 acousticbrainz genre task: content-based music genre recognition from multiple sources. In *MediaEval 2017 AcousticBrainz*, 2017.

[8] David Brackett. *Categorizing sound: genre and twentieth-century popular music*. Univ of California Press, 2016.

[9] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. In *International Conference on Learning Representations*, 2018.

[10] Emanuele Coviello, Riccardo Miotto, and Gert R.G. Lanckriet. Combining content-based auto-taggers with decision-fusion. In *Conference of the International Society on Music Information Retrieval*, pages 705–710, 2011.

[11] Alastair JD Craft, Geraint A Wiggins, and Tim Crawford. How many beans make five? the consensus problem in music-genre classification and a new evaluation method for single-genre categorisation systems. In *Conference of the International Society on Music Information Retrieval*, pages 73–76, 2007.

[12] Rene De La Briandais. File searching using variable length keys. In *Western Joint Computer Conference*, IRE-AIEE-ACM '59 (Western), pages 295–298, New York, NY, USA, 1959. ACM.

[13] Arthur Flexer. A closer look on artist filters for musical genre classification. In *Conference of the International Society on Music Information Retrieval*, pages 341–344, 2007.

[14] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Springer series in statistics New York, 2001. pages 241–249.

[15] Andrew Gelman, Aleks Jakulin, Maria Grazia Pittau, and Yu-Sung Su. A weakly informative default prior distribution for logistic and other regression models. *The Annals of Applied Statistics*, 2(4):1360–1383, 2008.

[16] Romain Hennequin, Jimena Royo-letelier, and Manuel Moussallam. Audio based disambiguation of music genre tags. In *Conference of the International Society of Music Information Retrieval*, pages 645–652, 2018.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] Jin Ha Lee and J Stephen Downie. K-pop genres: A cross-cultural exploration. In *Conference of the International Society on Music Information Retrieval*, pages 529–534, 2013.

[19] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6:167–195, 2015.

[20] Pasquale Lisena, Konstantin Todorov, Cécile Cecconi, Françoise Leresche, Isabelle Canno, Frédéric Puyrenier, Martine Voisin, Thierry Le Meur, and Raphaël Troncy. Controlled vocabularies for music metadata. In *Conference of the International Society on Music Information Retrieval*, pages 424–430, 2018.

[21] Michael Mandel, Douglas Eck, and Yoshua Bengio. Learning tags that vary within a song. In *International Society for Music Information Retrieval Conference, ISMIR 2010*, pages 399–404, 08 2010.

[22] Sergio Oramas, Oriol Nieto, Francesco Barbieri, and Xavier Serra. Multi-label music genre classification from audio, text and images using deep features. In *Conference of the International Society on Music Information Retrieval*, pages 23–30, 2017.

[23] Lorena Otero-Cerdeira, Francisco J. Rodríguez-Martínez, and Alma Gómez-Rodríguez. Ontology matching: A literature review. *Expert Systems with Applications*, 42(2):949–971, 2015.

[24] Panagiotis Papadimitriou, Panayiotis Tsaparas, Ariel Fuxman, and Lise Getoor. Taci: Taxonomy-aware catalog integration. *IEEE Transactions on Knowledge and Data Engineering*, 25(7):1643–1655, July 2013.

[25] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning

in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] Simone Paolo Ponzetto and Roberto Navigli. Large-scale taxonomy mapping for restructuring and integrating wikipedia. In *International Joint Conference on Artifical Intelligence*, IJCAI'09, pages 2083–2088, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[27] Natalia Prytkova, Gerhard Weikum, and Marc Spaniol. Aligning multi-cultural knowledge taxonomies by combinatorial optimization. In *International Conference on World Wide Web*, WWW '15 Companion, pages 93–94, New York, NY, USA, 2015. ACM.

[28] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333, 2009.

[29] Steven S. Aanen, Damir Vandic, and Flavius Frasincar. Automated product taxonomy mapping in an e-commerce environment. *Expert Systems with Applications*, 42(3):1298–1313, 2015.

[30] Hendrik Schreiber. Improving genre annotations for the million song dataset. In *Conference of the International Society of Music Information Retrieval*, pages 241–247, 2015.

[31] Hendrik Schreiber. Genre ontology learning: Comparing curated with crowd-sourced ontologies. In *Conference of the International Society on Music Information Retrieval*, pages 400–406, 2016.

[32] Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. On the stratification of multi-label data. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 145–158, Berlin, Heidelberg, 2011. Springer-Verlag.

[33] Mohamed Sordo, Oscar Celma, Matin Blech, and Enric Guaus. The Quest for Musical Genres: Do the Experts and the Wisdom of Crowds Agree? In *Conference of the International Society on Music Information Retrieval*, pages 255–260, 2008.

[34] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI Conference on Artificial Intelligence*, pages 4444–4451, 2017.

[35] Dennis Spohr, Laura Hollink, and Philipp Cimiano. A machine learning approach to multilingual and cross-lingual ontology matching. In *International Semantic Web Conference*, pages 665–680, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[36] Bob L. Sturm. Classification accuracy is not enough. *Journal of Intelligent Information Systems*, 41(3):371–406, 2013.

[37] Aaron Swartz. Musicbrainz: A semantic web service. *IEEE Intelligent Systems*, 17(1):76–77, 2002.

[38] Tobias Swoboda, Matthias Hemmje, Mihai Dascalu, and Stefan Trausan-Matu. Combining taxonomies using word2vec. In *ACM Symposium on Document Engineering*, DocEng'16, pages 131–134, New York, NY, USA, 2016. ACM.

[39] Brian Tomasik, Joon Hee Kim, Margaret Ladlow, Malcolm Augat, Derek Tingle, Rich Wicentowski, and Douglas Turnbull. Using regression to combine data sources for semantic music discovery. In *Conference of the International Society on Music Information Retrieval*, pages 405–410, 2009.

[40] Strother H. Walker and David B. Duncan. Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54:167–178, 1967.

[41] Jun Wang, Xiaoou Chen, Yajie Hu, and Tao Feng. Predicting High-level Music Semantics using Social Tags via Ontology-based Reasoning. In *Conference of the International Society on Music Information Retrieval*, pages 405–410, 2010.

[42] Eric W Weisstein. Statistical rank, 2019. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/StatisticalRank.html.

[43] Tianxing Wu, Guilin Qi, Haofen Wang, Kang Xu, and Xuan Cui. Cross-lingual taxonomy alignment with bilingual biterm topic model. In *AAAI Conference on Artificial Intelligence*, pages 287–293, 2016.

[44] G. K. Zipf. *Human behavior and the principle of least effort*. Cambridge, MA, Addison-Wesle, 1949.

GENERATING STRUCTURED DRUM PATTERN USING VARIATIONAL AUTOENCODER AND SELF-SIMILARITY MATRIX

I-Chieh Wei[1] **Chih-Wei Wu**[2] **Li Su**[1]
[1]Institute of Information Science, Academia Sinica, Taiwan
[2]Netflix, Inc., USA
sma1033@iis.sinica.edu.tw, chihweiw@netflix.com, lisu@iis.sinica.edu.tw

ABSTRACT

Drum pattern generation is a task that focuses on the rhythmic aspect of music and aims at generating percussive sequences. With the advancement of machine learning techniques, several models have been proven useful in producing compelling results. However, one of the main challenges is to generate structurally cohesive sequences. In this study, a drum pattern generation model based on Variational Autoencoders (VAEs) is presented; Specifically, the proposed model is built to generate symbolic drum patterns given an accompaniment that consists of melodic sequences. A self-similarity matrix (SSM) is incorporated in the process for encapsulating structural information. Both the objective evaluation and the subjective listening test highlight the model's capability of creating musically meaningful transitions on structural boundaries.

1. INTRODUCTION

Music generation has become an increasingly popular research field as machine learning techniques continue to thrive [3]. Generating symbolic music sequences using variants of deep neural networks (DNNs) has shown promising results with various degrees of success [6, 9, 10, 19, 22]. In the meantime, drum pattern generation, a subtask that mainly concerns the creation of drum sequences, receives relatively less attention. While some models designed for melodic sequences could be applied to drums directly [6, 19], techniques developed specifically for drum patterns are still in need of further exploration.

In Western music genres such as rock, pop, and jazz, drums usually provide the rhythmic support to melodic instruments and reflect the structure of a song. For instance, drum patterns within the same section (e.g., verse) are typically derived from the same rhythmic motif, and new patterns such as drum fills would appear around the structural boundaries (e.g., from verse to chorus). In other words, drum patterns not only enhance the rhythmic progression, but also facilitate the structural segmentation. Additionally, in order to achieve the rhythmic coherence, drum patterns tend to correlate with other instruments (e.g., rhythmic guitar and bass guitar). These particularities suggest that the structural and rhythmic information from other instruments is crucial for designing reasonable drum patterns.

To build a model that accounts for the above considerations, we explore the idea of using a self-similarity matrix (SSM) as an intermediate structural representation for drum pattern generation. Particularly, we utilize a Variational Autoencoder Generative Adversarial Network (VAE-GAN) to predict the corresponding drum SSM given the SSM of polyphonic mixture of melodic instruments. Subsequently, another VAE-based model generates MIDI drum tracks based on the predicted drum SSM. The contributions of this work include:

(i) a new way of incorporating structural information in the context of drum pattern generation,

(ii) a novel bar selection mechanism that encourages self-repetition in similar sections, and

(iii) the insights into the model's capability of handling transitions between structural sections.

2. RELATED WORK

Drum pattern generation involves the creation of rhythmic patterns with all types of drums; it is an important subtask in conditional music generation problems such as multitrack and lead sheet generation [6, 14]. According to the input and output representation, drum pattern generation models can be roughly divided into (i) symbolic-domain and (ii) audio-domain models.

Systems operating in symbolic-domain work exclusively on discretized representations such as MIDI, a format that allows systems to concentrate on essential information at the semantic level. The majority of prior studies falls into this category. Some of the early systems adopt Genetic Algorithms (GA) to create variants of rhythmic patterns [2, 8, 11, 15] and explore new patterns through a simulated evolution process. However, designing an effective fitness function is non-trivial and relies heavily on the

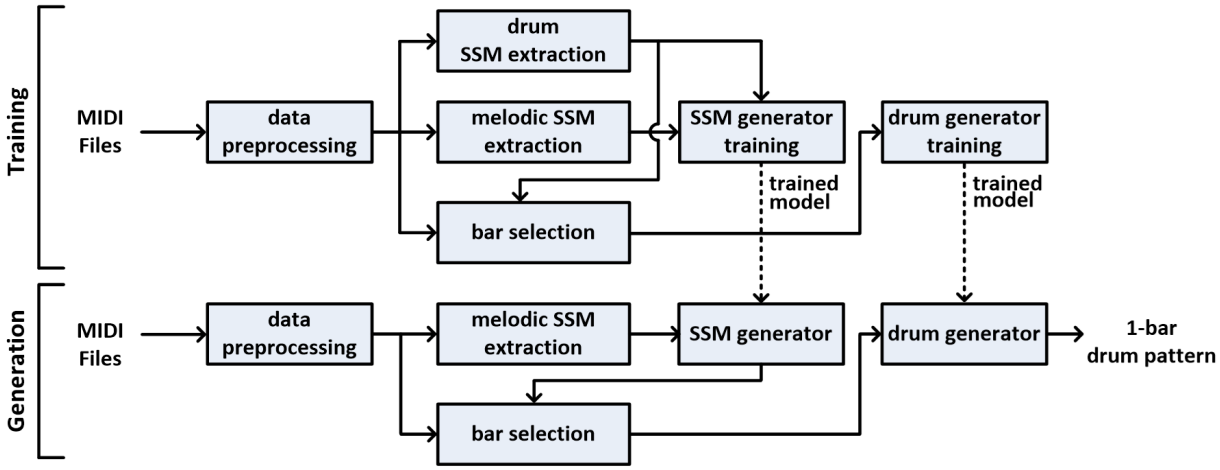


Figure 1. Block diagram of the proposed drum pattern generation system.

domain knowledge. In addition to GA, probabilistic models [16] and deep learning-based models [1,7,14] have also been proposed; these methods generally avoid predefined rules and learn from the data directly, but their performance and generality would vary depending on the data source.

Systems operating in the audio domain directly deal with continuous signals such as waveform. This type of systems usually has a higher degree of freedom in terms of the output. Training such systems requires a set of audio signals with manual annotations. To automate the annotation process, Automatic Drum Transcription (ADT) [21], another on-going research topic, would be a necessary intermediate step. As a result, prior studies in this category are relative scarce. Donahue *et al.* proposed to use generative adversarial network (GAN) for creating audio such as electronic drum beats [5], yet still, the current resulting pieces are short clips without long-term structure.

In this paper, we build our system in both audio and symbolic domains. By utilizing the audio data synthesized from the symbolic representation, we take advantage of a large collection of symbolic data and reserve the possibility of future extension to audio domain. To narrow down the scope of this task, we focus on generating drum patterns in Western music genres with a standard drum kit (e.g., Hihat, Snare Drum, Kick Drum, Toms, etc.). Inspired by previous studies on structural analysis [17] and self-similarity constraint for music generation [13], we use SSM to encapsulate the structural information and investigate its potential impacts on the generated sequences. More details are elaborated in the following sections.

3. METHOD

Figure 1 illustrates the flowchart of the proposed drum pattern generation system. We focus on generating drum patterns given a song as a conditional input. The goal of this conditional generation is to generate drum patterns that are both rhythmically and structurally compatible to the given song. To facilitate the preservation of global structure, we propose a system consisting of two generative models, namely, the SSM generator and the drum pattern generator. In the training phase, the MIDI file is separated into the drum track and the melodic track, followed by the calculation of their corresponding SSMs. The SSM generator is trained to predict drum SSM based on the given melodic SSM, and the drum generator is trained to predict drum patterns based on drum SSM and a bar selection mechanism. In the generation phase, two generators are used sequentially to predict the drum SSM and drum patterns given the melodic tracks.

3.1 Data preprocessing

The input of the proposed model is the audio rendered from all the melodic tracks in the MIDI dataset using a software synthesizer; [1] these audio signals are mono-channel sampled at 44.1 kHz. The advantage of using synthesized audio rather than symbolic data is to enable future adaptation to real-world audio. The tempo of each song is normalized to 120 BPM. Every spectrogram calculated from the audio is denoted as a tensor $\mathbf{Y}$ in shape of (b, f, t), where b is the number of bars, f is the number of frequency bins, and t are the number of time frames in a one-bar spectrogram, respectively. In this study, we set $b = 8$, $f = 84$, and $t = 96$.

For each synthesized audio clip, we first compute the constant-Q transform (CQT) spectrogram using `LibROSA` library. Subsequently, we divide the spectrogram into bars according to the beat and downbeat attributes in the MIDI data, and normalize the time step to 96 frames per bar through linear interpolation. The resulting size of each single-bar spectrogram is therefore 84×96.

The drum patterns are represented in a binary-valued matrix $\mathbf{B} \in \{0, 1\}^{i \times t}$, where i and t denote the activated instruments and the number of time steps respectively.

There are two types of SSM used in this work. The first one is melodic SSM and the second one is drum SSM. Both SSMs are 256×256 matrices computed using pairwise Euclidean distance between two sets of bar-level

[1] http://www.fluidsynth.org/, last access 2019/06/28

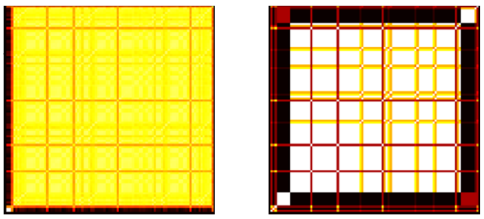


Figure 2. SSM examples of *Can't Buy Me Love* by Beatles. Left: melodic SSM. Right: drum SSM from original MIDI drum track. In the two figures, each pixel represents a Euclidean distance value between two bar-level spectrograms/symbolic drums and brighter color indicates a shorter distance (i.e., higher similarity).

spectorgrams. Since different songs vary in length, we zero-pad all the songs up to a uniform length of 256 bars. It should be noted that melodic SSM is computed using CQT spectrogram converted from synthesized audio domain data, whereas drum SSM is computed using symbolic drum track data directly.

3.2 Drum SSM generator

The motivation of generating a drum SSM from a melodic SSM is shown in Figure 2. It can be observed from Figure 2 that melodic and drum SSM are structurally correlated, and drum SSM seems to provide a relatively clearer view of structural boundaries. This difference could originate from the distinctive roles of percussive versus melodic instruments in Western music. For example, in pop or rock music, drum patterns tend to be homogeneous within a musical section. Sudden changes of drum patterns usually occur before transitioning into a new musical section. As a result, drum SSM could effectively reflect the global structure. Based on these observations, we assume that (i) it is possible to infer drum SSM given melodic SSM because they are highly correlated, and (ii) drum SSM provides more information about song structure information than the melodic SSM does.

To explicitly capture structural information of the entire song prior to drum pattern generation, a model that infers the drum SSM from a melodic SSM is needed. In this work, we propose a drum SSM generator based on the VAE-GAN model in [12]. The VAE-GAN model is a GAN consisting of a VAE-based Generator and a Discriminator. In the training stage, the VAE-based SSM generator is trained to infer the drum SSM based on the input melodic SSM, and the discriminator is trained to distinguish the VAE-generated drum SSM from the original one. The model is optimized by minimizing the total loss function $\mathcal{L}_{ssm}$ consisting of three loss terms, namely the reconstruction loss, KL-divergence loss, and the adversarial loss $\mathcal{L}_D$:

$$\begin{aligned}\mathcal{L}_{ssm} = &-\mathrm{E}_{v\sim q_s(z_s|s_m)}[\log p_s(s_d|z_s)] \\ &+ \mathrm{KL}(q_s(z_s|s_m)\|p(z_s)) + \mathcal{L}_D, \end{aligned} \quad (1)$$

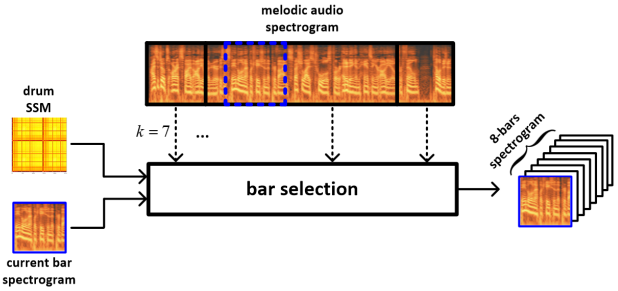


Figure 3. Bar selection mechanism to pick spectrogram data from the current and the other 7 relevant bars.

where s_d represents the generated drum SSM, s_m represents the input melodic SSM and z_s is the latent space representation of s_m.

In the generation stage, we feed the melodic SSM into the pre-trained generator and obtain the predicted drum SSM, which will be used in the following bar selection process.

3.3 Bar selection

To incorporate the structural information into drum pattern generation, the drum SSM is used in a bar selection mechanism. The proposed bar selection mechanism is based on an assumption that musical bars with higher similarities are more likely to provide relevant information for generating compatible drum patterns. To achieve this goal, we first find the k-nearest bars for every bar-level spectrogram according to the drum SSM. These bars are identified by finding the k smallest values in every column of the drum SSM. The process is illustrated in Figure 3. In our study, we set $k = 7$. The corresponding spectrograms of these eight bars are weighted and then stacked into an eight-channel feature representation, which will be used as the input to the subsequent drum pattern generator (see Section 3.4). The weighting coefficient of each channel is $1 - \mathrm{norm}(d(k, k_i))$, where k_i is the ith nearest bar, d is the Euclidean distance, and norm represents normalization over the column of the similarity matrix.

3.4 Drum pattern generator

Figure 4 illustrates the drum pattern generation model. The model is modified from the VAE proposed by Larsen et al. [12] The virtue of VAE is its capability of generating diverse output through simple manipulation in the latent space. For instance, drum patterns in-between two distinctive rhythmic styles can be generated by morphing the latent vector c and z.

To train this VAE model, we feed the encoder E with spectrogram and use symbolic drum track data as ground truth to minimizing the following loss function $\mathcal{L}_{drum}$:

$$\begin{aligned}\mathcal{L}_{drum} = &-\mathrm{E}_{z\sim q(z|y)}[\log p(x|z)] \\ &+ \mathrm{KL}(q(z|y)\|p(z)) + r(c, \hat{c}), \end{aligned} \quad (2)$$

where $r(c; \hat{c}) := |c - \hat{c}|$. x represents the generated drum

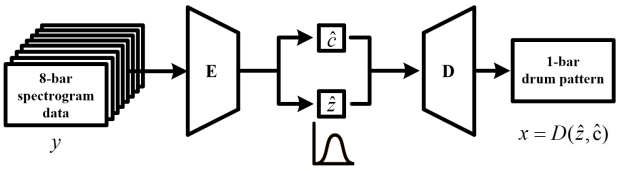


Figure 4. Illustration of the VAE-based drum pattern generator. The 8-bar spectrogram y is obtained from the bar selection mechanism. The encoder maps the spectrogram into two latent spaces, a Gaussian vector $\hat{z}$ and a note density value $\hat{c}$. The values sampled from these two latent spaces are fed to the decoder to generate drum patterns.

patterns, y is the input spectrogram, and c and $\hat{c}$ are the ground-truth and the estimated note density, respectively.

4. IMPLEMENTATION

4.1 Dataset

In this work, we use the Lakh pianoroll dataset (LPD-5) [6], a collection from cleaned MIDI data in the Lakh MIDI Dataset (LMD) [18]. LPD-5 contains 21,425 songs and each song has five tracks (piano, guitar, string, bass, and drums) extracted from the original MIDI data. The dimensionality of each bar in melodic tracks is 128 (pitch) $\times$ 96 (time step). For drum tracks, we process the data with following steps: (i) remove MIDI pitches (representing different percussive instruments) that are relatively inactive (e.g., less than 0.1% of all active drum notes); (ii) reduce time steps from 96 to 16 (see Section 4.2 for more details); (iii) apply binarization on each activated drum note, and (iv) calculate the note count in each single bar as a proxy for note density (rhythmic complexity). This procedure results in a down-sampled drum matrix with a dimensionality of 46$\times$16.

4.2 Data cleaning

Although the LPD-5 dataset has included a series of operations to clean up the original LPD [6], incomplete or noisy examples can still be found. To further improve the data integrity, we proceed with the following steps:

First, we remove songs with inconsistent duration after synthesis. Some tracks contain only a few notes throughout the entire file, and the empty bars are automatically removed during synthesis. By excluding these songs, we ensure the correctness of information regarding song progression (e.g., beat and downbeat locations).

Second, we remove songs with empty or noisy drum tracks. Specifically, we estimate the distribution of note count from drum tracks and exclude the ones that are outside of the two standard deviation range. This outlier removal process reduces the noise and avoids issues that might be induced by data sparsity.

Finally, we apply 16th beat quantization on drum tracks and remove the notes that are largely shifted during the quantization process. In LPD-5 dataset, 95% of activated drum notes are ether on the 16th beat grid or within a tolerance range of 96th beat. Our experiment shows that less than 5% of drum notes are affected by this data cleaning process. Therefore, we believe that 16th beat grid is applicable to provide a compact data representation while retaining a meaningful temporal resolution.

generation method	cosine similarity
OMD	1.0000
ODS	0.9208
PDS	0.9164
NB	0.9056

Table 1. Similarity measure of drum SSMs from different generation method, higher similarity value indicates less deviation from original song structure after generation.

4.3 Experimental setup

After the cleaning process, 9,907 songs remain in the dataset. We randomly split the dataset into 90% and 10% for training and testing, respectively. For each song, drum and melodic parts are extracted accordingly. The melodic tracks are rendered into wave format and transformed into per-bar CQT spectrogram. For drum tracks, we segment the data in bar-level and apply binarization. The training process is done by minimizing the loss function as defined in Section 3.3 and Section 3.4; the selected optimizer is ADAM with a batch size of 64. The models are implemented using Tensorflow.

A similar preprocessing procedure is applied to the data in testing phase. The major difference between the training and the testing is the source of drum SSM. During testing, only the melodic SSM is available, and the drum SSM is predicted using the pre-trained drum SSM generator.

4.4 Model parameters

Both the input and output dimensions of the drum SSM generator (as described in Section 3.2) are 256$\times$256. The encoder is composed of eight convolutional (CONV) layers followed by three fully-connected (FC) layers with skip connections. The FC layer is connected to a bottleneck layer that generates a 32 dimensional latent vector. Similarly, the decoder has the same layers in reverse order. The discriminator is a network similar to the encoder with a sigmoid output; this activation function is chosen for its potential probabilistic interpretation.

For drum pattern generation, the architecture of our VAE (i.e., no discriminator) is similar to the drum SSM generator. The input and output dimensions are changed to 84$\times$96$\times$8 and 46$\times$16$\times$1, respectively. The total numbers of parameters for the SSM generator and the drum pattern generator are around 17M and 62M, respectively. The total training time for the two models is 84 hours on a single 2080 Ti GPU. More implementation details can be found in our Github repository. [2]

[2] https://github.com/Sma1033/drum_generation_with_ssm

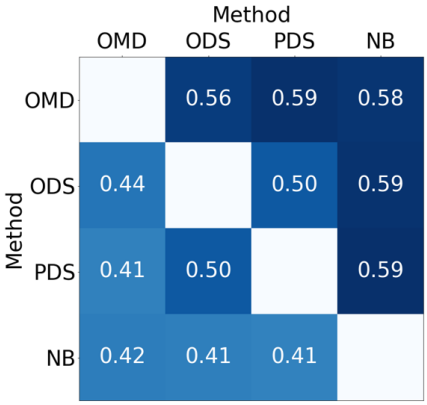


Figure 5. Head-to-head win rate between different models in the listening test.

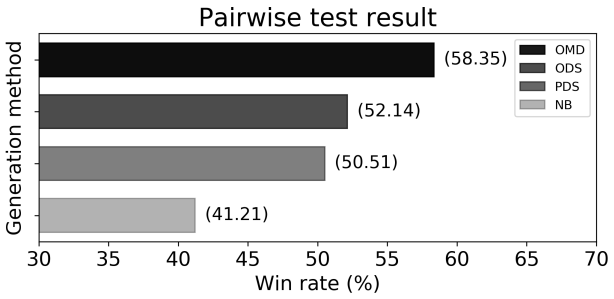


Figure 6. Global win rate of different models in pairwise listening test.

5. EVALUATION

5.1 Compared methods

To evaluate the quality of generated drum patterns of the proposed model, we conduct both objective and subjective tests on four different derived methods:

- **(OMD) Original MIDI Drums** are the drum patterns predefined in the MIDI files. These drum patterns are directly taken from the drum tracks and serve as the oracle samples among all methods.
- **(ODS) Original Drum SSM** is the model that generates the drum patterns with our pre-trained drum pattern generator. In this case, the drum SSM used for bar selection is computed from the oracle drum tracks (i.e., OMD).
- **(PDS) Predicted Drum SSM** is the model that generates the drum patterns with our pre-trained drum pattern generator. In this case, a drum SSM used for bar selection is predicted from melodic SSM using a pre-trained SSM generator (see Section 3)
- **(NB) Neighboring Bars** is the baseline model. Instead of applying bar selection mechanism, this model simply includes the neighboring bars (i.e., previous 4 bars, current bar, and subsequent 3 bars) to create the 8-bar feature representation; no information from SSM is used. This model ignores global structure and incorporates local structure naively.

5.2 Objective test

In our objective test, we compute the similarity between the oracle drum SSM (i.e., OMD) and the SSMs of the drum patterns generated by ODS, PDS, and NB. This evaluation examines the general quality of drum patterns. Ideally, the SSM with high similarity to oracle implies a better preservation of the structural information. For simplicity, we use a standard cosine similarity as our metric. We randomly collect 100 drum tracks generated from the LPD test set, and calculate the cosine similarity between the original and generated SSMs. The average of results are presented in Table 1. Both drum SSM informed methods (i.e., ODS and PDS) outperformed NB, which suggested the competence for drum SSM to preserve structural information. Interestingly, NB can achieve a relatively high similarity score without any additional information of the global structure. This result implies the need for a better and perceptually relevant evaluation.

5.3 Subjective evaluation - pairwise test

5.3.1 Experiment settings

In order to evaluate the perceptual quality of different models, we conducted a listening test to compare the above mentioned four methods. In the test, various samples generated by different methods were presented to participants. There are 10 trials per test; each trial consists of two different samples derived from the same melodic track. After listening, the participants were asked to select the sample with higher rhythmic compatibility. 10 different songs are included in the evaluation; 5 songs are randomly selected from LPD-5 test set, and another 5 songs are randomly collected online in order to test the generality of the methods. In order to examine the model's capability of handling transitions, each sample contains a structural boundary. Particularly, we select samples that include one transition from the verse to the chorus, and the duration is roughly 16 seconds. Listening examples are available online. [3]

5.3.2 Results

The evaluation for 1,350 listening pairs are provided by 135 respondents. 67.5% of the subjects have no music composition experience, and 92% of the subjects have never played drums. The listening test results are presented in Figure 6 and Figure 5. Based on the results, the following observations can be made:

First, OMD performs the best among all models. This result is expected since OMD has the most stable and compelling drum patterns compared to the others. However, the margin between OMD and ODS is relatively small. One possible explanation is that OMD tends to be highly

[3] https://sma1033.github.io/drum_generation_with_ssm/

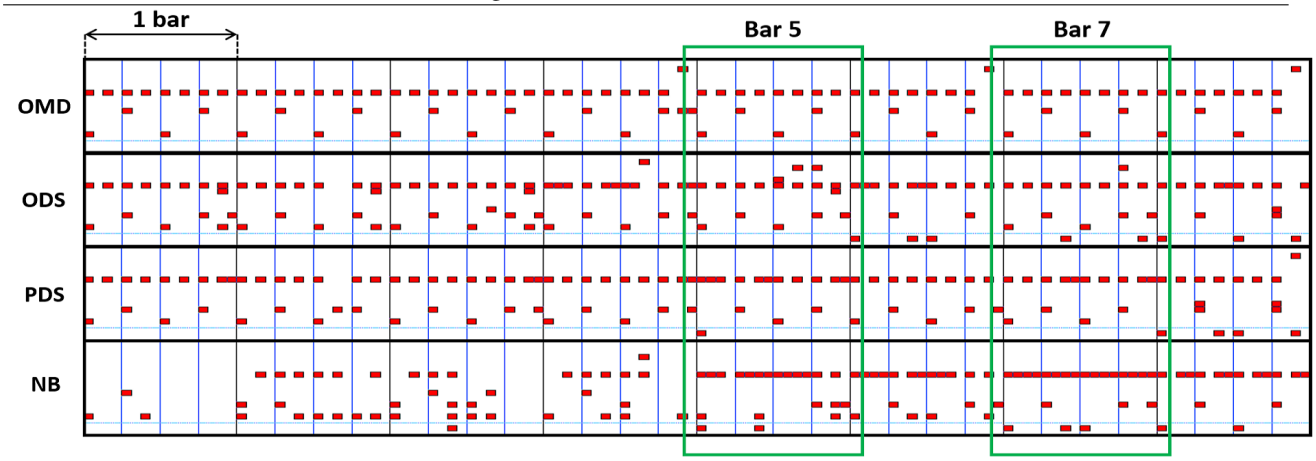


Figure 7. A 8-bar example of drum tracks from four different method in sec 5.1

consistent and predictable, and this can sometimes be regarded as conservative and even boring. On the contrary, the unexpected instrumentation or sequences in other models may accidental attract the audience's attention.

Second, it appears that both ODS and PDS outperform NB. To verify the significance of these comparisons, the t-tests of ODS/NB and PDS/NB pairs are conducted, which produce the p-values of **0.0087** and **0.001**. The results suggest that incorporating SSM in drum pattern generation models is a promising approach.

Third, the performance of PDS is comparable to ODS. The statistics from pairwise listening test suggest that the quality of drum patterns from PDS and ODS are similar for general public. This result not only indicates the effectiveness of the pre-trained SSM generator, but also shows the viability of generating meaningful drum patterns based on melodic SSM.

5.4 Subjective evaluation for professionals

Apart from the aforementioned pairwise test, we also conducted another listening test on professional musicians. The objective of this test is to collect descriptive feedback from subjects with extensive experiences in music composition and drum performance. Two professional drummers and one professional composer (with experience ranging from 3 to 14 years) participated in this test. Each participants was invited to listen to 5 selected songs; each song contains 3 different drum tracks (i.e., ODS, PDS, and NB) presented in random order, resulting in a total number of 15 clips. To further investigate the long-term structure of the generated drum patterns, the duration of each song is extended to 64 seconds. The participants were encouraged to provide detailed comments after each trial. The thematic analysis approach [4, 20] was applied to extract common themes from their comments in a bottom-up manner. The results of thematic analysis are described as follows.

The first theme regards the structural compatibility between the generated drum patterns and the melodic track. From this perspective, ODS seems to receive the best feedback among three competing models. In many occasions, the comments for ODS include "good distinction between sections", whereas PDS and NB are rarely mentioned. Moreover, for transition part between sections, ODS is reported as having active rhythmic changes (e.g., drum-fills). Overall, the professional listeners seem to prefer ODS in terms of its structure.

The second theme regards the stability and variability of the generation result. The comments from professional listeners indicate NB's ability of generating unstructured yet unexpected patterns. ODS and PDS, on the other hand, do not surprise the professional listeners. Figure 7 provides a visual example of all methods. According to Figure 7, ODS and PDS are visually more similar to OMD (e.g., bar 5 and bar 7). However, NB does provide unconventional patterns at several locations, which could be interpreted as "thinking outside of the box".

6. CONCLUSION

We have presented a conditional drum pattern generation model to generate drum patterns based on given melodic tracks. In particular, the model captures the global structure of melodic sequences using SSM and is capable of producing structurally coherent drum sequences. Results from both the objective and subjective evaluation are promising, and the comments from professional listeners also highlight the strength of the model to incorporate drum patterns with the melodic structure. Possible future directions include:

1. Design a user-friendly control interface for general public users. This could potentially encourage more interactions among users and facilitate the music creation process.

2. Generate genre-specific drum patterns according to the input condition. With genre-specific contents, users may customize the styles of outputs.

3. Develop a model that can ultimately perform drum pattern generation in audio domain. This would enable more potential applications in real-world scenarios and increase the expressivity of the model.

7. ACKNOWLEDGEMENT

We thank three professional musicians, Hsin-Ming Lin, Ying-Shu Kuo, and Jian-Hong Chen, for their qualitative assessment on generated drum patterns and our colleague, Yu-Fen Huang, for her generous assistance on the thematic analysis of the questionnaires. We would also like to express our gratitude to every participant of the subjective listening test for providing responses and constructive feedback.

8. REFERENCES

[1] Eric Battenberg and David Wessel. Analyzing Drum Patterns Using Conditional Deep Belief Networks. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 37–42, 2012.

[2] Gilberto Bernardes, Caros Guedes, and Bruce Pennycook. Style emulation of drum patterns by means of evolutionary methods and statistical analysis. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pages 1–4, 2010.

[3] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. Deep learning techniques for music generation - A survey. *CoRR*, abs/1709.01620, 2017.

[4] Kathy Charmaz and Linda Liska Belgrave. Grounded theory. *The Blackwell encyclopedia of sociology*, 2007.

[5] Chris Donahue, Julian Mcauley, and Miller Puckette. Adversarial Audio Synthesis. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–16, 2019.

[6] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

[7] Hamid Eghbal-zadeh, Richard Vogl, Gerhard Widmer, and Peter Knees. A GAN based drum pattern generation UI prototype. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2018.

[8] Damon Horowitz. Generating Rhythms with Genetic Algorithms. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 1994.

[9] Cheng-zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2017.

[10] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, and Douglas Eck. Music transformer: generating music with long-term structure. *ICLR*, 2019.

[11] Maximos A. Kaliakatsos-Papakostas, Andreas Floros, Nikolaos Kanellopoulos, and Michael N Vrahatis. Genetic Evolution of L and FL – systems for the Production of Rhythmic Sequences. In *Proceedings of the Annual Conference Companion on Genetic and Evolutionary Computation (GECCO)*, pages 461–468, 2012.

[12] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International Conference on Machine Learning (ICML)*, 2016.

[13] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints. *Journal of Creative Music Systems.*, 2018.

[14] Hao-Min Liu and Yi-Hsuan Yang. Lead sheet generation and arrangement by conditional generative adversarial network. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 722–727, 2018.

[15] Cárthach Ó Nuanáin, Perfecto Herrera, and Sergi Jordà. Target-Based Rhythmic Pattern Generation and Variation with Genetic Algorithms. In *Proceedings of the Sound and Music Computing Conference (SMC)*, 2015.

[16] Jean-François Paiement, Yves Grandvalet, Samy Bengio, and Douglas Eck. A Generative Model for Rhythms. In *Neural Information Processing Systems (NIPS), Workshop on Brain, Music and Cognition*, 2007.

[17] Jouni Paulus, Meinard Müller, and Anssi Klapuri. Audio-based music structure analysis. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 625–636, 2010.

[18] Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching.* PhD thesis, Columbia University, 2016.

[19] Adam Roberts, Jesse Engel, and Douglas Eck. Hierarchical Variational Autoencoders for Music. In *Neural Information Processing Systems (NIPS)*, volume 256, pages 1–6, 2017.

[20] Anselm Strauss and Juliet M Corbin. *Grounded theory in practice*. Sage, San Jose State University, USA, 1997.

[21] Chih Wei Wu, Christian Dittmar, Carl Southall, Richard Vogl, Gerhard Widmer, Jason Hockman, Meinard Mueller, and Alexander Lerch. A review of automatic drum transcription. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 26(9):1457–1483, 2018.

[22] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2017.

RENDERING MUSIC PERFORMANCE WITH INTERPRETATION VARIATIONS USING CONDITIONAL VARIATIONAL RNN

Akira Maezawa Kazuhiko Yamamoto Takuya Fujishima
Yamaha Corporation

ABSTRACT

Capturing and generating a wide variety of musical expression is important in music performance rendering, but current methods fail to model such a variation. This paper presents a music performance rendering method that explicitly models the variability in interpretations for a given piece of music. Conditional variational recurrent neural network is used to jointly train, conditioned on the music score, an encoder from a music performance to a latent representation of interpretation and a decoder from the latent interpretation back to the music performance. Evaluation demonstrates the method is capable of predicting and generating an expressive performance, and that the decoder learns a latent space of musical interpretation that is consistent with human perception of interpretation.

1. INTRODUCTION

Music performance rendering is the task of generating a human-like performance data from a piano music score. That is, for each note in a given music score, it generates a set of expressive performance parameters such as the onset timing, the duration and the velocity. It is an important task in music production, allowing a composer to generate human-like piano part of a new composition, or a musician to listen to a convincing preview of a digital sheet music. It is not only important to generate a convincing performance, but also to allow humans to interact in the generation. For example, it is useful to be able to adjust expressive parameters, or, like how a musician might ask to another musician, feed a reference performance snippet, in expression of which the system should perform.

Capturing and generating a wide variety of musical expression are important in music performance rendering, but current methods are limited in such capabilities. For example, most methods permit control over concrete musical concepts like the average tempo and the average velocity [2], but cannot manipulate abstract musical concepts that cannot be labeled, such as liveliness within a performance. It is also not possible to feed a short reference performance snippet and ask the system to play with that particular musical musical interpretation.

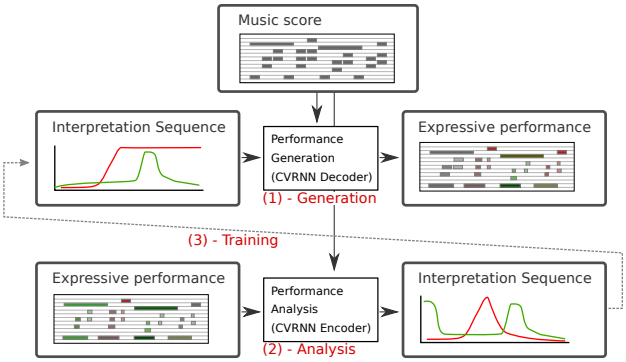


Figure 1. The overview of our method.

These limitations are rooted in the attempt to directly map between the music score and expressive performance [4]. Such a model entwines musical interpretation and its execution, but a human performer presumably decouples *musical intent* and its *execution* given the score. For example, when playing a piece multiple times, a musician might play lively the first time but calmly on the second. Given the intent, however, its execution is consistent based on musical contexts, such as whether a note is a melody or if a particular measure is a beginning of a new phrase. For example, a lively playing might be executed, almost second nature, as shorter chords and louder melody, or calm playing as softer and more broken chords.

To address this problem, we present a music performance rendering method that explicitly decouples intent and its execution based on a music score. We achieve this by jointly training a performance decoder (performance renderer) and a performance encoder (performance analyzer) that are conditioned on (1) a music score, and (2) an **interpretation sequence**, a latent low-dimensional sequence representation that expresses the underlying musical intent. It may be either generated automatically or manipulated coherently. Thus, it is possible to generate different interpretations to a score by either modifying the interpretation sequence or encoding a performance snippet as an initial value for generating the interpretation sequence.

The conceptual overview is shown in Figure 1. Our method, given a music score and an interpretation sequence, generates an expressive performance for the given music (denoted (1)). Conversely, it can encode an expressive performance into the interpretation sequence (denoted (2)). To achieve such a capability, the encoder and the de-

coder are jointly trained by encoding a human performance and trying to recover it with the decoder (denoted by path (3)).

Our contributions are as follows: (1) we propose a new machine learning model, CVRNN, which extends variational recurrent neural network (VRNN [6]) to accept position-dependent conditional inputs; (2) we apply convolutional recurrent network (CRNN) to extract features from the music score; (3) we apply these two models and present a first deep generative model that is simultaneously capable of analyzing and generating an expressive music performance for a given music score with adjustable musical interpretation.

We invite the readers to check out audio demonstrations at `https://sites.google.com/view/cvrnn-performance-render`.

2. RELATED WORK

Music performance rendering is the task of generating an expressive performance from a music score [4]. The research focuses mostly on generating a sequence of expressive onset timings, durations and velocities for a piano piece. Unlike the generation of improvisation [20], performance rendering needs to understand the musical contexts of the given music score, and generate a natural performance constrained by the score.

Previous studies formulate performance rendering as a prediction task of expressive parameters based on features extracted from the music score, such as melodic features [22], perceptual features [3], or neighboring contexts [9, 18]. Then, expressive performance to a new music score can be predicted using methods like a hidden Markov model [12], a decision tree [18], or a dynamic Bayesian network [8]. More recently, deep learning has been incorporated to better extract the features or predict the performance. For example, deep neural networks have been shown to be effective for dynamics prediction [21], tempo prediction [15], and piano performance generation [11, 17].

The capability to adjust the generated performance is important, but current methods cannot capture the abstract variations in playing. For example, the tempo or dynamics [1, 7] can be adjusted, but controlling a more abstract musical idea remains difficult. It is possible to manipulate the performance by mixing the parameters of performance renderers trained on multiple corpora [2], but it fundamentally cannot identify commonalities and differences among a set of corpora.

3. METHOD

Our method generates an expressive piano performance data to a given piano music score, and an interpretation sequence that is either automatically generated, manually adjusted, or initialized by a performance snippet. The score contains a sequence of notes (pitch, position and duration), the average tempo, the time signatures, and the key signatures. The interpretation sequence is a sequence of low-dimensional vectors, each element of which is associated with each note in the score. It is an abstract representation of playing style, each vector of which is called the **interpretation vector**. For each note, based on the interpretation vector and musical context inferred from the music score, our method estimates (1) the fine note onset timing, (2) the duration, (3) the note-on velocity, and (4) the tempo fluctuation of human performance.

To generate the performance, we use a deep generative model that has been trained to generate an expressive performance data, conditioned on the music score and the interpretation sequence. To acquire a temporally coherent manifold over the space of interpretation vector, we propose CVRNN, an extension of VRNN that accepts the music score as a conditioning input. Essentially, we jointly train three models: (1) a music score feature extractor that extracts, for each note, relevant features (**music score feature**) from the music score, (2) an encoder that maps a human performance to an interpretation sequence, and (3) a decoder that maps an interpretation sequence to an expressive performance.

Hereon, n indicates the index of a note encountered in the music score, lexicographically sorted by the onset beat position and the pitch. x_n indicates the expressive performance data, c_n indicates the music score feature that has been extracted from the nth note on the score, and z_n indicates the interpretation vector associated with x_n.

3.1 Representation of expressive performance

The generated expressive performance x_n comprises of the note velocity, the note duration, the micro-onset timing deviation and the tempo.

The **note velocity** is an integer between 1 and 127 indicating the MIDI velocity value, encoded as a 128-dimensional one-hot vector.

The **note duration** is an integer between 1 and 800, indicating the duration as $1/100$ths of a beat relative to the current tempo (*e.g.* duration of "42" means 0.42 beats). It is encoded as an 800-dimensional one-hot vector.

The **micro-onset timing** is an integer between -50 and 49, indicating the number of $1/100$ths of a beat elapsed after an ideal onset time. The ideal onset time is computed based on the generated tempo curve up to the current point in the score. It is represented as a 100-dimensional one-hot vector, ith element of which means that the onset is lagging by $(i - 50)/100$ beats.

The **tempo** is an integer indicating the the difference between the current tempo and the tempo written in the music score, in bpm. It is represented as a 100-dimensional one-hot vector, ith element of which means that the output should be faster than the written tempo by $i - 50$ bpm. It is computed by interpolating the music score position and the playback position of the performance using Gaussian regression with a squared exponential kernel with a FWHM of 2 beats.

3.2 Extraction of the music score feature

The music score contains a sequence of notes, average dynamics, average tempi, meters, and key signatures. From

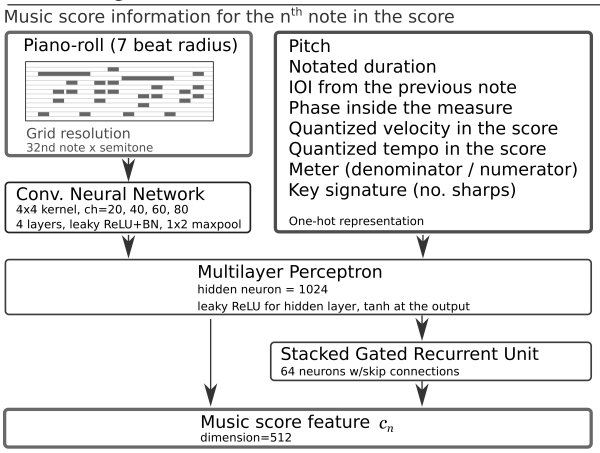


Figure 2. Overview of music score feature extractor for extracting the music score feature.

the music score, we extract the **music score feature** c_n, which represents the nth note and its surrounding context.

To extract the feature, we use the following information: the **key signature** as a 12 dimensional one-hot vector that indicates the number of accidentals; the **time signature** as a tuple of 12 dimensional one-hot vector, indicating the numerator and the denominator; the **pitch** as a 128 dimensional one-hot vector indicating the MIDI note number; the **inter-onset interval** from the previous note as a multiple of 96th note, represented as a 512 dimensional one-hot vector; the **duration** of the note as a multiple of 96th note, represented as a 192 dimensional one-hot vector; the written **tempo** quantized to a multiple of 30 bpm, represented as a 9 dimensional one-hot vector (30-300 bpm); the written **velocity** quantized to a multiple of 10, represented as a 12 dimensional one-hot vector; the **piano-roll** centered about the current position in the score, spanning a radius of 7 beats at a 32nd note resolution (224x128 dimension). The velocity and the tempo are quantized at a low resolution because different music notation software map the dynamics and tempo markings to similar velocity and bpm values, but the exact mapping differ.

Based on these inputs, we extract the feature using a CRNN as shown in Figure 2. First, the piano-roll passes through a CNN. It is comprised of four layers, whose kernel sizes are all (4×4), and channel sizes from the lower to the higher layers are [20, 40, 60, 80]. Each layer is followed by max-pooling with a kernel size of (2×1), batch normalization and leaky ReLU nonlinearity. The output the CNN is concatenated with the remaining one-hot vectors mentioned above. Then, the concatenated vector is passed through a multi-layer perceptron (MLP) with 1024 hidden units and leaky ReLU nonlinearity at the hidden layer. The MLP output passes through a tanh nonlinearity.

Second, the output of the perceptron is embedded to 64 dimensions using a linear layer. The embedded vector is then passed to a RNN consisting of 3 stacks of gated recurrent units (GRU) [5] with 64 neurons each.

Finally, the outputs of the MLP and the GRU stack are concatenated to create the music score feature c_n.

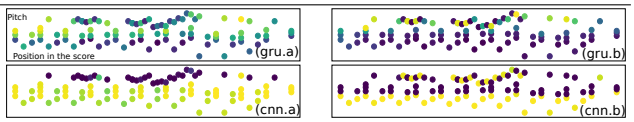


Figure 3. Activation of the most active components of GRU and CNN activations (dark=negative, yellow=positive).

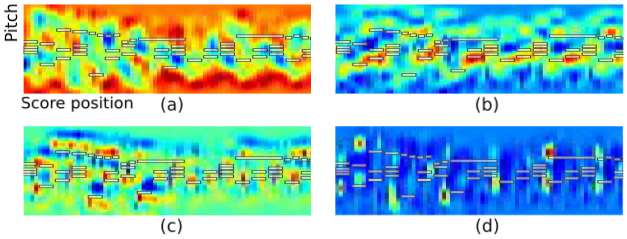


Figure 4. Some feature maps learned from the CNN, overlayed with the piano-roll (blue=negative, red=positive).

3.2.1 Analysis of the music score feature extractor

Here we illustrate some musically relevant concepts that are acquired by the music score feature extractor. We have trained our model using the same dataset used in Section 4, and extracted the music score feature from measures 5–6 of Chopin's Nocturne Op. 9-2 and extracted the GRU and the CNN activations. Some of the highest GRU and CNN activations are shown in Figure 3, showing, for each note, the value of the corresponding activation in different color, positioned at the beat position and the pitch written on the score, *a la* piano-roll. We can see that the GRU acquires concepts like the number of notes stacked below the current point in the score ("gru.a") or top notes ("gru.b"). The CNN extracts longer contextual information such as the melody ("cnn.a"), and the bottom notes ("cnn.b"). Neither the GRU nor the CNN becomes dead: the standard deviation of the activation of the CNN and the GRU are comparable (about 0.5 to 1.0), where about 90 GRU activations have standard deviation of above 0.1, and about 150 for the CNN.

To analyze how these concepts are acquired, we show in Figure 4 the feature maps of the final layers of the CNN, overlayed to the piano-roll. The tendency of the map suggests that the CNN expresses concepts like the register (Fig. 4a), rising arpeggio (b), top and bottom melodic contour (c), or melodic contour (d).

3.3 CVRNN for joint encoding and decoding of the interpretation sequence

We assume that an expressive performance depends on both (1) the musical context, expressed through the music score feature, and (2) a corresponding sequence of low-dimensional music interpretation vectors z_n, dimension of which is set to 5. It is necessary for z_n to learn (1) a manifold that z_n resides in, that explains interpretation reasonably well, and (2) a model of temporal evolution of z_n, so that the generated performance is temporally coher-

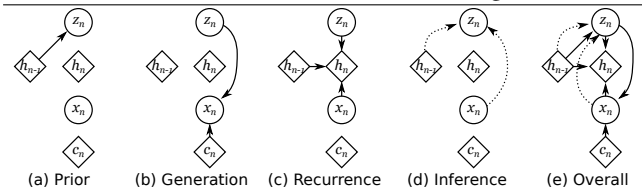


Figure 5. Dependency of the CVRNN model. Solid arrows indicate conditional dependence of the generative process, and dotted arrows indicate conditional dependence of the approximate posterior distribution.

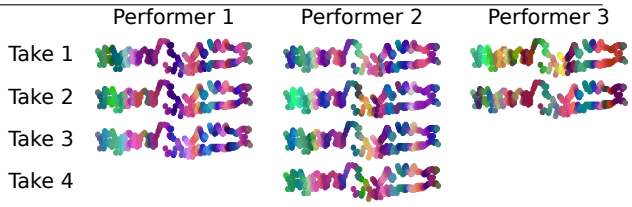


Figure 6. Visualization of the interpretation sequence, for different takes of three professional pianists. Horizontal axis indicates the music score position, vertical axis indicates the pitch, and the color indicates 3D projection of the temporally-smoothed interpretation vector.

ent. We achieve (1) by decoupling the music score and the generative model of z_n, such that z_n learns a repertoire-independent low-dimensional representation of variability of playing. We achieve (2) by basing the temporal evolution of z_n on the state of a RNN h_n.

To meet these requirements, we model the generation process with a CVRNN. There are three key components in a CVRNN, as shown in the dependency diagram in Figure 5: (1) generation of the interpretation sequence z_n given the underlying h_n (called the **prior** distribution), (2) generation of an expressive performance x_n given the interpretation sequence z_n and music score features c_n (called the **generation** distribution), and (3) generation of the interpretation vector z_n given a true human performance x_n (called the **inference** distribution). Finally, the true or the generated expressive performance x_n and the interpretation vector z_n are used to update the underlying h_n, used then to predict the next interpretation vector z_{n+1}.

3.3.1 Generation

We assume the following generative process:

$$p(x_{\leq N}, z_{\leq N}|c_{\leq N}) = \prod_{n=1}^{N} \underbrace{p(x_n|z_n, c_n)}_{\text{generation}} \underbrace{p(z_n|x_{<n}, z_{<n})}_{\text{prior}}. \tag{1}$$

Thus, the data is generated autoregressively by sampling z_n from the prior, generating x_n, and feeding it back to the prior to sample the z_{n+1} for the next note in the score.

For the prior distribution, the previous RNN state h_{n-1} passes through a three-layer densely-connected [10] fully-connected network with 250 units each with a leaky ReLU nonlinearity. By dense connection, we mean that the input to the current layer is a concatenation of the outputs of previous layers. It outputs the mean and log-variance of a Normal distribution $\mu(h_{n-1})$ and $\gamma(h_{n-1})$. Then z_n is sampled as follows:

$$z_n|h_{n-1} \sim \mathcal{N}(\mu(h_{n-1}), \exp(\gamma(h_{n-1}))). \tag{2}$$

From the interpretation vector z_n, we pass it through another feature extractor $\psi(z_n)$ to obtain a feature that describes the current musical interpretation. $\psi(\cdot)$ has the same three-layer architecture as mentioned above.

For the generation of x_n, we use the sampled interpretation vector z_n and the current music context c_n:

$$x_n|z_n, c_n \sim g(c_n, \psi(z_n)), \tag{3}$$

where $g(c, z)$ indicates a Cartesian product of categorical distributions over the one-hot representations of the velocity, the micro-onset timing, the duration and the tempo, parameterized by an output of a three-layer fully-connected network with leaky ReLU nonlinearity for the hidden layers and softmax for the output layer, applied separately for the four output variables.

To recurrently update h_n, we extract a feature vector from x_n by passing it through a network $\phi(\cdot)$ with a same architecture as ψ to obtain $\phi(x_n)$. Then, the underlying state h_n is updated as follows:

$$h_{n+1} = f(h_n, [c_n, \phi(x_n), \psi(z_n)]), \tag{4}$$

where $f(h, x)$ is state update equation of a stacked GRU of three layers, given h as the current state variable and x as the input that has been embedded to 64 dimensions by a fully-connected layer.

Let us elaborate on the modeling assumptions. First, the prior in Eq. 2 is independent of c_n. This forces z_n to express music interpretation abstractly, such that it is decoupled from the musical context, which provides strong hints on how to execute the performance given an interpretation. Second, x_n and h_n depends only indirectly via the bottleneck z_n. This is unlike the original formulation of VRNN where x_n and h_n are directly dependent. We found that such a bottleneck is vitally critical for learning a meaningful representation of z_n; if x_n depends on h_n, then the model simply learns to generate an auto-regressive model of x_n using h_t, almost completely bypassing z_n. Such an error mode occurs because it is much easier for a model to learn to simply predict the next note, instead of having to go through the hassle of trying to explain how it could have varied with a different interpretation.

3.3.2 Inference

In addition to the generative process, we also define an approximate posterior distribution, or the inference distribution, so that variational technique can be used to train the model [14].

We assume the following dependency for the inference distribution:

$$q(z_{\leq N}|x_{\leq N}) = \prod_{n=1}^{N} \underbrace{q(z_n|x_{\leq n}, z_{<n})}_{\text{inference}}. \tag{5}$$

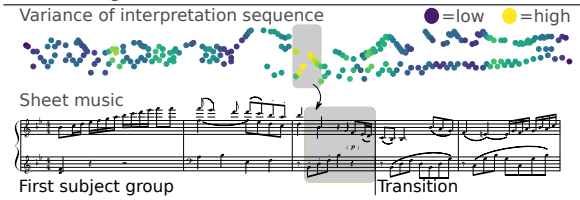


Figure 7. The variance of the interpretation vectors over different performers at each note.

If we assume that z_n is normally distributed and conditionally independent of $z_{<n}$ and $x_{<n}$ given h_{n-1}, it becomes:

$$q(z_n|x_{\leq n}, z_{<n}) = q(z_n|x_n, h_{n-1})$$
$$= \mathcal{N}(z_n|\eta(h_{n-1}, \phi(x_n)), \exp(\nu(h_{n-1}, \phi(x_n)))), \quad (6)$$

where $\eta(\cdot)$ and $\nu(\cdot)$ are the outputs of a neural network with the same architecture as $\psi(\cdot)$, that take $\phi(x_n)$ and h_{n-1} as the inputs. The inference is independent of the music score c_n, so that it learns a *repertoire-agnostic* model for inferring z_n given an expressive human playing.

3.3.3 Training

We train the model by minimizing the KL divergence from the approximate posterior to the true posterior. It amounts to the minimizing of the following note-level loss $l_n(\Theta)$ w.r.t. the set of model parameters Θ, accrued over every note in the training data:

$$l_n(\Theta) = \mathbb{E}_{z_n \sim q(z|x_n, h_{n-1})}[\log p(x_n|z_n, c_n)]$$
$$+ \mathrm{KL}(q(z|x_n, h_{n-1})||p(z|h_{n-1})). \quad (7)$$

The expectation is computed using the reparametrization trick [14]. We use truncated backpropagation with truncation length of 30 notes (longer truncation length of 50 notes yielded qualitatively similar outputs). We also augment the data by adding zero-reverting Wiener noise to the tempo curve and randomly transposing each piece by -7 to 7 semitones at every epoch. To minimize the loss, we use Adam [13], with gradient clipping for gradient values above 5.

3.4 Analysis of the learnt manifold of the interpretation vector

We briefly illustrate the essences of musical expression acquired by the interpretation sequence. Figure 6 shows the interpretation sequence of first 20 bars of Mozart's K333 piano sonata, mvt. 1 (from the top to the introduction of second subject group), played by three professional pianists for multiple takes. The figure shows that the interpretation sequence tends to remain similar within each performer, and different among different performers.

We can also show where in the music score has the highest variance of interpretation vector among the nine takes. Figure 7 shows the variance for each note, from which we can see that the maximum variance occurs at the structural boundary from the first subject group to the transition, supporting findings that music expression varies significantly at structural boundaries [19].

4. EVALUATION

We evaluate our method's capability to (1) create a natural expressive performance given the music score, and to encode an expressive performance into (2) perceptually relevant space that is (3) representative of the corresponding performance. For training, we used an in-house dataset comprising of 380 classical pieces, mostly Late romantic and Baroque. The piano performance comprised of 130 in-house performances and 250 performances obtained from the piano e-competition archive [1]. Some performances were different interpretations of the same piece. Furthermore, corresponding music scores were obtained. To generate a one-to-one mapping between the performance and the score, each performance was aligned to the music score using symbolic music alignment based on [16] with manual alignment correction. Then, the notes in the performance and the score were matched using maximum bipartite matching, using the pitch and the onset timing to determine the edge weights between notes in the score and the performance.

We have used 370 of 380 pieces for training. Of 370 pieces, all but 500 notes were used for training, and remaining notes for the optimization of the hyperparameters (validation). To test the decoder in experiment 1 and 2, we have sampled 10 piano music scores from the Mutopia project [2], and corrected the key signature, meter and downbeat positions. It contained a wide variety of pieces from Baroque to Ragtime. To test the encoder in experiment 3, we have used 10 pieces of 380 pieces that were not used for training.

4.1 Listening test

We evaluated the naturalness of the generated performance using different methods. For each of the ten pieces, we extracted a random 15-second segment, and generated performance using three methods: (1) method **Deadpan** directly played back the SMF data of the music score data that has been exported from MakeMusic's Finale Version 25; (2) method **Finale** used "Human Playback" function of Finale to create a natural performance, serving as a reproducible reference music performance method; and (3) the **proposed** method. Furthermore, we extracted seven 15-second excerpts from human playback in the training dataset, which serves as an **oracle** method. The selected pieces were different from the first three methods, because the human playing data to the first three were not always available. To make a fair comparison, we removed the sustain pedal data from the oracle, because the pedal is contained in none of the other methods.

Next, the MIDI data were synthesized using a high-quality piano synthesizer and presented in a random order to 11 participants. The participants were asked to evaluate the naturalness and expressiveness, on a rating from 1 to 5, 1 being unnatural or unexpressive, 3 being moderately natural or contains some expression, and 5 being the most

[1] www.piano-e-competition.com
[2] www.mutopiaproject.org

Table 1. Mean opinion scores of the performances

	Deadpan	Finale	Proposed	Oracle
Naturalness	3.02	3.08	**3.29**	3.23
Expressiveness	2.75	2.98	3.14	**3.62**

natural or expressive. The participants were between age 25 and late fifties, working on music technology.

The results are shown in Table 1. Kruskal-Wallis H-test was used for test of significance ($p < 0.01$), since Shapiro test showed non-normality. We found that the effects were significant for both expressiveness and naturalness, for all of the method pairs.

This shows that the system is capable of generating a performance that is as natural as human playing, but the expression has a room for improvement. One possible reason is that the system does not make use of music score markings like expression and phrase markings. The audio on our demo webpage suggests nonetheless that the system learns to "improvise" a noticeable evolution of musical expression over a timespan of multiple notes.

4.2 Test on the perception of the interpretation vector

Second, we analyzed the capability of the interpretation sequence to create a *perceptually distinguishable and consistent* space of musical interpretation.

First, a pair of non-identical and non-overlapping segments were sampled from a piece, which we call segments A and B. Second, a vector Δ was sampled from a 5-dimensional, zero-mean and unit-variance Normal distribution. Third, two interpretations are generated for each of A and B, where rendition A_1 and B_1 are generated with interpretation vector $\mu(h_{n-1}) + \Delta \otimes \exp(\gamma(h_{n-1}))$, and A_2 and B_2 with $\mu(h_{n-1}) - \Delta \otimes \exp(\gamma(h_{n-1}))$. Fourth, A_1, A_2, B_1 and B_2 were successively presented to the participants, randomly presenting B_2 before B_1. They were asked to identify whether the relationship of music expression between $A_1 \rightarrow A_2$ is the same as $B_1 \rightarrow B_2$ or $B_1 \leftarrow B_2$. The participants rated the perceived relationship with a confidence on a two-point scale 2 (0 means cannot tell, 2 means strongly confident about the response). We repeated this experiment on 9 other pairs of segments. After the experiment, we changed the signs of the responses, such that negative confidence indicates the wrong guess ($A_1 \rightarrow A_2$ is $B_1 \leftarrow B_2$), and positive indicates the right guess ($A_1 \rightarrow A_2$ is $B_1 \rightarrow B_2$).

The average rating was 0.34, meaning that there is an agreement between the change of interpretation vector and human perception of interpretation. To test the significance, Wilcoxon signed rank test was used under the null hypothesis that the median rating is zero, since Shapiro test showed non-normality. The effect was significant with $p < 0.01$.

This shows that the encoder does capture a perceptually coherent space of music interpretation, and its modification creates a perceptually consistent difference in the interpretation. Qualitatively, we found that by changing the interpretation vector, there are simultaneous and smooth changes not only in the dynamics and the tempo but also nontrivial aspects like breaking of the chords or the articulation of the accompaniment.

Table 2. Pearson's correlation coefficient between the generated performance and the true performance.

	Num. notes	$z_n = 0$	$z_n \sim$ prior
Velocity	10	0.61	0.72
	40	0.37	0.68
Onset deviation	10	0.02	0.07
	40	0.19	0.24
Duration	10	0.77	0.77
	40	0.82	0.82
BPM	10	0.07	0.21
	40	0.00	0.14

4.3 Test on the predictive capability of the encoder

Finally we evaluated the capability to encode a given performance, by priming h and z with an encoded true human performance, and comparing the true human and the generated performances. First, we primed the decoder by feeding to the encoder the first 20 notes to a performance to infer the initial value of z_n. Next, the remaining 10 or 40 notes were decoded using two different decoders: (1) use z_n sampled from the prior that has been primed in the first step, and (2) fix $z_n = 0$. We repeated this experiment on 100 different starting points, and evaluated the Pearson's correlation coefficient between the generated and human performance.

Table 2 shows the correlation coefficients. The result shows that the encoder is capable of encoding information pertinent to music performance. Significant improvements are seen for the velocity and the tempo, showing that the interpretation space is captures these aspects. At the same time, there are still rooms for improving the prediction of the onset deviation and the tempo.

This shows that we can indeed feed a reference performance snippet to the system, and the system would generate a performance in style of the snippet.

5. CONCLUSION

We have presented music performance rendering method that explicitly encodes underlying sources of expression variations. Our method opens door to wider possibilities for the co-creation of music performance between a machine and humans, by enabling an interpretable and adjustable performance rendering and analysis system. We also believe the capability to decouple musical intent and its execution given the intent opens door to a new abstraction layer for performance analysis.

Future work includes the inference of additional outputs like the pedals and note-off velocity, better expressiveness through incorporation of music score markings, disentanglement of the latent space and interfaces for interactive music performance rendering.

6. REFERENCES

[1] Sergio Canazza, Giovanni De Poli, and Antonio Roda. Caro 2.0: An interactive system for expressive music rendering. *Advances in Human-Computer Interaction*, 2015:1–13, 2015.

[2] Carlos Cancino-Chacón and Maarten Grachten. The basis mixer: a computational romantic pianist. In *Late-Breaking Demo, ISMIR*, 2016.

[3] Carlos Cancino-Chacón and Maarten Grachten. A computational study of the role of tonal tension in expressive piano performance. In *Proc. ICMPC*, 2018.

[4] Carlos E. Cancino-Chacón, Maarten Grachten, Werner Goebl, and Gerhard Widmer. Computational models of expressive music performance: A comprehensive and critical review. *Frontiers in Digital Humanities*, 5:25, 2018.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. EMNLP*, pages 1724–1734, 2014.

[6] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *NIPS*, pages 2980–2988, 2015.

[7] Simon Dixon, Werner Goebl, and Gerhard Widmer. The "Air Worm": an interface for real-time manipulation of expressive music performance. In *Proc. ICMC*, 2005.

[8] Sebastian Flossmann, Maarten Grachten, and Gerhard Widmer. Expressive performance rendering: Introducing performance context. *Proc. SMC*, pages 155–160, 2009.

[9] Anders Friberg and Erica Bisesi. Using computational models of music performance to model stylistic variations. *Expressiveness in music performance: Empirical approaches across styles and cultures*, pages 240–259, 2014.

[10] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proc. CVPR*, pages 4700–4708, 2017.

[11] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. Graph neural network for music score data and modeling expressive piano performance. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proc. ICML*, pages 3060–3070, 2019.

[12] Tae Hun Kim, Satoru Fukayama, Takuya Nishimoto, and Shigeki Sagayama. Polyhymnia: An automatic piano performance system with statistical modeling of polyphonic expression and musical symbol interpretation. In *Proc. NIME*, pages 96–99, 2011.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015.

[14] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *Proc. ICLR*, 2014.

[15] Akira Maezawa. Deep linear autoregressive model for interpretable prediction of expressive tempo. In *Proc. SMC*, pages 364–371, 2019.

[16] Akira Maezawa and Kazuhiko Yamamoto. MuEns: A multimodal human-machine music ensemble for live concert performance. In *Proc. CHI*, pages 4290–4301, 2017.

[17] Dasaem Jeong Taegyun Kwon Juhan Nam. VirtuosoNet: A hierarchical attention RNN for generating expressive piano performance from music score. In *Workshop on Machine Learning for Creativity and Design, NeurIPS*, pages 1–6, 2018.

[18] Kenta Okumura, Shinji Sako, and Tadashi Kitamura. Laminae: A stochastic modeling-based autonomous performance rendering system that elucidates performer characteristics. In *Proc. ICMC*, 2014.

[19] Caroline Palmer. Music performance. *Annual review of psychology*, 48(1):115–138, 1997.

[20] Ian Simon and Sageev Oore. Performance RNN: Generating music with expressive timing and dynamics. `https://magenta.tensorflow.org/performance-rnn`, 2017.

[21] Sam Van Herwaarden, Maarten Grachten, and W Bas De Haas. Predicting expressive dynamics in piano performances using neural networks. In *Proc. ISMIR*, pages 45–52, 2014.

[22] Gerhard Widmer, Sebastian Flossmann, and Maarten Grachten. YQX plays Chopin. *AI Magazine*, 30(3):35, 2009.

AN INTERACTIVE WORKFLOW FOR GENERATING CHORD LABELS FOR HOMORHYTHMIC MUSIC IN SYMBOLIC FORMATS

Yaolong Ju[1] **Samuel Howes**[1] **Cory McKay**[2]
Nathaniel Condit-Schultz[3] **Jorge Calvo-Zaragoza**[4] **Ichiro Fujinaga**[1]
[1] Schulich School of Music, McGill University, Canada
[2] Department of Liberal and Creative Arts, Marianopolis College, Canada
[3] School of Music, Georgia Institute of Technology, USA
[4] Department of Software and Computing Systems, University of Alicante, Spain
yaolong.ju@mail.mcgill.ca, samuel.howes@mail.mcgill.ca, cory.mckay@mail.mcgill.ca,
natcs@gatech.edu, jcalvo@dlsi.ua.es, ichiro.fujinaga@mcgill.ca

ABSTRACT

Automatic harmonic analysis is challenging: rule-based models cannot account for every possible edge case, and manual annotation is expensive and sometimes inconsistent, undermining the training and evaluation of machine learning models. We present an interactive workflow to address these problems, and test it on Bach chorales. First, a rule-based model was used to generate preliminary, consistent chord labels in order to pre-train three machine learning models. These four models were grouped into an ensemble that generated chord labels by voting, achieving 91.4% accuracy on a reserved test set. A domain expert then corrected only those chords that the ensemble did not agree on unanimously (20.9% of the generated labels). Finally, we used these corrected annotations to re-train the machine learning models, and the resulting ensemble attained an accuracy of 93.5% on the reserved test set, a 24.4% reduction in the number of errors. This versatile interactive workflow can either work in a fully automatic way, or can capitalize on relatively minimal human involvement to generate higher-quality chord labels. It combines the consistency of rule-based models with the nuance of manual analysis to generate relatively inexpensive high-quality ground truth for training effective machine learning models.

1. INTRODUCTION AND BASIC METHODOLOGY

In general, harmonic analysis refers to the identification of harmonies from the musical surface. As a key part of the foundation of modern Western music theory, harmonic analysis is inherently complex. It is based on low-level sensory distinctions (consonance vs dissonance), local constructs (counterpoint, voice-leading), and global musical structures (harmonic function, form, tonality, etc.). Learning it is thus a time-consuming process, requiring years of training. Furthermore, many prominent music theorists (e.g., Rameau, Riemann, Schenker) have proposed different approaches to harmonic analysis. This means it is often possible to analyze the same passage in numerous legitimate ways. For example, some analysts prefer interpretations with fewer chords, while others prefer interpretations with more frequent harmonic changes. We characterize these general strategies as "melodic" and "harmonic", respectively (Fig. 1 illustrates these interpretive strategies). Complicating matters further, analysts often disagree, and are not always internally consistent [12]. Given the complexity, subjectivity, and inconsistency of harmonic analysis, it is challenging to systemize it.

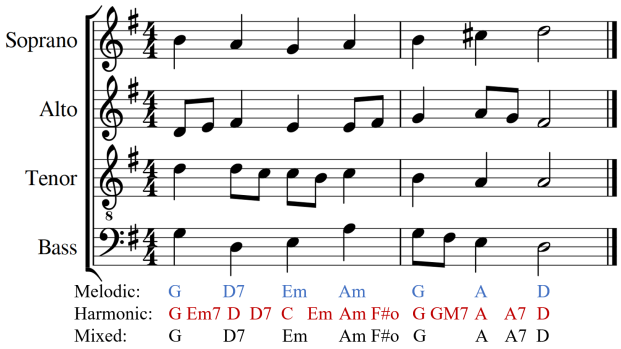


Figure 1. A passage with important differences between melody-oriented (blue) and harmony-oriented (red) analyses. The final analysis (black) mixes the two styles. Such inconsistencies are quite common, even between expert analyses.

In spite of these challenges, there have been various attempts to automate harmonic analysis. Data generated by automated approaches could be used to populate a large-scale, searchable database, which would serve as an invaluable resource for music research. For example, such a database could be used in corpus studies to answer re-

search questions about musical style or the development of modern harmonic practices. Automatic harmonic analysis can also be used in automatic composition and interactive accompaniment systems.

Some researchers have developed rule-based (RB) models for automatic harmonic analysis [4, 8, 10, 21–23]. Although these approaches generate chord labels that are internally consistent, they often fail to produce correct analyses for even moderately exceptional passages, as it is extremely complicated to define rules that are comprehensive enough to account for all possibilities.

Other researchers have made use of manual annotations by experts, who can better respond to exceptions [2,5,6,9,16,17]. Such ground truth can be used to train machine learning (ML) models for automatic harmonic analysis [3, 11, 14, 15, 18, 20, 24]. Although the annotations created by human analysts are more nuanced, manual harmonic annotations require an enormous amount of time and expertise, and can be inconsistent [12], which may undermine a ML model's effectiveness, especially when limited amounts of training data are available.

Due to these difficulties, few large high-quality datasets and automatic harmonic analysis models exist, a situation that has significantly limited the computational study of Western harmony.

In this paper, we combine the strengths of existing approaches to address the common problems of automatic harmonic analysis within a single interactive workflow, using a set of largely homorhythmic [1] Bach chorales. The proposed workflow is illustrated in Fig. 2 and described below:

1. To solve the problem of analytical inconsistency, we use an existing RB model [4] to generate preliminary, consistent chord labels according to a particular analytical style.

2. These analyses are used to pre-train three ML models, [2] which together with the RB model form an algorithm ensemble, where each model within the ensemble labels all the chords. The most-preferred chord labels [3] are then output as *Analysis 1*.

3. To improve the quality of the analyses, a human expert examines only those chords for which the ensemble did not agree unanimously, and corrects them as needed. We call this process "partial manual modification". Compared to annotating chorales from scratch, the amount of required work for the expert is significantly reduced. The first three steps of this workflow are shown in Part 1 of Fig. 2.

4. Once the expert's corrections are obtained (*Analysis 2*), we re-train the ML models. The most-preferred chord labels from the new ensemble are chosen as the final chord labels (*Analysis 3*), which is shown in Part 2 of Fig. 2. This paradigm of manually modifying the generated data and re-training the ML models is known as "interactive machine learning" [1,7].

This workflow is not limited to Bach chorales. With an adapted RB model (Model 4 in Fig. 2), it can easily be applied to other genres of music in a fully automatic way (ending with Analysis 1) or interactively if an expert analyst is available (ending with Analysis 3). The source code, data, and results from this project can be found at: https://bit.ly/2QUdGwH.

2. DETAILS OF METHODOLOGY

This section introduces additional details of the interactive workflow shown in Fig. 2 and described in Section 1.

2.1 Input Data Encoding and Processing

The workflow currently accepts music encoded in Humdrum's **kern symbolic representation. Any other formats that can be faithfully converted to **kern can also be used.

Each chord label consists of the letter-name of the root and the quality of the chord (e.g., C major). Triads can be major, minor, or diminished; and seventh chords can be major, minor, dominant, half-diminished, or fully diminished. Functional Roman numerals are not used, and chordal inversions are not specified.

Chord labels are appended to the original **kern file for each chorale and aligned with the music as "onset slices" [11,13], as shown in Fig. 4. An onset slice is formed whenever a new note onset occurs in *any* musical voice, and consists of a list of all pitch classes sounding at that moment.

Additionally, all chorales and corresponding chord labels were transposed to the same key to make the tonal relationships between pitch classes consistent across the dataset. [4]

2.2 Input Features

Each onset slice is mapped to a feature vector for processing by Model 1, Model 2, and Model 3 of the workflow. These features, [5] and the codes used to refer to them in Section 3, are as follows:

1. **PC12** : A 12-D binary vector of enharmonic pitch classes present in the slice.

2. **M**: A 3-D indication of the metrical context of the slice (down-beat, on-beat, off-beat).

3. **O**: A 12-D vector indicating which PC12 pitch classes are real onsets and which are artificial (see Fig. 4).

4. **Wn**: A variable size vector containing the (non-Wn) features from the n previous and following slices

[1] Homorhythm is a texture where all parts share a very similar rhythm, as in Fig. 1. It is commonly used in hymn and chorale settings.

[2] See the caption of Fig. 2 for the details of these models.

[3] If there is a tie, prefer the label for which the rule-based algorithm voted.

[4] The built-in key transposition function from music21 was used, with the Aarden-Essen key profile (https://bit.ly/2FSIwQY). Chorales were transposed to C major or A minor depending on their mode.

[5] A multi-label one-hot schema was used to encode the features as inputs for the ML algorithms.

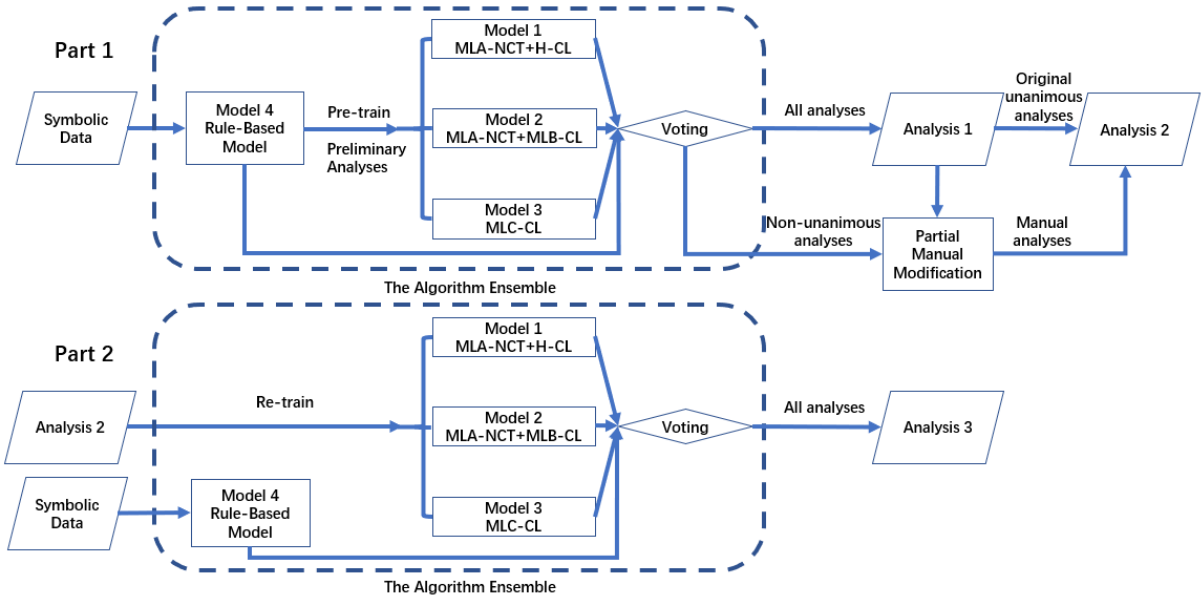


Figure 2. Interactive workflow for automatic harmonic analysis. There are four models within the algorithm ensemble, three of which are trainable. Models 1 and 2 both use a machine learning algorithm (MLA) to identify and remove non-chord tones (NCTs). After this, Model 1 (MLA-NCT+H-CL) uses a heuristic (H) algorithm and Model 2 (MLA-NCT+MLB-CL) uses a ML algorithm (MLB) to infer chord labels (CL) from the remaining chord tones. We term this process "NCT-first harmonic analysis", as shown on the right side of Fig. 3. Model 3 (MLC-CL) uses a single ML algorithm (MLC) to infer chord labels (CL) directly from the pitch-class collections, without removing NCTs. We term this process "direct harmonic analysis", as shown on the left side of Fig. 3.

(e.g., W1 indicates that features for the directly preceding and directly following slices are included in the features of the current slice). These surrounding slices are called "contextual windows".

The workflow allows for experimentation with different feature configurations. For example, a "PC12M" configuration indicates a 15-D vector, with O and Wn features omitted. This notation is adopted in Section 3.

2.3 Rule-Based Algorithms

We use an existing RB model [4] to generate preliminary chord labels (Model 4 in Fig. 2). This tool is publicly accessible online. [6] A "harmonic" rather than "melodic" style of analysis is used (see Fig. 1), which prefers more chord changes and fewer non-chord tones (NCTs) [19], and is better-suited to the typical chorale texture. An overview of the specific heuristics of this style can be found at: https://bit.ly/2XCmNVo. We also used a heuristic algorithm (H-CL from Fig. 2) in Model 1 to infer chord labels from remaining chord tones. The details of this algorithm can be found at: https://bit.ly/2MBL0dp.

2.4 Machine Learning Algorithms

As shown in Fig. 2, the workflow includes three ML algorithms (MLA, MLB, and MLC) to pre-train. MLA treats NCT identification as a multi-label problem; the output of MLA is a 12-dimensional vector specifying which pitch classes are both present and identified as NCTs; MLB and MLC treat chord labeling as a multi-class problem; they output similar vectors identifying the predicted chord label among all candidates.

We tested Support Vector Machines (SVMs) and Deep Neural Networks (DNNs) as MLA, MLB, and MLC classifiers. For DNN, we used three hidden layers, each with 300 hidden units. Adaptive Moment Estimation was used as an optimizer, with loss functions of binary cross-entropy for MLA and categorical cross-entropy for MLB and MLC. SVM used a linear kernel function.

3. EXPERIMENTS

3.1 Data

The experiments below were performed on a modified [7] dataset of Bach chorales originally produced by Craig Sapp. [8] This modified dataset consists of 369 chorales.

To evaluate the performance of our workflow, 39 chorales were randomly chosen before the experiments began and partitioned into a set reserved for final testing in Experiment 2. These reserved chorales had their chords hand-labelled in their entirety by a human expert.

The remaining 330 non-reserved chorales were used for training, validation (early-stopping) and internal testing.

[6] https://bit.ly/2Gh6IhA

[7] Available at: https://bit.ly/2VWHB8w. Some corrections were made to the music and Chorale 150 was added to the dataset. Chorales 130 and 316 were excluded, since the original **kern files and the music21-parsed results are different.

[8] https://bit.ly/2D4ju10

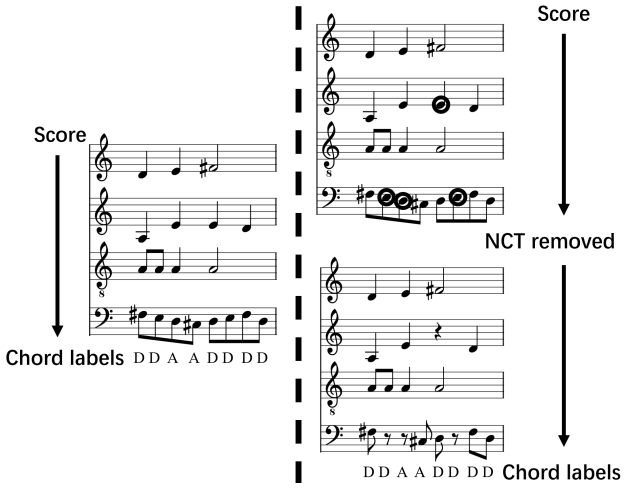


Figure 3. Comparison of "direct harmonic analysis" (left, used by Model 3 in Fig. 2) and "NCT-first harmonic analysis" (right, used by Model 1 and Model 2 in Fig. 2) approaches to automatic harmonic analysis. The former identifies chords directly from the score, while the latter first identifies and removes non-chord tones from the score, and then generates chord labels from the remaining chord tones.

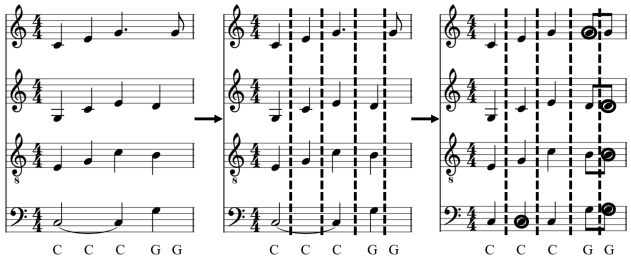


Figure 4. Illustration of note onset slices, aligned with chord labels. An onset slice is created whenever a new note onset occurs in *any* musical voice (middle). Any note sustained from a previous slice becomes an "artificial onset" in the new slice (right, circled).

The initial "ground truth" for these remaining 330 chorales consisted of the labels predicted by the RB model (Model 4), which was found to be quite effective, if not perfect [4]. This imperfect "ground truth" was used in Experiment 1 (see Section 3.2) to get a preliminary sense of how well the workflow's component classifiers performed. Final evaluation was performed in Experiment 2 (see Section 3.3) with the proper, hand-annotated 39-chorale reserved test set.

3.2 Experiment 1

Experiment 1 tested the effectiveness of several different workflow configurations by experimenting on varying input features and learning algorithms (see Section 2.4). The performance of Models 1, 2, and 3 from Fig. 2 were tested.

3.2.1 Experimental Setup

Ten-fold cross-validation was performed on the 330 non-reserved chorales described in Section 3.1. For the DNN experiments, we divided the non-reserved portion of the dataset (330 chorales) into training (80%), validation (10%) and internal testing (10%) folds. The SVM data was divided into training (90%, the union of the DNN training and validation sets) and internal testing (10%, matching the DNN internal test sets) folds. When the W features were included (see Section 2.2), n was set to 1 for MLA and MLC, and to 2 for MLB (represented as W1/2).

3.2.2 Results

The results of Experiment 1 are shown in Table 1. The highest classification value of 90.1% was achieved by Model 2 using PC12MOW1/2 input features. Results show that the addition of a small contextual window (feature Wn) improved the performances of Model 2 and Model 3 significantly.[9] This reflects the general music theoretical understanding that, in cases of ambiguous harmony (e.g., an incomplete chord), a chord's immediate context is essential to label it properly.

It is important to note that these Experiment 1 findings are based on imperfect ground truth (see Section 3.1), and so must be interpreted more as preliminary indications rather than as confirmed truth. Experiment 2 was performed in order to obtain more empirically meaningful results.

3.3 Experiment 2

Experiment 2 compared the performance of the classifier ensemble after fully automated training (Analysis 1 in Fig. 2) with that of the ensemble after human-assisted re-training (Analysis 3 in Fig. 2). This set of experiments involved evaluation on a reserved expert-labelled test set (see Section 3.1).

3.3.1 Experimental Setup

Classification models were pre-trained, had their outputs manually corrected, re-trained, and tested using the full workflow described in Section 1. Pre-training was done using the Model 4 output, just as in Experiment 1.

For the DNN training, we used 90% of the 330 non-reserved chorales as the training set and 10% as the validation set. A cross-validation-like training scheme was used: we conducted 10 experiments by training 10 models with rotated training and validation folds, while the testing fold (39 reserved chorales) remained the same. All 330 non-reserved chorales were used to train each of the SVM classifiers. Only the PC12MOW1/2 input features (see Section 2.2) were used in Experiment 2. For the W features, n was set to 1 for MLA and MLC, and to 2 for MLB (represented as W1/2).

Once Analysis 1 (see Fig. 2) was obtained, the human expert manually corrected only those chords that the ensemble did not agree on unanimously. The corrected labels (Analysis 2) were then used to re-train Models 1, 2, and 3. The 39 manually-labelled reserved test chorales were then used to test the original pre-trained models, and then the

[9] $p<0.05$ in Students' t-tests comparing all Model 2 and 3 accuracies for PC12 and PC12M with those of PC12W1/2 and PC12MW1/2.

Model	Metric	PC12	PC12M	PC12W1/2	PC12MW1/2	PC12MOW1/2
SVM	CA1	**81.7±1.4%**	81.6±1.4%	82.7±1.0%	83.0±1.0%	83.5±0.9%
	CA2	73.0±1.5%	73.1±1.6%	85.4±1.3%	86.1±1.5%	87.4±1.5%
	CA3	74.9±1.6%	75.6±1.5%	85.4±1.3%	85.9±1.3%	87.7±1.5%
DNN	CA1	81.0±1.5%	**81.7±1.5%**	85.3±0.9%	85.6±0.9%	85.8±0.9%
	CA2	74.2±1.8%	75.1±1.6%	**88.5±1.3%**	**89.6±1.3%**	**90.1±1.5%**
	CA3	74.6±1.8%	75.3±1.4%	87.5±1.7%	88.3±1.7%	89.0±2.0%

Table 1. Experiment 1 cross-validation classification accuracies, averaged across folds. Uncertainty values indicate standard error across folds. Values indicate the percentage of onset slices "correctly" classified by Model 1 (CA1), Model 2 (CA2), and Model 3 (CA3), based on the Model 4 "ground truth". Columns indicate features (see Section 2.2) and rows indicate machine learning algorithms (see Section 2.4). The best performance in each column is highlighted in bold.

Model	Metric	PC12MOW1/2 Pre-trained	PC12MOW1/2 Re-trained
SVM	CA1	85.9%	87.0%
	CA2	88.6%	89.8%
	CA3	87.7%	89.3%
	CAVote	**91.4%**	92.7%
	PUA	79.1%	79.0%
DNN	CA1	85.4±0.2%	88.1±0.2%
	CA2	88.9±0.3%	91.3±0.4%
	CA3	87.9±0.7%	90.5±0.3%
	CAVote	90.9±0.2%	**93.5±0.2%**
	PUA	80.4±1.2%	79.7±0.4%
RB	CA4	90.7%	

Table 2. Experiment 2 classification accuracies on the reserved test set. DNN values are averaged across models trained using different training/validation sets, and uncertainty values indicate standard error across these folds. Values indicate how many onset slices were correctly classified by Model 1 (CA1), Model 2 (CA2), Model 3 (CA3), Model 4 (CA4), the ensemble as a whole (CAVote), and just those CAVote predictions that were unanimous (PUA). "PC12MOW1/2" indicates the input features (see Section 2.2. "Pre-trained" indicates performance before manual correction (i.e., Analysis 1 in Fig. 2), and "Re-trained" indicates performance after re-training on the corrected data (i.e., Analysis 3 in Fig. 2). The best performance in each column is highlighted in bold.

re-trained models. Performance on this reserved test set is shown in Table 2.

3.3.2 Results

One can see in Table 2 that the original RB algorithm (Model 4 in Fig. 2) attains a chord accuracy of 90.7%, which serves as our baseline. The highest accuracy obtained by the pre-trained ensemble is 91.4%, using PC12MOW1/2, SVM classifiers, and voting. This (pre-trained) performance is achieved without any expert human intervention. It is of interest that CAVote here is higher than CA4, even though the classifiers in CAVote were trained on the RB output; this is perhaps because the RB model is overfitting the theoretical model underlying it, and that the pre-trained ensemble trained on it may in fact be smoothing out some of this overfitting to result in a slightly more general model. A comparison of Table 1 and Table 2 indicates that the Table 1 performance with artificial ground truth is quite similar to the performance of Table 2 pre-trained classifiers on the proper test set; this encouragingly suggests that there is little or no overfitting.

Table 2 also shows that performance improved after re-training in most cases. [10] The best-performing [11] configuration attains an accuracy of 93.5%, using voting DNNs trained on PC12MOW1/2 features.

The partial manual modification workflow is also found to be relatively efficient, as the expert analyst is only required to provide manual analyses for about 20.9% [12] of all slices. Compared to examining and annotating every slice, the amount of required work is reduced substantially.

4. DISCUSSION

According to the results, our interactive workflow performed well on the Bach dataset using a "harmonic" style of analysis. It was found that quite good performance could be achieved with our rule-based model (90.7% on the reserved test data), that performance could be improved slightly using the RB model to self-train a classifier ensemble (91.4% on the test data), and that still greater improvements resulted from partial manual modification and re-training (93.5% on the test data). Although these improvements may seem small in absolute terms, they are statistically significant, and they represent meaningful fractional decreases in the error rate (drops of 7.5% comparing pre-trained CAVote to RB, 30.1% comparing re-trained CAVote to RB, and 24.4% comparing re-trained CAVote to pre-trained CAVote). Of particular importance, the first two approaches require no human intervention, and the third requires much less expert labor than full manual annotation.

Figure 5 provides an illustration of how this approach can be effective, using an excerpt from one of the test set chorales. Although some algorithms within the ensemble

[10] $p<0.05$ in Students' t-tests comparing results before and after re-training for CA1, CA2, CA3, and CAVote, but not PUA.

[11] $p<0.05$ in Students' t-tests comparing results of CAVote to CA1, CA2, CA3, and CA4.

[12] This value is inferred from Table 2: 100% - PUA.

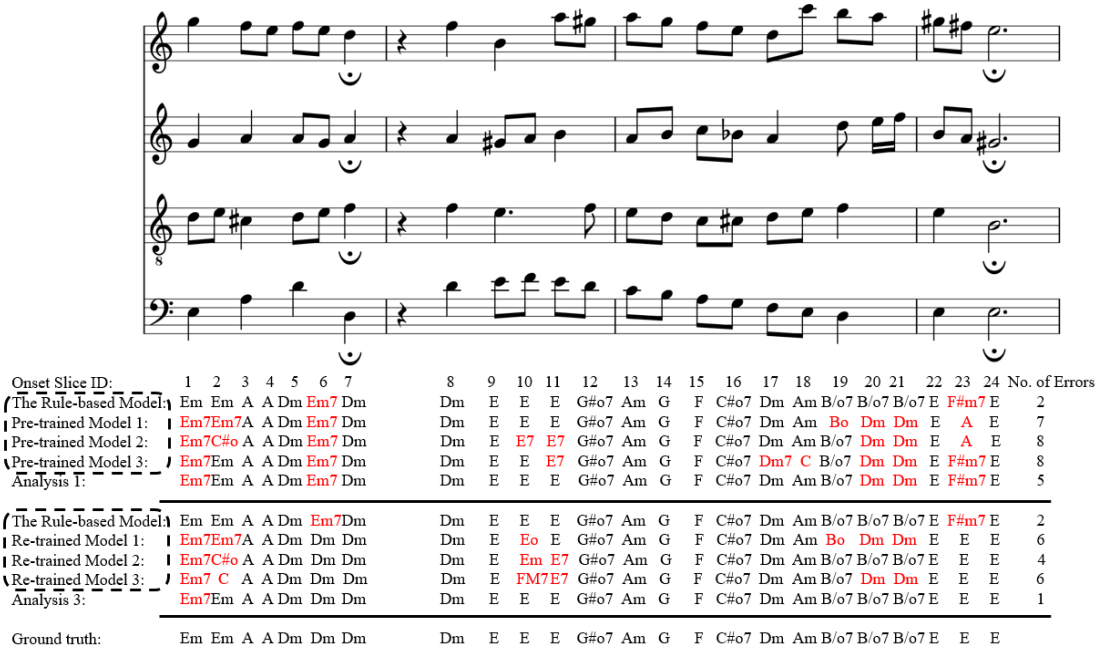


Figure 5. An illustration of how classifications evolve as processes proceed as outlined in Fig. 2, based on measures 9 through 12 of BWV 315 "Gib dich zufrieden und sei stille". Chord labels were generated by a DNN-based algorithm ensemble using PC12MOW1/2 features (see Section 2.2). The algorithm ensemble is made up of the four models within the dashed rectangle, which vote to generate Analyses 1 and 3. The labels above the first horizontal line were generated in a fully automatic way, without any human intervention. The labels between the two horizontal lines (other than the rule-based model) were generated automatically after re-training on partially corrected data. The chord labels highlighted in red are errors compared to the ground truth provided by an expert analyst.

make errors, the re-trained ensemble ultimately generates better answers in Analysis 3. Upon examining the errors, we find that some of them are reasonable alternative versions of the ground truth: chords with the same roots, but with or without an added seventh (slices 1, 11, 17, and 19); or chords that are subsets of the ground truth chords (slices 20 and 21). As a result, some of the "errors" that the ensemble makes in this particular excerpt are in fact theoretically acceptable answers. This is encouraging, as it suggests that at least some of the "mistakes" made by the classifiers may not in fact be mistakes at all. We still count them as mistakes, however, because consistency in analytical style is one of the goals of this work.

5. CONCLUSION AND FUTURE RESEARCH

We present a versatile interactive workflow for generating chord labels for homorhythmic music. It can be used in a fully automatic way or, with a relatively small amount of effort from an expert human analyst who corrects a small, automatically selected fraction of the generated analyses, a re-trained classifier ensemble can be produced that performs even better.

Results show that this workflow is quite compelling: it combines the consistency of rule-based models with the nuance of manual analysis to generate relatively inexpensive, high-quality ground truth for training effective machine learning models. The resulting classifier ensemble is able to automatically generate highly consistent and accurate chord labels, which can serve as invaluable resources for musicians, composers, and music researchers alike.

There are currently a few limitations to our research. First, music21's automatic key-finding might not be ideal for our dataset (early tonal music), and may have resulted in reduced performance due to faulty transpositions. Instead of transposing all chorales to the same key, a better, but more complicated solution would be to augment our data by transposing all chorales to all 12 possible keys. Second, the RB model can be improved to include chords of other qualities (e.g., augmented-sixth chords). Finally, the ground-truth annotations were prepared by a single expert annotator, and it would be better to repeat this process using annotations from multiple experts.

An important next step will be to test this workflow using other analytical styles (e.g., the "melodic" style), which can be done simply by specifying different heuristics in the RB model. We also plan to tackle the larger category of homophonic music, which includes any music with a primary melodic line accompanied by harmonic support. A greater variety of homophonic textures poses a challenge to our RB model because more individual onset slices are harmonically ambiguous, requiring larger contextual windows to correctly interpret the harmony. In light of this, we will modify our workflow to address homophonic music accordingly. Finally, we will investigate training and evaluation protocols that permit multiple valid chord labels per slice.

ACKNOWLEDGEMENT

We would like to thank the Social Sciences and Humanities Research Council of Canada (SSHRC) and the Fonds de recherche du Québec-Société et culture (FRQSC) for their generous funding. We would also like to acknowledge the contributions of our many collaborators on the Single Interface for Music Score Searching and Analysis (SIMSSA) project, especially Emily Hopkins and Gabriel Viglienson.

6. REFERENCES

[1] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120, 2014.

[2] John Ashley Burgoyne, Jonathan Wild, and Ichiro Fujinaga. An expert ground truth set for audio chord recognition and music analysis. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 633–638, 2011.

[3] Tsung-Ping Chen and Li Su. Functional harmony recognition of symbolic music data with multi-task recurrent neural networks. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*, pages 90–97, 2018.

[4] Nathaniel Condit-Schultz, Yaolong Ju, and Ichiro Fujinaga. A flexible approach to automated harmonic analysis: Multiple annotations of chorales by Bach and Prætorius. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*, pages 66–73, 2018.

[5] Trevor de Clercq and David Temperley. A corpus analysis of rock harmony. *Popular Music*, 30(1):47–70, January 2011.

[6] Johanna Devaney, Claire Arthur, Nathaniel Condit-Schultz, and Kirsten Nisula. Theme and variation encodings with roman numerals (TAVERN): A new data set for symbolic music analysis. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, pages 728–734, 2015.

[7] Jerry Alan Fails and Dan R. Olsen Jr. Interactive machine learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pages 39–45, 2003.

[8] Mark Granroth-Wilding. *Harmonic analysis of music using combinatory categorial grammar*. PhD thesis, The University of Edinburgh, 2013.

[9] Thomas Hedges and Martin Rohrmeier. Exploring Rameau and beyond: A corpus study of root progression theories. In *Proceedings of the 1st International Conference on Mathematics and Computation in Music*, pages 334–337, 2011.

[10] Tim Hoffman and William P. Birmingham. A constraint satisfaction approach to tonal harmonic analysis. Technical report, *Electrical Engineering and Computer Science Department. CSE-TR-397-99. University of Michigan*, 2000.

[11] Yaolong Ju, Nathaniel Condit-Schultz, Claire Arthur, and Ichiro Fujinaga. Non-chord tone identification using deep neural networks. In *Proceedings of the 4th International Workshop on Digital Libraries for Musicology*, pages 13–16, 2017.

[12] Hendrik Vincent Koops, W. Bas de Haas, John Ashley Burgoyne, Jeroen Bransen, Anna Kent-Muller, and Anja Volk. Annotator subjectivity in harmony annotations of popular music. *Journal of New Music Research*, 48:1–21, 2019.

[13] Pedro Kröger, Alexandre Passos, Marcos Sampaio, and Givaldo De Cidra. Rameau: A system for automatic harmonic analysis. In *Proceedings of International Computer Music Conference*, pages 273–281, 2008.

[14] Kristen Masada and Razvan Bunescu. Chord recognition in symbolic music using semi-Markov conditional random fields. In *Proceedings of the 18th International Society for Music Information Retrieval Conference*, pages 272–278, 2017.

[15] Lesley Mearns. *The computational analysis of harmony in Western art music*. PhD thesis, Queen Mary University of London, 2013.

[16] Néstor Nápoles López. *Automatic harmonic analysis of classical string quartets from symbolic score*. Master's thesis, Universitat Pompeu Fabra, 2017.

[17] Markus Neuwirth, Daniel Harasim, Fabian Claude Moss, and Martin Rohrmeier. The annotated Beethoven corpus (ABC): A dataset of harmonic analyses of all Beethoven string quartets. *Frontiers in Digital Humanities*, 5:16, 2018.

[18] Alexandre Passos, Marcos Sampaio, Pedro Kröger, and Givaldo De Cidra. Functional harmonic analysis and computational musicology in Rameau. In *Proceedings of the 12th Brazilian Symposium on Computer Music*, pages 207–210, 2009.

[19] Ian Quinn. Are pitch-class profiles really key for key? *Zeitschrift der Gesellschaft für Musiktheorie [Journal of the German-speaking Society of Music Theory]*, 7(2):151–163, 2010.

[20] Christopher Raphael and Joshua Stoddard. Functional harmonic analysis using probabilistic models. *Computer Music Journal*, 28(3):45–52, 2004.

[21] Craig Stuart Sapp. Computational chord-root identification in symbolic musical data: Rationale, methods, and applications. *Computing in Musicology*, 15:99–119, 2007.

[22] Ricardo Scholz, Vitor Dantas, and Geber Ramalho. Automating functional harmonic analysis: The Funchal system. In *Proceedings of the 7th IEEE International Symposium on Multimedia*, pages 759–764, 2005.

[23] David Temperley and Daniel Sleator. Modeling meter and harmony: A preference-rule approach. *Computer Music Journal*, 23(1):10–27, 1999.

[24] Wan Shun Vincent Tsui. *Harmonic analysis using neural networks*. Master's thesis, University of Toronto, 2002.

QUANTIFYING MUSICAL STYLE: RANKING SYMBOLIC MUSIC BASED ON SIMILARITY TO A STYLE

Jeff Ens
Simon Fraser University
jeffe@sfu.ca

Philippe Pasquier
Simon Fraser University
pasquier@sfu.ca

ABSTRACT

Modelling human perception of musical similarity is critical for the evaluation of generative music systems, musicological research, and many Music Information Retrieval tasks. Although human similarity judgments are the gold standard, computational analysis is often preferable, since results are often easier to reproduce, and computational methods are much more scalable. Moreover, computation based approaches can be calculated quickly and on demand, which is a prerequisite for use with an online system. We propose StyleRank, a method to measure the similarity between a MIDI file and an arbitrary musical style delineated by a collection of MIDI files. MIDI files are encoded using a novel set of features and an embedding is learned using Random Forests. Experimental evidence demonstrates that StyleRank is highly correlated with human perception of stylistic similarity, and that it is precise enough to rank generated samples based on their similarity to the style of a corpus. In addition, similarity can be measured with respect to a single feature, allowing specific discrepancies between generated samples and a particular musical style to be identified.

1. INTRODUCTION

Measuring musical similarity is a fundamental challenge, related to many tasks in Music Information Retrieval (MIR). In this paper, we focus on measuring the similarity between a MIDI file and an arbitrary musical style. In a musical context, the term style can refer to historical periods, composers, performers, sonic texture, emotion, and genre [8]. Here, we use the term style to denote the musical characteristics exhibited by a corpus $\mathcal{C} = \{C_1, ..., C_n\}$, as expressed by a feature set $\mathcal{F}$. Depending on the contents of $\mathcal{C}$, style may correspond to something as specific as a subset of a composer's work, as general as the entirety of Western Classical Music, or as personal as the musical preferences of an individual.

We propose StyleRank [1], a method for ranking MIDI files based on their similarity to a style delineated by $\mathcal{C}$. It can be used as a tool for musicological research, to evaluate Style Imitation (SI) systems, and to filter the output of an SI system. An SI system aims to generate music that exhibits the stylistic characteristics of $\mathcal{C}$ [27]. The primary contributions are as follows: a collection of novel features for symbolic music representation; an efficient MIDI feature extraction tool written in C++ with bindings in Python; a measure of similarity with respect to an arbitrary style delineated by $\mathcal{C}$; and two experiments demonstrating that this measure is robust, and highly correlated with human perception of stylistic similarity.

2. MOTIVATIONS

There are several motivating factors for this research. In general, modelling human perception of musical similarity is of particular interest within the areas of Musicology, Music Cognition, and Music Theory [42]. Moreover, robust measures of musical similarity are critical for many MIR tasks, including database querying, music recommendation, and genre recognition. Although human perception is the gold standard for measuring musical similarity, natural human limitations place restrictions on the quantity and speed at which judgments can be collected, directly motivating automated measures of musical similarity.

More specifically, there are inherent challenges in designing a robust and reproducible listening experiment to evaluate SI systems. There are many variables which directly effect the quality of an experimental result, such as the number of participants, the listening environment, the sound equipment, and the number of samples selected for comparison. Even controlling for those variables, there is significant variability in how music is perceived, based on one's level of training [5] and musical background [13, 16, 32], which can result in a limited inter-rater agreement [34]. This is a particular issue, as it may hamper reproducibility and comparison with previously published results.

In most cases, sampling from an SI system is a stochastic process, and as a result, generated samples vary in quality. Developing a filtering process for generated material is a high priority concern, as low quality samples are undesirable when using a generative model in a production setting. Although measuring the log-likelihood of a sample can be

[1] The code is available at https://github.com/jeffreyjohnens/style_rank

useful as a proxy for quality, there are cases where log-likelihood significantly diverges from human perception. Theis et al. provide examples of generated images with high log-likelihood and extremely low quality [37]. To the best of our knowledge, there are no pre-existing methods for ranking generated samples with respect to an arbitrary style.

3. RELATED WORK

A wide variety of similarity measures have been developed to measure melodic [41], harmonic [9,25,31] and rhythmic similarity [38]. Many of these algorithms measure similarity by comparing two symbolic sequences [43]. Stylistic similarity, however, is rarely exhibited through sequence similarity, but rather through the repeated use of particular musical devices (i.e. melodic phrases, voice leading, and chord voicing) interspersed throughout the material [43]. In order to address this concern, approaches based on compression or pattern extraction have been proposed to measure similarity [2, 6, 21]. Since we aim to measure similarity with respect to $\mathcal{C}$, a more suitable approach will leverage information about the discriminative aspects of the entire corpus $\mathcal{C}$, rather than only taking two MIDI files into consideration.

In the context of SI system evaluation, the Turing Test [40] and the Consensual Assessment Technique [3] have been used to measure the stylistic similarity between generated artifacts $\mathcal{G} = \{\mathcal{G}_1, ..., \mathcal{G}_m\}$ and a particular style $\mathcal{C}$ [20,28]. Objective measures have also been used to evaluate SI systems. Dong et al. measure the ratio of empty bars, pitch class diversity, note duration, rhythmic consistency, and tonal distance [10]. Trieu and Keller propose a variety of metrics ranging from rhythmic variety to harmonic consistency [39]. Since these metrics produce a single scalar value, it is easy to compare $\mathcal{C}$ and $\mathcal{G}$. However, these high-level metrics are likely only capable of measuring stylistic similarity in a very general sense. Sturm and Ben-Tal. plot distributions of meter, mode, number of tokens, pitch and pitch class for $\mathcal{C}$ and $\mathcal{G}$, but do not provide an automated method for analyzing discrepancies [36].

More comprehensive methodologies have been proposed, which involve computing all pairwise inter-set distances between samples in $\mathcal{C}$ and $\mathcal{G}$ ($D_{\mathcal{CG}} = [\texttt{dist}(c, g) : (c \in \mathcal{C}) \wedge (g \in \mathcal{G})]$), as well as all pairwise intra-set distances for samples within a set ($D_{\mathcal{GG}} = [\texttt{dist}(g_i, g_j) : (g_i \in \mathcal{G}) \wedge (g_j \in \mathcal{G}) \wedge (g_i \neq g_j)]$). [2] CAEMSI [14], a domain independent framework for the analysis of SI systems, provides a statistical method to test the null hypothesis $H_0 : (D_{\mathcal{GG}} \neq D_{\mathcal{CG}}) \vee (D_{\mathcal{CC}} \neq D_{\mathcal{CG}}) \vee (D_{\mathcal{CC}} \neq D_{\mathcal{GG}})$ against the alternative hypothesis $H_1 : D_{\mathcal{GG}} = D_{\mathcal{CC}} = D_{\mathcal{CG}}$. Yang and Lerch extract multi-dimensional features from each MIDI file [44]. For each feature, $D_{\mathcal{CG}}$ and $D_{\mathcal{CC}}$ are constructed using Euclidean distance and smoothed using kernel density estimation [26, 33]. The distance between $D_{\mathcal{CC}}$ and $D_{\mathcal{CG}}$ is measured using (1) the area of overlap and (2) the Kullback–Leibler Divergence [18]. In contrast to both of these approaches, which involve evaluating the similarity between $\mathcal{G}$ and $\mathcal{C}$, StyleRank is optimized to evaluate the similarity of a single sample $g \in \mathcal{G}$ to $\mathcal{C}$.

[2] Note that we adapt the set-builder notation to construct a list (e.g., $[i/2 : 0 \leqslant i < 4] = [0, 0, 1, 1]$), which unlike a set, may contain duplicate values.

4. FEATURES

Although the features extracted by jSymbolic2 [23] are quite comprehensive, many features are high-level, and thus, ill-suited for the fine-grained distinctions that are necessary to rank stylistically similar MIDI files. For example, the `Chord Type Histogram` feature contains only 11 categories. In order to capture the complexity of the musical material being analyzed, we extract a variety of high-dimensional categorical distributions from a single MIDI file. A categorical distribution is a discrete probability distribution describing a random variable that has k possible distinct states. In what follows we adopt the following notation. Given a set x, $||x||$ denotes the number of elements in the set x, $\min(x)$ and $\max(x)$ denote the minimum and maximum element in x respectively, and x_i denotes the i^{th} element in x. $x \backslash y$ is the set difference between x and y, and $x \times y$ is the Cartesian product of x and y. $\ll$ indicates a left bitwise shift and $\gg$ indicates a right bitwise shift. $\&$, $\vee$, and $|$ refer to the bitwise `AND`, `XOR`, and `OR` operations, respectively.

4.1 Pitch Class Set Representations

In order to reduce the number of chords, we discard octave information and represent chords as pitch class sets, using a 12-bit integer to denote the presence or absence of a particular pitch class ($\text{C} = 0, \text{C\#} = 1, ..., \text{B} = 11$). For example, the C-major chord $\{60, 64, 67\}$ corresponds to the pitch class set $x = \{0, 4, 7\}$, which corresponds to the integer $\sum_{i=1}^{||x||}(1 \ll x_i) = 2^0 + 2^4 + 2^7 = 145$. Since there are 12 pitch classes, there are $2^{12} = 4096$ pitch class sets, which greatly reduces the possible number of chords. However, it is possible to further reduce this space if we create an equivalence class for all transpositionally equivalent pitch class sets. For example, the pitch class sets $\{0, 4, 7\}$ and $\{2, 5, 10\}$ are transpositionally equivalent, as both are major chords, the only difference being their root. This results in 352 distinct pitch class sets (PCD). Using Eq. (1c) a PCD can be calculated, where x is an 12-bit integer. Notably, pitch class sets are considered equivalent under the reversal operation when calculating the Forte number of a pitch class set [15]. Consequently, the pitch class sets $\{0, 4, 7\}$ and $\{0, 3, 7\}$ have the same Forte number, but correspond to different PCD's.

$$\texttt{rot}(x, n, i) = (x \ll i) \mid (x \gg (n-i)) \;\&\; (2^n - 1) \quad \text{(1a)}$$

$$\texttt{reduce}(x, n) = \min(\{\texttt{rot}(x, n, i) : 0 \leqslant i < n\}) \quad \text{(1b)}$$

$$\texttt{pcd}(x) = \texttt{reduce}(x, 12) \quad \text{(1c)}$$

Alternatively, a pitch class set x can be represented as the set of scales which are supersets of x. Given a scale $\mathbb{S}$, let $\mathbb{S}_i = \{(s + i) \mod 12 : s \in \mathbb{S}\}$. The scale representation can be calculated with Eq. (2), where

$\mathbb{S}^{\text{M}} = \{0,2,4,5,7,9,11\}$ and $\mathbb{S}^{\text{H}} = \{0,2,3,5,7,8,11\}$ denote the major and harmonic minor scales respectively. $\phi(\cdot)$ returns 1 if the predicate $\cdot$ is true and 0 otherwise.

$$\texttt{sc}(x) = \Big(\sum_{i=1}^{12} \phi(x \subseteq \mathbb{S}_i^{\text{M}}) \ll i\Big) + \Big(\sum_{i=1}^{12} \phi(x \subseteq \mathbb{S}_i^{\text{H}}) \ll (12+i)\Big) \quad (2)$$

4.2 Feature Definitions

Given a MIDI file M, for each note $n \in M$, $\texttt{ons}(n)$ returns the onset time of n in ticks, $\texttt{dur}(n)$ returns the duration of n in ticks, and $\texttt{pitch}(n)$ returns the pitch. An ordered set containing the unique onsets $O = \{\texttt{ons}(n) : n \in M\}$ is constructed, and the i^{th} chord is the set of notes $\mathbb{C}^i = \{n : (\texttt{ons}(n) \leqslant O_i) \wedge (\texttt{ons}(n) + \texttt{dur}(n) > O_i)\}$. $\texttt{isOns}(\mathbb{C}, n)$ and $\texttt{isTie}(\mathbb{C}, n)$ are functions that return 1 if n is an onset or a tie respectively, and 0 otherwise. The function $\texttt{pc}_i(\mathbb{C}, n)$ returns 1 if n corresponds to the pitch class i and 0 otherwise. In order to simplify the feature definitions, we use Eq. (3d), which accepts a chord $\mathbb{C}$ and a set of functions F, and only returns 1 if there is an element in X for which each $f \in F$ evaluates to 1. As a result, $\mathbf{I}\big(\mathbb{C}, \{\texttt{isOns}, \texttt{pc}_i\}\big)$ is 1 if there is a note $n \in \mathbb{C}$ that is an onset and is equivalent to the pitch class i.

$$\texttt{pc}_i(\mathbb{C},n) = \begin{cases} 0, & \text{if } \texttt{pitch}(n) \mod 12 \equiv i \\ 1, & \text{otherwise} \end{cases} \quad (3a)$$

$$\texttt{isOns}(\mathbb{C},n) = \begin{cases} 0, & \text{if } \max(\{\texttt{ons}(n) : n \in \mathbb{C}\}) > \texttt{ons}(n) \\ 1, & \text{otherwise} \end{cases} \quad (3b)$$

$$\texttt{isTie}(\mathbb{C},n) = 1 - \texttt{isOns}(\mathbb{C},n) \quad (3c)$$

$$\mathbf{I}(\mathbb{C},F) = \begin{cases} 0, & \text{if } \max\big(\{\prod_{i=1}^{||F||} F_i(\mathbb{C},n) : n \in \mathbb{C}\}\big) < 1 \\ 1, & \text{otherwise} \end{cases} \quad (3d)$$

Table 1 provides formal definitions of all the features, where $\mathbb{C}^t$ denotes the t^{th} chord, $\mathbb{M}^t$ denotes the t^{th} melody pitch, $\mathbb{P}^t = \{\texttt{pitch}(n) : n \in \mathbb{C}^t\}$, $\mathbb{O}^t = \{\texttt{ons}(n) : n \in \mathbb{C}^t\}$, and $\mathbb{K}^t = \{\texttt{pitch}(n) : (n \in \mathbb{C}^t) \wedge \texttt{isOns}(n)\}$. $\texttt{popcount}(\cdot)$ is a function that counts the number of set bits in an integer, $\texttt{pc}(x) = x \mod 12$ and $\texttt{pcc}(x) = |(x \mod 12) - 6|$. Dissonance is calculated using Stolzenburg's periodicity function [35], which we refer to as $\texttt{stol}(\cdot)$. Let $\texttt{diss}(\mathbb{P}, \mathbb{T}) = \frac{1}{||\mathbb{T}||} \sum_{x \in \mathbb{T}} \texttt{stol}(\bar{\mathbb{P}}^x)$, where $\mathbb{P}$ and $\mathbb{T}$ are pitch sets, and $\bar{\mathbb{P}}^x = \{\mathbb{P}_i - x : \mathbb{P}_i \in \mathbb{P}\}$. $\texttt{voiceMotion}(\cdot)$ is a function that accepts two successive pitch sets $(\mathbb{P}^t, \mathbb{P}^{t+1})$ and returns an integer corresponding to the type of voice motion. $\texttt{tonnetzLength}(\cdot)$ is a function that accepts a pitch class set and returns the length of the shortest path through Tonnetz [24] vertices containing each pitch class.

Each function is calculated for all valid values of t, resulting in a categorical distribution with unsigned 64-bit integers as the categories. For example, given a standard 4-voice Bach chorale containing m chords, the function $\texttt{ChordSize}$ is calculated for $0 \leqslant t < m - 2$, producing a categorical distribution with the categories $\{0,1,2,3,4\}$. In some cases, we weight values by chord duration, denoted by a $\star$ in the table. In the case that a function returns a set of values ($\texttt{IntervalDist}$), we combine the returned sets to form the categorical distribution. Since the number of categories k grows exponentially large for some features (e.g., $\texttt{ChordShape}$), we restrict $k \leqslant 1000$ by ranking categories according to the number of samples they appear in, removing infrequently occurring categories.

4.3 Implementation

We implement the feature extraction tool in C++, using pybind11 [17] to create Python bindings. The Midifile library [3] is used to parse MIDI files.

5. SIMILARITY COMPUTATION

In the most general sense, we are interested in measuring the similarity between a single MIDI file $\mathcal{X}$ and a corpus $\mathcal{C} = \{\mathcal{C}_1, ..., \mathcal{C}_n\}$. We represent each MIDI file by applying a non-empty set of feature transformations $\mathcal{F} = \{f_1, ..., f_k\}$, producing a set of categorical distributions for each MIDI file. For each $f_i \in \mathcal{F}$, we aim to measure the similarity between a single categorical distribution $f_i(\mathcal{X})$ and a set of categorical distributions $f_i(\mathcal{C}) = \{f_i(\mathcal{C}_1), ..., f_i(\mathcal{C}_n)\}$. Using a distance metric $\mathscr{D}$, the average similarity could be calculated $\frac{1}{n}\sum_{i=1}^{n} 1 - \mathscr{D}(f_i(\mathcal{C}_i), f_i(\mathcal{X}))$. However, this approach does not leverage information about the discriminative aspects of the entire corpus. The results in Experiment 1 demonstrate the deficiencies of this approach. Instead, we use Random Forests [7] to construct an embedding space before measuring the average similarity. Although neural networks are often ideal for learning embeddings, the time required to train k neural networks is prohibitive for an online system.

Decision trees are commonly used to model complex data. When used to classify data, each terminal node represents a discrete class label, and an arbitrary input is classified based on the terminal node it reaches. Using a trained Random Forest, an input can be represented based on the terminal node it reaches in each decision tree. Given a Random Forest containing N decision trees each with L terminal nodes, an input can be represented as a vector $v \in \{0,1\}^{N \times L}$. To learn an embedding for a single feature transformation $f_i \in \mathcal{F}$, we train a Random Forest to discriminate between a collection of items $f_i(\mathcal{G}) = \{f_i(\mathcal{G}_1), .., f_i(\mathcal{G}_m)\}$ and a corpus $f_i(\mathcal{C}) = \{f_i(\mathcal{C}_1), ..., f_i(\mathcal{C}_n)\}$. Concretely, each $f_i(\mathcal{G}_i) \in f_i(\mathcal{G})$ is given the label 0, and each $f_i(\mathcal{C}_i) \in f_i(\mathcal{C})$ is given the label 1. We refer to the vector produced for a sample $\mathcal{X}$ as $\mathbf{R}_{\mathcal{X}}^{\mathcal{G},\mathcal{C},f_i}$. Breiman measures the similarity of two vectors using the dot product [7]. In order to weight each feature transformation ($f_i \in \mathcal{F}$) equally, we use cosine similarity (Eq. (4a)), which is simply the normalized dot product. The similarity between $\mathcal{X}$ and $\mathcal{C}$ with respect to a set of

[3] https://midifile.sapp.org/

	Feature Name	Function	Description
Chord	ChordDissonance ⋆	$\lfloor \texttt{diss}(\mathbb{K}^t, \mathbb{K}^t) \rfloor$	the dissonance of onsets based on periodicity [35]
	ChordDistinctDurationRatio	$\big(1 \ll ||\{\texttt{dur}(n) : n \in \mathbb{C}^t\}||\big) \mid 2^{||\mathbb{C}^t||}$	the ratio of distinct note durations to chord size
	ChordDuration	$\max(\mathbb{O}^{t+1}) - \max(\mathbb{O}^t)$	the duration of a chord
	ChordLowestInterval	$\min\big(\mathbb{P}^t \backslash \{\min(\mathbb{P}^t)\}\big) - \min(\mathbb{P}^t)$	the difference between the lowest two notes
	ChordOnset	$\big(\sum_{i=1}^{||\mathbb{C}^t||}(\texttt{isOns}(\mathbb{C}_i^t) \ll (i-1))\big) \mid 2^{||\mathbb{C}^t||}$	an integer representing which notes are onsets
	ChordOnsetPCD ⋆	$\texttt{pcd}\big(\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^t, \{\texttt{isOns}, \texttt{pc}_i\}) \ll i)\big)$	distinct pitch class set excluding ties
	ChordOnsetRatio	$\big(1 \ll \sum_{n \in \mathbb{C}^t} \texttt{isOns}(n)\big) \mid 2^{||\mathbb{C}^t||}$	the ratio of onsets to chord size
	ChordOnsetShape ⋆	$\sum_{i=1}^{||\mathbb{C}^t||}(\texttt{isOns}(\mathbb{C}^t, \mathbb{C}_i^t) \ll (\mathbb{P}_i^t - \min(\mathbb{P}^t)))$	piano roll type representation of onset pitches
	ChordOnsetTiePCD ⋆	$\texttt{pcd}\big(\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^t, \{\texttt{isOns}, \texttt{pc}_i\}) \ll i)\big) + \texttt{pcd}\big(\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^t, \{\texttt{isTie}, \texttt{pc}_i\}) \ll i)\big) \ll 12$	concatenated distinct pitch class set of onsets and distinct pitch class set of ties
	ChordOnsetTieReduced ⋆	$\texttt{reduce}\big(\big(\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^t, \{\texttt{isOns}, \texttt{pc}_i\}) \ll i)\big) + \big(\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^t, \{\texttt{isTie}, \texttt{pc}_i\}) \ll (12+i))\big)\big)$	concatenated pitch class set of onsets and pitch class set of ties reduced using Eq. (1b)
	ChordPCD ⋆	$\texttt{pcd}\big(\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^t, \{\texttt{pc}_i\}) \ll i)\big)$	distinct pitch class set
	ChordPCDWBass ⋆	$\texttt{pcd}\big(\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^t, \{\texttt{pc}_i\}) \ll i)\big) + 2^{12+\texttt{pc}(\min(\mathbb{P}^t))}$	distinct pitch class set with bass pitch class
	ChordPCSizeRatio	$\big(1 \ll ||\{\texttt{pc}(p) : p \in \mathbb{P}^t\}||\big) \mid 2^{||\mathbb{P}^t||}$	the ratio of distinct pitch classes to chord size
	ChordRange (ϕ_1)	$\max(\mathbb{P}^t) - \min(\mathbb{P}^t)$	the range of pitches in a chord
	ChordShape ⋆	$\sum_{p \in \mathbb{P}^t}(1 \ll (p - \min(\mathbb{P}^t)))$	piano roll type representation of chord pitches
	ChordSize	$||\mathbb{C}^t||$	the number of notes in a chord
	ChordTonnetz ⋆	$\texttt{tonnetzLength}(\{\texttt{pc}(x) : x \in \mathbb{P}^t\})$	length of shortest path through Tonnetz [24] vertices
Chord Transition	ChordSizeNgram	$||\mathbb{C}^t|| + (||\mathbb{C}^{t+1}|| \ll 8) + (||\mathbb{C}^{t+2}|| \ll 16)$	an n-gram of chord sizes ($n = 3$)
	ChordTranBassInterval	$\texttt{pc}(\min(\mathbb{P}^{t+1}) - \min(\mathbb{P}^t))$	pitch class interval between two lowest notes
	ChordTranDissonance	$\lfloor \texttt{diss}(\mathbb{P}^t, \mathbb{P}^{t+1}) \rfloor$	the dissonance of intervals based on periodicity [35]
	ChordTranDistance	$\lvert \min(\mathbb{P}^{t+1}) - \min(\mathbb{P}^t) \rvert + \lvert \max(\mathbb{P}^{t+1}) - \max(\mathbb{P}^t) \rvert$	approximated voice leading distance
	ChordTranOuter	$\texttt{pc}(\phi_1(\mathbb{P}^t)) + (\texttt{pc}(\phi_1(\mathbb{P}^{t+1})) \ll 8) + (\texttt{pc}(\min(\mathbb{P}^t) - \min(\mathbb{P}^{t+1})) \ll 16)$	pitch class transition using only the outer notes
	ChordTranPCD	$\texttt{reduce}\big(\big(\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^t, \{\texttt{pc}_i\}) \ll i)\big) + \big(\sum_{i=0}^{11}(\mathbf{I}(\mathbb{C}^{t+1}, \{\texttt{pc}_i\}) \ll (12+i))\big), 24\big)$	transition between distinct pitch class sets
	ChordTranRepeat	$(\prod_{n \in \mathbb{C}^t} \texttt{isOns}(n))(\mathbb{P}^t = \mathbb{P}^{t+1})$	chord repetition with onsets
	ChordTranScaleDistance	$\texttt{popcount}\big(\texttt{sc}(\mathbb{P}^t) \vee \texttt{sc}(\mathbb{P}^{t+1})\big)$	hamming distance between scale representations
	ChordTranScaleUnion	$\texttt{popcount}\big(\texttt{sc}(\mathbb{P}^t) \mid \texttt{sc}(\mathbb{P}^{t+1})\big)$	the union between scale representations
	ChordTranVoiceMotion	$\texttt{voiceMotion}(\mathbb{P}^t, \mathbb{P}^{t+1})$	type of voice motion (contrary, oblique, etc.)
Mel.	MelodyNgram	$\sum_{i=0}^{3}(\mathbb{M}_{t+i+1} - \mathbb{M}_{t+i} \mod 12) \ll 8i$	n-gram of melodic intervals ($n = 3$)
	MelodyPCD	$\texttt{pcd}\big(\sum_{i=0}^{11}\mathbf{I}(\{\mathbb{M}^{t+i} : 0 \leqslant i < 5\}, \{\texttt{pc}_i\}) \ll i\big)$	distinct pitch class of successive melody notes
Inter.	IntervalClassDist	$\{\texttt{pcc}(p_i - p_j) : (p_j < p_i) \wedge (p_i, p_j \in \mathbb{P}^t \times \mathbb{P}^t)\}$	interval class for each combination of chord pitches
	IntervalDist	$\{\texttt{pc}(p_i - p_j) : (p_j < p_i) \wedge (p_i, p_j \in \mathbb{P}^t \times \mathbb{P}^t)\}$	interval for each combination of chord pitches

Table 1. Definitions for Chord features, Chord Transition features, Melody features (Mel.), and Interval features (Inter.). The ⋆ symbol indicates that a categorical distribution is weighted by chord duration.

features $\mathcal{F}$ is computed using Eq. (4b), which produces a scalar value on the range $[0, 1]$.

$$\texttt{cos}(X, Y) = \frac{X \cdot Y}{\sqrt{\sum_{i=1}^{N} X_i^2}\sqrt{\sum_{i=1}^{N} Y_i^2}} \tag{4a}$$

$$S_{\mathcal{X}}^{\mathcal{G},\mathcal{C},\mathcal{F}} = \frac{1}{||\mathcal{C}||||\mathcal{F}||}\sum_{c \in \mathcal{C}}\sum_{f \in \mathcal{F}} \texttt{cos}(\mathbf{R}_{\mathcal{X}}^{\mathcal{G},\mathcal{C},f}, \mathbf{R}_{c}^{\mathcal{G},\mathcal{C},f}) \tag{4b}$$

6. EXPERIMENTS

In the following experiments, we train a Random Forest [7] using the scikit-learn python module [30]. We set the maximum tree depth at 5, the number of trees to 500, and measure the quality of the split using entropy. The class weight is balanced to be robust against size discrepancies between $\mathcal{C}$ and $\mathcal{G}$.

6.1 Experiment 1 : Analytic Testing

We test StyleRank with styles delineated by a single composer, and by an entire genre, using the Classical Archives MIDI dataset [4] . In total there are 75 composers, and 6 musical genres. More details on the composition of the dataset can be found in the Appendix [5] . We keep only one MIDI file per composition. Each MIDI file is represented as a list of pitches, sorted lexicographically according to onset and pitch. To compare two pieces, the Levenshtein distance [19] is measured twice, once for the first 100 pitches in each piece, and once for the last 100 pitches. We eliminate pieces which have a Levenshtein distance less than 0.75, after normalizing the distance on the range $[0, 1]$. We choose this conservative value to ensure all duplicates are removed.

Given two styles $A = \{a_1, ..., a_m\}$ and $B = \{b_1, ..., b_n\}$, where $m = 2n$, let $\mathcal{C} = \{a_i : 1 \leqslant i \leqslant n\}$,

[4] https://www.classicalarchives.com/midi.html

[5] https://github.com/jeffreyjohnens/style_rank/tree/master/appendix

$\mathcal{G}_A = \{a_i : n < i \leqslant 2n\}$, $\mathcal{G}_B = B$, and $\mathcal{G} = \mathcal{G}_A \cup \mathcal{G}_B$. By construction $\mathcal{G} \cap \mathcal{C} = \varnothing$. We train a Random Forest and compare two distributions $x = [S_g^{\mathcal{G},\mathcal{C},\mathcal{F}} : g \in \mathcal{G}_A]$ and $y = [S_g^{\mathcal{G},\mathcal{C},\mathcal{F}} : g \in \mathcal{G}_B]$, where $\mathcal{F}$ denotes the set of features described in Table 1. Ideally, each value in x should be larger than all values in y, since elements in $\mathcal{G}_A$ and $\mathcal{C}$ belong to the same style (A). However, depending on the specificity of the style, there may be some degree of overlap between A and B. In order to determine if there is a measurable difference between x and y we directly compare the means $(\bar{x} > \bar{y})$, and we calculate the p-value $(p^{\bar{x}>\bar{y}})$ for a One-Sided Mann-Whitney test [22] with the alternative hypothesis that $\bar{x} > \bar{y}$.

In cases where multiple statistical comparisons are performed, it is common practice to apply a correction to the raw p-values. The Bonferroni correction [11] is calculated by dividing the desired level of significance $(\alpha = 0.05)$ by the number of comparisons. The Benjamini–Yekutieli procedure [4] controls the false discovery rate under arbitrary dependence assumptions, and is less conservative than the Bonferroni correction. Given m null hypotheses and their corresponding p-values $P_1, ..., P_m$, the p-values are sorted in ascending order. For a given level of significance, in our case $\alpha = 0.05$, reject the null hypothesis for the first k values that satisfy $P_k \leqslant k\alpha/(m * c(m))$ where $c(m) = \sum_{i=1}^{m} 1/i$.

Table 2 shows the results of 1000 trials, reporting the percentage of trials where $\bar{x} > \bar{y}$, and the percentage of trials where $p^{\bar{x}>\bar{y}}$ is significant, applying no correction $(\alpha = 0.05)$, the Benjamini–Yekutieli procedure (FDR), and the Bonferroni correction (Bon). We compare StyleRank against three distance measures, Cosine, Manhattan and Euclidean, replacing $S_g^{\mathcal{G},\mathcal{C},\mathcal{F}}$ with $\frac{1}{||\mathcal{C}||||\mathcal{F}||} \sum_{c\in\mathcal{C}} \sum_{f\in\mathcal{F}} 1 - \mathscr{D}(f(c), f(g))$.

6.2 Experiment 2: Congruity with Human Perception

In order to evaluate how well StyleRank correlates with human perception, we use data from the BachBot [20] experiment. In total, there were 5,967 participants, including 1329 novices, 2786 intermediate, 1341 advanced and 511 experts. Liang et al. generated 36 samples $(\mathcal{G})$ from a neural network trained on a collection of Bach Chorales $(\mathcal{C})$. Participants were asked to discriminate between a generated musical excerpt and an actual Bach chorale. They were each asked to complete 5 comparisons.

For each $g \in \mathcal{G}$, we count the number of times it was mistakenly classified as a Bach chorale N_g^{miss}, and the number of times it was correctly identified as computer generated N_g^{corr}. The raw count data can be found in the Appendix. We take the relative frequency of missclassifications $T_g = N_g^{\text{miss}}/(N_g^{\text{miss}} + N_g^{\text{corr}})$ as an indication of how similar g is to the style of Bach's Chorales $(\mathcal{C})$. This results in $\binom{36}{2} = 630$ pairwise comparisons for which we have a ground truth ranking. Using a chi-square contingency test [29] we can measure the degree to which we are certain that there is a difference between two samples. We measure accuracy using Eq. (5b), where p_{ij} is the p-value for the chi-square contingency test comparing the counts for the i^{th} and j^{th} examples, $\phi(\cdot)$ is a function returning 1 if the predicate $\cdot$ is true and 0 otherwise, and α denotes the threshold for significance.

$$f(x,y) = \begin{cases} 1, & \text{if } \phi\big(S_x^{\mathcal{G},\mathcal{C},\mathcal{F}} < S_y^{\mathcal{G},\mathcal{C},\mathcal{F}}\big) = \phi\big(T_x < T_y\big) \\ 0, & \text{otherwise} \end{cases} \tag{5a}$$

$$\texttt{acc}(\mathcal{G},\mathcal{C},\alpha) = \frac{\sum_{i=1}^{||\mathcal{G}||} \sum_{j=i+1}^{||\mathcal{G}||} f(\mathcal{G}_i,\mathcal{G}_j)\phi(p_{ij} < \alpha)}{\sum_{i=1}^{||\mathcal{G}||} \sum_{j=i+1}^{||\mathcal{G}||} \phi(p_{ij} < \alpha)} \tag{5b}$$

The results for Experiment 2 are presented in Table 3. We report the accuracy, calculated using Eq. (5b), for a random ranking (Random), StyleRank with the jSymbolic [23] features (jSymbolic), Log-likelihood (Loglik), and StyleRank. All the default features are extracted using jSymbolic, and features with zero standard deviation are removed. This results in a single feature vector with dimension of 453, for which we train a single Random Forest. Using the Performance RNN [12], which was trained with the same representation and data as the original BachBot, we evaluate the negative log-likelihood $\mathscr{L}_g$ of each of the generated examples (loglik). To calculate the accuracy we simply replace the term $S_X^{\mathcal{G},\mathcal{C},\mathcal{F}} < S_Y^{\mathcal{G},\mathcal{C},\mathcal{F}}$ with $\mathscr{L}_X < \mathscr{L}_Y$ in Eq. (5a).

7. DISCUSSION

Collectively, the results of both experiments demonstrate that StyleRank is robust to corpora of varying sizes, and highly correlated with human perception of stylistic similarity. In the Appendix, we expand Experiment 1 to demonstrate that StyleRank's performance is robust, even when the number of distinct styles in $\mathcal{G}$ is increased. In Experiment 1, there is a large difference between raw distance measures and StyleRank. This highlights the limitations of the approach described by Yang and Lerch, which uses euclidean distance to measure the distance between feature vectors [44]. Although euclidean distance works well in low-dimensional settings, it does not scale well to high dimensions. In fact, it has been shown that Manhattan distance performs better than Euclidean distance in high dimensional settings [1], which we also see in our own experimental results. Understandably, there is a decrease in performance when analyzing styles delineated by genre, as these styles have more variance, and are less consistent than the work of a single composer. Overall, these results demonstrate that StyleRank can proficiently rank MIDI files with different styles.

The results for Experiment 2 demonstrate that StyleRank is capable of making fine-grained distinctions between MIDI files that correspond with human perception of stylistic similarity. It is worth noting that participants found it difficult to discriminate between generated and human-composed samples in the BachBot experiment, evidenced by the average classification accuracy of novice (0.57), intermediate (0.64), advanced (0.68), and expert

		StyleRank				Cosine				Manhattan				Euclidean			
	size	μ	Sig	FDR	Bon	μ	Sig	FDR	Bon	μ	Sig	FDR	Bon	μ	Sig	FDR	Bon
Composer	10	0.963	0.86	0.725	0.0	0.837	0.624	0.381	0.0	0.879	0.662	0.413	0.0	0.827	0.565	0.28	0.0
	25	0.951	0.888	0.807	0.609	0.808	0.583	0.422	0.24	0.793	0.578	0.415	0.244	0.729	0.532	0.363	0.226
	50	0.926	0.905	0.873	0.78	0.705	0.559	0.454	0.333	0.751	0.599	0.468	0.34	0.717	0.565	0.428	0.3
	100	1.0	0.986	0.973	0.951	0.713	0.636	0.59	0.515	0.723	0.633	0.568	0.486	0.715	0.626	0.571	0.504
Genre	10	0.81	0.379	0.0	0.0	0.68	0.193	0.0	0.0	0.686	0.2	0.0	0.0	0.645	0.176	0.0	0.0
	25	0.867	0.578	0.376	0.198	0.729	0.348	0.084	0.038	0.74	0.374	0.053	0.021	0.691	0.298	0.06	0.022
	50	0.88	0.715	0.59	0.432	0.776	0.484	0.266	0.126	0.747	0.489	0.253	0.088	0.714	0.344	0.158	0.082
	100	0.927	0.847	0.774	0.671	0.766	0.555	0.406	0.265	0.755	0.566	0.44	0.284	0.785	0.462	0.269	0.178

Table 2. The normalized frequency over 1000 trials where $\bar{x} > \bar{y}$ (μ), $p^{\bar{x}>\bar{y}} < 0.05$ (Sig), $p^{\bar{x}>\bar{y}}$ is significant after applying the FDR correction (FDR), and $p^{\bar{x}>\bar{y}}$ is significant after applying the Bonferonni correction (Bon). Size denotes the size of the corpus $||\mathcal{C}|| = ||\mathcal{G}_A|| = ||\mathcal{G}_B||$.

	Novice				Intermediate			
	$\alpha = 5.0$	$\alpha = 0.5$	$\alpha = 0.05$	$\alpha = 0.005$	$\alpha = 5.0$	$\alpha = 0.5$	$\alpha = 0.05$	$\alpha = 0.005$
Random	.482 ± .025	.479 ± .031	.466 ± .044	.440 ± .062	.500 ± .023	.500 ± .026	.502 ± .033	.499 ± .037
jSymbolic	.471 ± .006	.463 ± .008	.472 ± .012	.491 ± .015	.478 ± .011	.474 ± .013	.467 ± .014	.456 ± .017
Loglik	.629 ± .000	.669 ± .000	.764 ± .000	.817 ± .000	.654 ± .000	.668 ± .000	.690 ± .000	.732 ± .000
StyleRank	.716 ± .001	.774 ± .002	.855 ± .004	.899 ± .005	.702 ± .002	.715 ± .002	.758 ± .002	.808 ± .002
	Advanced				Expert			
Random	.511 ± .010	.514 ± .013	.512 ± .017	.515 ± .019	.493 ± .019	.492 ± .025	.492 ± .032	.485 ± .038
jSymbolic	.481 ± .011	.480 ± .011	.470 ± .014	.474 ± .013	.452 ± .008	.449 ± .009	.482 ± .012	.464 ± .013
Loglik	.673 ± .000	.694 ± .000	.730 ± .000	.724 ± .000	.657 ± .000	.692 ± .000	.741 ± .000	.800 ± .000
StyleRank	.718 ± .001	.756 ± .001	.806 ± .002	.808 ± .002	.692 ± .002	.745 ± .003	.821 ± .004	.881 ± .005

Table 3. The accuracy of each model, calculated using Eq. (5b), with standard error calculated over 10 trials.

(0.71) participants [20]. Based on our experimental results, the jSymbolic [23] feature set is no better at predicting rankings than a random model. This is likely due to the fact that high level features are not sufficiently discriminative for this task. In contrast to the jSymbolic feature set, our method involves full categorical distributions, which we believe are critical in measuring fine-grained differences. Importantly, there is a substantial difference between the accuracy of rankings based on log-likelihood and StyleRank. Interestingly, both log-likelihood and StyleRank best model high certainty ($\alpha = 0.005$) comparisons made by self identified novices. This may be an artifact of increased variance as the number of ground truth comparisons decreases as α increases.

It should be noted that participants in the BachBot experiment were not directly asked to rank samples according to their similarity to the style of Bach's chorales. We extrapolated a ranking from the number of times a sample was miss-classified, which is an indirect way of measuring stylistic similarity. However, since these rankings were based on a large sample size, we are confident that they are reflective of human perception.

8. APPLICATION

StyleRank can be used in a variety of settings. Importantly, we must note that there are no limitations on the composition of $\mathcal{G}$. For example, one could compare k different sets with $\mathcal{G} = \{\mathcal{G}_i^1, ..., \mathcal{G}_{n_1}^1, \mathcal{G}_1^2, ..., \mathcal{G}_{n_2}^2, ..., \mathcal{G}_1^k, ..., \mathcal{G}_{n_k}^k\}$. First of all, the method can be use to rank samples generated by an SI system, based on their similarity to $\mathcal{C}$. StyleRank can be used to filter highly dissimilar samples automatically. Filtering is as simple as taking the samples $g \in \mathcal{G}$ with a similarity $S_g^{\mathcal{G},\mathcal{C},\mathcal{F}}$ above some threshold, and discarding the rest. Secondly, StyleRank can be used to rank models. Given k models, let $\mathcal{G} = \{\mathcal{G}^1, ..., \mathcal{G}^k\} = \{\mathcal{G}_i^1, ..., \mathcal{G}_{n_1}^1, ..., \mathcal{G}_1^k, ..., \mathcal{G}_{n_k}^k\}$, where $\mathcal{G}^i$ denotes the set of samples generated by the i^{th} model. Then the distributions $x_i = [S_g^{\mathcal{G},\mathcal{C},\mathcal{F}} : g \in \mathcal{G}^i]$ can be compared using an appropriate statistical test. Third, the method can be used to isolate the specific features f that deviate from the style delineated by $\mathcal{C}$ by comparing the distributions $x_f = [S_g^{\mathcal{G},\mathcal{C},f} : g \in \mathcal{G}]$ for each f in a set of features $\mathcal{F}$. In addition, StyleRank can be used as a tool for musicologists to explore variations in style.

9. CONCLUSION

Quantifying musical stylistic similarity is a difficult task. We propose StyleRank, a method to rank individual MIDI files based on their similarity to an arbitrary style. Experimental evidence supports our approach, demonstrating that our method is robust, and is highly correlated with human perception of stylistic similarity. Future work involves applying this approach to other domains where SI systems are being developed. Additional features can be added to the current collection, in particular rhythm-based features, as the current collection is pitch-centric. Although we believe our experiments to be fairly comprehensive, continued validation of the proposed method on additional data is always beneficial.

10. ACKNOWLEDGMENTS

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Helmut Hugo Eppich Family Graduate Scholarship.

11. REFERENCES

[1] C.C. Aggarwal, A. Hinneburg, and D.A. Keim. On the surprising behavior of distance metrics in high dimensional space. In *International Conference on Database Theory*, pages 420–434, 2001.

[2] T.E. Ahonen, K. Lemström, and S. Linkola. Compression-based similarity measures in symbolic, polyphonic music. In *ISMIR*, pages 91–96, 2011.

[3] T. M. Amabile. Social psychology of creativity: A consensual assessment technique. *Journal of Personality and Social Psychology*, 43(5):997–1013, 1982.

[4] Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, 29(4):1165–1188, 2001.

[5] M. Besson, D. Schön, S. Moreno, A. Santos, and C. Magne. Influence of musical expertise and musical training on pitch processing in music and language. *Restorative Neurology and Neuroscience*, 25(3-4):399–410, 2007.

[6] P. Boot, A. Volk, and W.B. de Haas. Evaluating the role of repeated patterns in folk song classification and compression. *Journal of New Music Research*, 45(3):223–238, 2016.

[7] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[8] R.B. Dannenberg. *Style in Music*, pages 45–57. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[9] W.B. de Haas, F. Wiering, and R.C. Veltkamp. A geometrical distance measure for determining the similarity of musical harmony. *International Journal of Multimedia Information Retrieval*, 2(3):189–202, Sep 2013.

[10] H.W. Dong, W.Y. Hsiao, L.C. Yang, and Y.H. Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *32nd AAAI Conference on Artificial Intelligence*, pages 34–41, 2018.

[11] O.J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[12] D. Eck, A. Roberts, J. Engel, C. Hawthorne, and I. Simon. Magenta, 2019. https://github.com/tensorflow/magenta.

[13] T. Eerola, T. Himberg, P. Toiviainen, and J. Louhivuori. Perceived complexity of western and african folk melodies by western and african listeners. *Psychology of Music*, 34(3):337–371, 2006.

[14] J. Ens and P. Pasquier. Caemsi : A cross-domain analytic evaluation methodology for style imitation. In *International Conference on Computational Creativity*, pages 64–71, 2018.

[15] A. Forte. *The Structure of Atonal Music*. Yale [paperbacks]. Yale University Press, 1973.

[16] E.E. Hannon and S.E. Trehub. Metrical categories in infancy and adulthood. *Psychological Science*, 16(1):48–55, 2005.

[17] W. Jakob, J. Rhinelander, and D. Moldovan. pybind11 – seamless operability between c++11 and python, 2017. https://github.com/pybind/pybind11.

[18] S. Kullback and R.A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[19] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady*, 10(8):707–710, 1966.

[20] F. T. Liang, M. Gotham, M. Johnson, and J. Shotton. Automatic stylistic composition of bach chorales with deep lstm. In *Proc. of the International Symp. on Music Information Retrieval*, pages 449–456, 2017.

[21] N.H. Liu, Y.H. Wu, and A.L. Chen. Efficient knn search in polyphonic music databases using a lower bounding mechanism. *Multimedia systems*, 10(6):513–528, 2005.

[22] H.B. Mann and D.R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947.

[23] C. McKay, J. Cumming, and I. Fujinaga. jsymbolic 2.2: Extracting features from symbolic music for use in musicological and mir research. In *Proc. of the International Symp. on Music Information Retrieval*, 2018.

[24] A.V. Oettingen. *Harmoniesystem in dualer Entwickelung*. W. Glaser, Dorpat, 1866.

[25] J.F. Paiement, D. Eck, and S. Bengio. A probabilistic model for chord progressions. In *Proc. of the International Symp. on Music Information Retrieval*, pages 11–15, 2005.

[26] E. Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[27] P. Pasquier, A. Eigenfeldt, O. Bown, and S. Dubnov. An introduction to musical metacreation. *Computer Entertainment*, 14(2):3–17, 2017.

[28] M. T. Pearce and G. A. Wiggins. Evaluating cognitive models of musical composition. In *Proceedings of the 4th international joint workshop on computational creativity*, pages 73–80. Goldsmiths, University of London, 2007.

[29] K. Pearson. *On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is Such that it Can be Reasonably Supposed to have Arisen from Random Sampling*, pages 11–28. Springer New York, New York, NY, 1992.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[31] J. Pickens and T. Crawford. Harmonic models for polyphonic music retrieval. In *Proc. of the International Conference on Information and Knowledge Management*, pages 430–437, 2002.

[32] J.B. Prince, M.A. Schmuckler, and W.F. Thompson. The effect of task and pitch structure on pitch-time interactions in music. *Memory & Cognition*, 37(3):368–381, 2009.

[33] M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, pages 832–837, 1956.

[34] M. Schedl, A. Flexer, and J. Urbano. The neglected user in music information retrieval research. *Journal of Intelligent Information Systems*, 41(3):523–539, 2013.

[35] F. Stolzenburg. Harmony perception by periodicity detection. *Journal of Mathematics and Music*, 9(3):215–238, 2015.

[36] B. L. Sturm and O. Ben-Tal. Taking the models back to music practice: Evaluating generative transcription models built using deep learning. *Journal of Creative Music Systems*, 2(1):1–29, 2017.

[37] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations*, 2016. arXiv:1511.01844.

[38] G.T. Toussaint. A comparison of rhythmic similarity measures. In *Proc. of the International Symp. on Music Information Retrieval*, pages 242–245, 2004.

[39] N. Trieu and R. Keller. Jazzgan: Improvising with generative adversarial networks. In *6th International Workshop on Musical Metacreation*, 2018.

[40] A. M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

[41] V. Velardo, M. Vallati, and S. Jan. Symbolic melodic similarity: State of the art and future challenges. *Computer Music Journal*, 40(2):70–83, 2016.

[42] A. Volk, E. Chew, Elizabeth Hellmuth M., and C. Anagnostopoulou. Music similarity: Concepts, cognition and computation, 2016.

[43] A. Volk, W.B. de Haas, and P. Kranenburg. Towards modelling variation in music as foundation for similarity. In *Proceedings of the 12th International Conference on Music Perception and Cognition and the 8th Triennial Conference of the European Society for the Cognitive Sciences of Music*. School of Music Studies, Aristotle University of Thessaloniki, 2012.

[44] L.C. Yang and A. Lerch. On the evaluation of generative models in music. *Neural Computing and Applications*, 1:1–12, Nov 2018.

AUDIO QUERY-BASED MUSIC SOURCE SEPARATION

Jie Hwan Lee* **Hyeong-Seok Choi*** **Kyogu Lee**
Music and Audio Research Group,
Graduate School of Convergence Science and Technology,
Seoul National University
{wiswisbus, kekepa15, kglee}@snu.ac.kr

ABSTRACT

In recent years, music source separation has been one of the most intensively studied research areas in music information retrieval. Improvements in deep learning lead to a big progress in music source separation performance. However, most of the previous studies are restricted to separating a few limited number of sources, such as vocals, drums, bass, and other. In this study, we propose a network for audio query-based music source separation that can explicitly encode the source information from a query signal regardless of the number and/or kind of target signals. The proposed method consists of a Query-net and a Separator: given a query and a mixture, the Query-net encodes the query into the latent space, and the Separator estimates masks conditioned by the latent vector, which is then applied to the mixture for separation. The Separator can also generate masks using the latent vector from the training samples, allowing separation in the absence of a query. We evaluate our method on the MUSDB18 dataset, and experimental results show that the proposed method can separate multiple sources with a single network. In addition, through further investigation of the latent space we demonstrate that our method can generate continuous outputs via latent vector interpolation.

1. INTRODUCTION

Music source separation, isolating the signals of certain instruments from a mixture, has been intensively studied in recent years. Due to the improvements in deep learning techniques, various approaches using deep learning for music source separation have been introduced. However, most of the previous studies are mainly focused on improving music source separation performances, not the range of separable sources. To tackle this problem, a few studies have tried to separate the fixed number of sources of interest by conditioning one-hot label in the deep learning network [14, 15].

*these authors contributed equally

While being the most straight-forward approach, we argue that such an approach is not a proper way to deal with the *outliers* when the generic and broadly defined class labels are the only available data at hand [7, 11]. To understand this situation more concretely, let us consider the mismatched situations where the target source is classified into a certain generic class but still somewhat far from the general characteristics of that broadly defined generic class. For example, consider the situation where we desire to separate 'distorted singing voice' or 'acoustic guitar' sources. In these cases, we can imagine that the performance can be boosted if we were to have more fine-grained labels such as 'distorted singing voice' or 'acoustic guitar' rather than generic classes such as 'vocals' or 'guitar'. One of the simplest ad-hoc solutions, therefore, can be manually annotating such *outliers* based on the music instrument ontology and conditioning those new classes into the deep learning network. Unfortunately, manually annotating an audio signal has limitation in many aspects. First, labeling an audio itself is costly. Second, given the same audio samples, the number of samples per class is reduced, hence it is likely that the separation performance degrades. Third, such a method is not scalable to new outlier samples, and is thus limited.

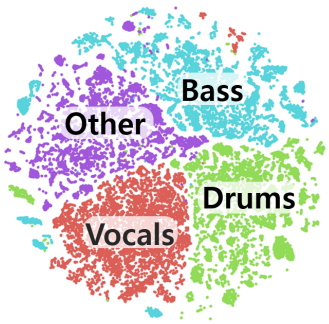


Figure 1. t-SNE visualization [8] of encoded latent vectors of the test dataset in MUSDB18. Without any classification loss, the Query-net is trained to output latent vectors that provide useful information about various instruments. It is observable that the latent vectors from the same class are clustered in the latent space while not being identical.

To deal with these problems, in this paper, a novel audio query-based music source separation framework is proposed. The main idea is to directly compress the diverse au-

dio samples into latent vectors – using the so-called *Query-net* – so that the audio samples can be mapped into non-identical points even when the samples are from the same class as illustrated in Fig. 1. The encoded latent vector is then fed into a separation network to output a source whose characteristics is similar to the audio sample taken into the Query-net. The proposed framework is scalable as the Query-net is able to encode an unseen singing voice or instrument sound into the continuous latent space. This property allows many useful utilities as follows. First, it is capable of separating various number of sources with a single network. Second, we can expect an increase in separation performance especially when the characteristics of the target source in the mixture is considered far from the given generic class since the user can manually select and encode the held-out sound sample that is deemed similar to the target signal. Third, it allows the natural control of the output of the separation network by interpolating the latent vectors in the continuous latent space.

To demonstrate the usefulness of the proposed method, we show various experiments using the MUSDB18 dataset [11]. The experiments show that the output of the separation network is highly dependent on the latent vector which allows smooth transition in signal level by controlling and interpolating the latent vectors. Also, we show that the proposed method becomes especially useful when the target source of interest is far from the general characteristic of coarsely defined sound class. Finally, we show that the proposed method can be even automated by iteratively encoding the separation output.

2. RELATED WORK

In this section, we first introduce previous music source separation studies that tried to separate mixture into multiple sound classes. One of the most basic ways is to estimate several separation masks with a single model. In [10], they tried to separate four sources with one stacked hourglass model [9]. While they showed a competitive results the method is not flexible as the model requires a fixed number of output. Next, [15] introduced a one-hot label conditioning approach and showed that their proposed method is capable of separating multiple sources. This method is more flexible than the aforementioned model but the model does not assume latent space, therefore, is not capable of manipulating output other than conditioning the one-hot label. Finally, [14] showed that they can embed each time-frequency bin of the mixture into a high-dimensional space using deep clustering [1] approach. However, this approach still has a limitation in that the model is not capable of encoding the audio signal directly into the latent space. Apart from the music source separation studies, [21] suggested a speaker-dependent speech separation method by incorporating a lstm-based anchor vector encoder which enables direct encoding of audio signal into a latent space. Using this technique, they showed that the proposed method can cluster the time-frequency bin embeddings that are close to the anchor vector in the latent space.

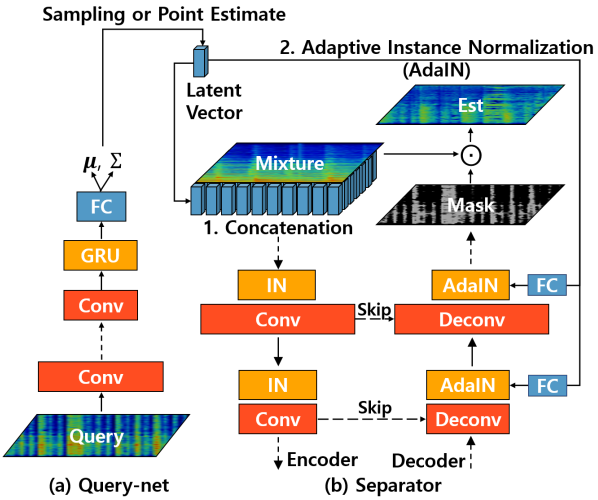


Figure 2. Illustration of the (a) Query-net and (b) Separator. The Query-net encodes the query into the latent vector and it is passed into the Separator by two methods. 1. Concatenation: The latent vector is concatenated with mixture spectrogram by tiling the latent vector along the spatial dimension. 2. AdaIN: Adaptive instance normalization is used in every layer of decoder part.

3. PROPOSED METHOD

3.1 Query-based Source Separation

The proposed framework is composed of two deep learning networks, Query-net $\mathbb{Q}(\cdot)$ and Separator $\mathbb{S}(\cdot)$. While most of the previous studies typically use $\mathbb{S}$ to extract a single class source from a mixture, we aim to separate the mixture by manipulating the additional input signal, a query. By doing so, we can expect to have a control over the mixture just by choosing a different query input which can be done either manually by the user or automatically by the system. Hence, the query signal is expected to be sampled from a similar sound class to the target signal within the mixture, but does not have to be identical. To achieve this, $\mathbb{Q}$ directly encodes the query audio signal into a latent vector so that we can control the output of $\mathbb{S}$ by manipulating the latent space.

$\mathbb{Q}$ is composed of 6 strided-convolutional layers followed by gated recurrent unit (GRU) layer. The stack of strided-convolutional layers are used to extract local features from the given query signal. Then, the extracted features are reshaped by stacking each feature map along the frequency axis. Finally, the reshaped tensor is passed into GRU and the last state of the GRU is used as a summary of the query signal. As we would like the encoded latent vector to have a meaningful high-level information, we designed $\mathbb{Q}$ to map the query into a small enough dimension compared to the dimension of the query signal. After the audio query has been encoded, the summarized information is passed into $\mathbb{S}$.

$\mathbb{S}$ is a U-Net [13] based network which has proven its effectiveness in many source separation studies [3, 10, 16, 18, 19]. It is a convolutional encoder and decoder with skip-

connections between the layers. $\mathbb{S}$ takes the mixture signal and estimate a sigmoid mask to separate the mixture into a source given the summarized information of query from $\mathbb{Q}$. To effectively pass the summary of the query signal to $\mathbb{S}$, we applied two methods. First, we simply concatenated the latent vector along the channel dimension of the input mixture spectrogram expecting the summarized information to be delivered from the start. Second, we used the adaptive instance normalization (AdaIN) technique in the decoding stage of $\mathbb{S}$, which is proven to be effective in many studies for conditioning latent vectors [2, 4]. AdaIN is simply done by applying two steps on each output $\mathbf{x}$ of the convolutional layer (before activation) of the decoder part of $\mathbb{S}$. First, each i-th feature map $\mathbf{x}_i$ is normalized using instance normalization technique [2]. Second, affine transformation is applied to the normalized feature map using learned scale and bias parameters which transforms encoded query vector $\mathbf{z}$ into $\mathbf{y}_s$ and $\mathbf{y}_i$ respectively as follows, $\mathbf{y}_s = W_s^T\mathbf{z}$, $\mathbf{y}_b = W_b^T\mathbf{z}$, where W_s and W_b denote the trainable parameters,

$$\text{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \cdot (\frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)}) + \mathbf{y}_{b,i}. \quad (1)$$

The overall framework of the proposed method is illustrated in Fig. 2.

3.2 Training

3.2.1 Data Sampling

We first describe how the mixture and target source are selected throughout the training phase.

Let, v_i be the single source sampled from i-th source class, where $i \in \{1, 2, 3..., K\}$ and K denote the total number of source classes. We split the classes into two groups by randomly assigning each source class into group T (Target) and R (Rest) without replacement until every class is assigned to one of the two groups. Next, we multiply binary value α_i to the v_i, where α_i being sampled from the Bernoulli distribution, $\alpha_i \sim Bernoulli(0.5)$. This was done to make sure that there are not too many sources included in the mixture. After then, as a data augmentation strategy [20], we scale each source by multiplying a value β_i to source v_i, where β_i is sampled from the Uniform distribution, $\beta_i \sim \mathcal{U}$[0.25, 1.25]. Finally, the sources in each group is added to form two waveforms s_T and s_R and the mixture m is constructed as the linear sum of s_T and s_R as follows,

$$m = s_T + s_R = \sum_{i \in T}(\beta_i \cdot \alpha_i \cdot v_i) + \sum_{j \in R}(\beta_j \cdot \alpha_j \cdot v_j). \quad (2)$$

As we used magnitude spectrogram as input of the modules, m, s_T, and s_R are transformed into short-time-Fourier-transform (STFT) domain, which we denote in capital letter M, S_T and S_R, respectively. Note that, we do not assume any musicality of mixture signal, hence each class is sampled from arbitrary mixture tracks.

3.2.2 cVAE with Latent Regressor

To design the proposed framework, we borrow the formulation of conditional variational autoencoder (cVAE). While the latent vector $\mathbf{z}$ can be deterministically encoded into the latent space, in cVAE framework, $\mathbf{z}$ is instead sampled from the Gaussian distribution, where the parameters of the distribution (mean and variance) are estimated from $\mathbb{Q}$. Then, $\mathbb{S}$ is used to reconstruct S_T given M and $\mathbf{z} \sim \mathbb{Q}(S_T)$. This is ensured by one of the two objectives of cVAE, namely, reconstruction loss $\mathcal{L}_R$. The purpose of $\mathcal{L}_R$ is to guarantee that the output of $\mathbb{S}$ is dependent on the encoded latent vector as follows,

$$\mathcal{L}_R = \mathbb{E}_{S_T \sim p(S_T),\, M \sim p(M),\, \mathbf{z} \sim \mathbb{Q}(S_T)}[\|S_T - \mathbb{S}(M, \mathbf{z})\|]_1. \quad (3)$$

Note that, in training phase, the latent vector $\mathbf{z}$ is sampled using re-parameterization trick to allow backpropagation in training phase [5].

Next, KL-divergence loss is used to make the distribution of $\mathbf{z}$ be close to the Gaussian distribution $\mathcal{N}(\mathbf{0}, I)$ to guarantee a sampling at test time.

$$\mathcal{L}_{\text{KL}} = \mathbb{E}_{S_T \sim p(S_T)}\left[\mathcal{D}_{\mathbf{KL}}(\mathbb{Q}(S_T) \| \mathcal{N}(\mathbf{0}, I))\right] \quad (4)$$

Apart from cVAE framework, we also adopted latent regressor used in [24] to enforce the output of $\mathbb{S}$ to be more dependent on the latent vector. First, a random vector z is drawn from the prior Gaussian distribution $\mathcal{N}(\mathbf{0}, I)$ and passed to $\mathbb{S}$. Then, $\mathbb{S}$ produces a reasonable output reflecting the information in the random vector. Finally, $\mathbb{Q}$ is reused to restore the random vector from the output from $\mathbb{S}$. Note that, unlike Eq. 3 and 4, only the mean values (μ) are taken from $\mathbb{Q}$ as a point estimate of $\mathbf{z}$.

$$\mathcal{L}_{latent} = \mathbb{E}_{M \sim p(M),\, \mathbf{z} \sim p(\mathbf{z})}\|\mathbf{z} - \mathbb{Q}(\mathbb{S}(M, \mathbf{z}))\|_1 \quad (5)$$

Finally, the total loss can be written as follows,

$$\mathcal{L}_{Total} = \lambda_R\mathcal{L}_R + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}} + \lambda_{latent}\mathcal{L}_{latent}. \quad (6)$$

3.3 Test

During the training phase, $\mathbb{S}$ was trained to separate the target source by using the target source as a query as in Eq. 3. In the test phase, however, the target source to be separated from the mixture is unknown. Hence, the target source and query can no longer be the same. Nevertheless, since we designed the output dimension of $\mathbb{Q}$ to be small enough, the latent vector $\mathbf{z}$ is trained to have a high-level information such as instrument class. In the test phase, therefore, we can utilize this property in many ways. For example, when the user wants to separate a specific source in the mixture, it is possible to collect a small amount of audio samples that have similar characteristics but not exactly the same to the source of interest. Then, the user can extract that specific source by feeding the collected audio samples into the Query-net and passing the summarized information to the Separator.

Apart from the query dependent approach, we can also take the average of latent vectors of each source class in the training set and use it as a representative latent vector that reflects the general characteristics of a single class.

4. EXPERIMENT

4.1 Dataset

We trained our network with the MUSDB18 dataset. The dataset consists of 100 tracks for training set and 50 tracks for test set and each track is recorded in 44.1kHz, stereo format. The dataset provides the mixture and coarsely defined labels for sources, namely, 'vocals', 'drums', 'bass' and 'other'. The class 'other' includes every instrument other than 'vocals', 'drums' and 'bass', providing the most coarsely defined class. We resampled the audio to 22050Hz and divided each track into 3-second segments. Magnitude spectrogram was obtained by applying STFT with a window size 1024 and 75% overlap. To restore the audio from the output, Inverse STFT is applied using the phase of the mixture. We evaluated our method on the test set of MUSDB18 using the official museval package [1] which computes signal-to-distortion ratio (SDR) as a quantitative measurement.

4.2 Experiment Details

The followings are the experimental details of our method. $\mathbb{Q}$ consists of 6 strided-convolutional layers with 4×4 filter size and the number of output channels for each layer is 32, 32, 64, 64, 128 and 128, respectively. Every strided-convolutional layer has the stride size of 2 along the frequency axis and only second, fourth and sixth layers have a stride size of 2 along the time axis. After every convolutional operation, we used instance normalization and relu. We used GRU with 128 units. The length of the query segment was fixed to 3-second in every experiment. For $\mathbb{S}$, the encoder part consists of 9 strided-convolutional layers and the decoder part consists of the same number of strided-deconvolutional layers, with a filter size of 4×4. The number of output channels for first, second, and third layer is 64, 128, 256, respectively, and 512 for the rest of the layers. Every layer has stride size of 2 along the frequency axis. And stride size along the time axis is set to 2 for every layer except the first layer of the encoder and the last layer of the decoder.

The dimension of the latent vector was set to 32 and the batch size was set to 5. The coefficients in Eq.6 were set to $\lambda_R = 10$, $\lambda_{\mathrm{KL}} = 0.01$, $\lambda_{latent} = 0.5$. The initial learning rate was set to 0.0002 and after 200000 iterations the rate was decreased to 5×10^{-6} for every 10000-iteration. We used Adam optimizer with $\beta_1 = 0.5$, $\beta_2 = 0.999$.

4.3 Manually Targeting a Specific Sound Source

To validate that our method captures the characteristics of the audio given in the query and separates them accordingly, we conducted an experiment of separating specific instruments. As shown in Fig. 3, an audio query of hi-hat and piano were given to the mixtures of (hi-hat + kick drum + bass) and (piano + electric guitar). Queries and mixtures were not from the train set, and both queries were not sampled from the mixture. We can observe in the hi-hat separation result that the kick drums and the bass which lie in the low-frequency band were mostly removed while broadband components of hi-hat remained. The result of piano separation is not as clear as in the case of hi-hat, but we can see the guitar was removed considerably.

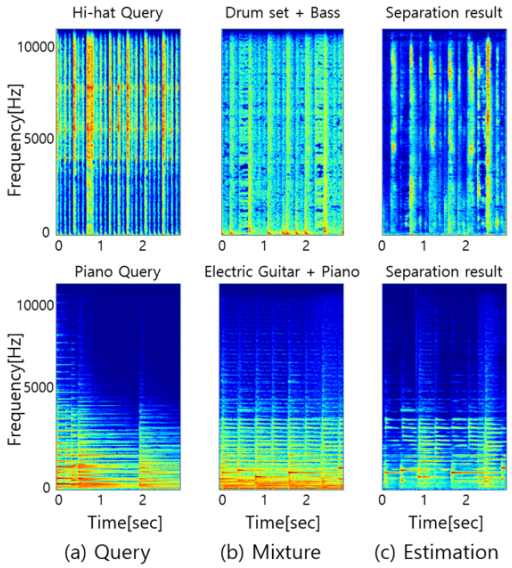


Figure 3. Results of manually targeting specific sound sources. The first row show the separation results of hi-hat from the mixture of hi-hat, kick drum and bass. The second row shows the separation results of piano from the mixture of electric guitar and piano. It is worth noting that the network was never trained to only separate a hi-hat component from 'drum' class nor piano from 'other' class.

The noticeable fact is that we trained our method only with the MUSDB18 dataset, which has no hierarchical class label information besides the coarsely defined labels of sources such as 'vocals', 'drums', 'bass' and 'other'. Under the definition of class in the dataset, hi-hat and kick drum are grouped into 'drums', and piano and electric guitar into 'other'. Although our method was never trained to separate the subclass from the mixture, it was able to separate hi-hat and piano from the mixture, which can be referred to as a zero-shot separation. These results indicate the proposed method can be well applied for audio query-based separation.

4.4 Latent Interpolation

Furthermore, we conducted a latent interpolation experiment using the mean vector of each source. The mean vector of each source was computed by averaging the latent vectors of each source in the training set, $\mathbf{z}_c = \frac{1}{N_c} \sum_i \mathbb{Q}(S_{c,i})$, where $S_{c,i}$ denotes i-th 3-second magnitude spectrogram in the sound class c and N_c denotes the number of segments in class c.

For the interpolation method, we used the spherical linear interpolation ($Slerp$) introduced in [23],

$$Slerp(\mathbf{z}_1, \mathbf{z}_2; \alpha) = \frac{\sin(1-\alpha)\theta}{\sin\theta}\mathbf{z}_1 + \frac{\sin\alpha\theta}{\sin\theta}\mathbf{z}_2, \quad (7)$$

[1] https://sigsep.github.io/sigsep-mus-eval

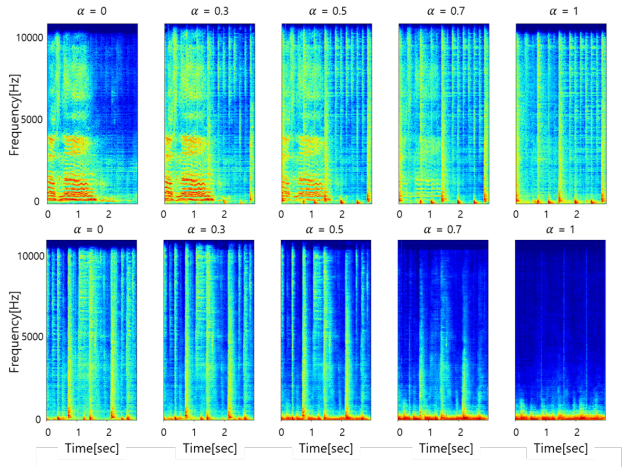


Figure 4. Results of the mean vector interpolation. The first row shows the interpolation results between vocals and drums. The second row shows the interpolation results between drums and bass.

where α denotes the weight of interpolation and θ denotes the angle between $\mathbf{z}_1$ and $\mathbf{z}_2$. As shown in Fig. 4, we interpolated between the mean vector of sound sources, drums ($\mathbf{z}_{drums}$) $\rightarrow$ bass ($\mathbf{z}_{bass}$) and vocals ($\mathbf{z}_{vocals}$) $\rightarrow$ drums ($\mathbf{z}_{drums}$). We can see the ratio of separated instruments changes as the weight α changes. These experimental results show that our method can generate continuous outputs just by manipulating a latent space.

4.5 Effects of Latent Vector on Performance

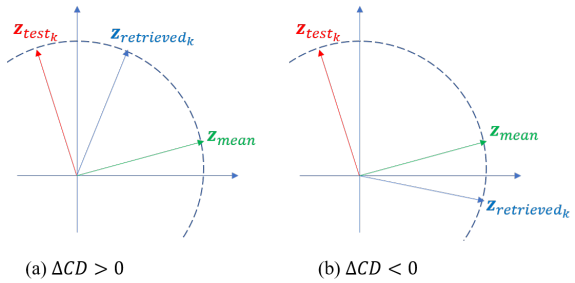


Figure 5. Illustration of two ΔCD cases. (a) shows the positive ΔCD case where we assume that the performance should be improved. (b) shows the negative ΔCD case where we assume that the performance should be worsened.

This subsection investigates the performance improvement varying the latent vector and see in which situation we can achieve a performance improvement. For the experiment, we first obtained the mean vector of each vocal track from the entire dataset as follows, $\mathbf{z}_i = \frac{1}{N_i}\sum_j \mathbb{Q}(S_{i,j})$, where i denotes a i-th vocal track, j denotes a j-th segment in i-th vocal track, and N_i denotes the number of segments in i-th vocal track. Then, we obtained the mean vector, $\mathbf{z}_{mean} = \frac{1}{100}\sum_{i \in training} \mathbf{z}_i$, of vocal tracks from training set. Finally, we retrieved the latent vector of certain vocal track $\mathbf{z}_{ret_k}$ from the training set which has the

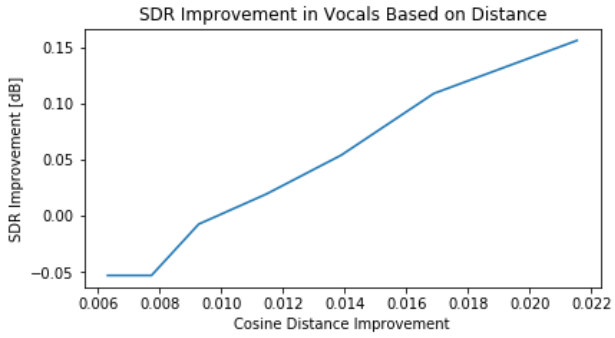


Figure 6. The relationship between SDR improvement (ΔSDR) and cosine distance difference (ΔCD) in vocal tracks.

closest cosine distance (CD) from k-th test vocal track $\mathbf{z}_{test_k} = \mathbf{z}_k$, $k \in test$ as follows,

$$\tilde{k} = \underset{i \in training}{\arg\min}\ CD(\mathbf{z}_i, \mathbf{z}_{test_k}),\ \ \mathbf{z}_{ret_k} = \mathbf{z}_{\tilde{k}}. \quad (8)$$

We compare the performance of two cases where the goal is to separate a k-th vocal track from test set. The first case is to use $\mathbf{z}_{mean}$ to separate a target source, $\hat{S}_{mean} = \mathbb{S}(M, \mathbf{z}_{mean})$. The second case is to use $\mathbf{z}_{ret_k}$ to separate a target source, $\hat{S}_{ret_k} = \mathbb{S}(M, \mathbf{z}_{ret_k})$. We defined performance improvement in terms of SDR as follows,

$$\Delta\text{SDR} = \text{SDR}(S_{GT_k}, \hat{S}_{ret_k}) - \text{SDR}(S_{GT_k}, \hat{S}_{mean}), \quad (9)$$

where S_{GT_k} denotes k-th ground truth vocal track from test set. To measure the distance between latent vectors we used cosine distance ($CD(\mathbf{z}_1, \mathbf{z}_2) = 1 - (\mathbf{z}_1 / \|\mathbf{z}_1\|_2) \cdot (\mathbf{z}_2 / \|\mathbf{z}_2\|_2)$) and defined cosine distance difference between ($\mathbf{z}_{test_k}, \mathbf{z}_{ret_k}$) and ($\mathbf{z}_{test_k}, \mathbf{z}_{mean}$) as follows,

$$\Delta CD = CD(\mathbf{z}_{test_k}, \mathbf{z}_{mean}) - CD(\mathbf{z}_{test_k}, \mathbf{z}_{ret_k}). \quad (10)$$

Fig. 5 illustrates two possible cases of using $\mathbf{z}_{ret_k}$. (a) shows the positive ΔCD case where we assume to induce positive effect on performance improvement ($\Delta\text{SDR} > 0$). In this case, we expect the performance to be improved since $\mathbf{z}_{ret_k}$ is expected to contain information close to $\mathbf{z}_{test_k}$ compared to $\mathbf{z}_{mean}$. (b) shows the negative ΔCD case where we assume to induce negative effect on performance improvement ($\Delta\text{SDR} < 0$). In this case, we expect the performance to be worsened as the system could not retrieve a $\mathbf{z}_{ret_k}$ that is close enough to $\mathbf{z}_{test_k}$. To empirically prove our assumption, we show the relationship between ΔSDR and ΔCD in Fig. 6. We can observe that the closer the vector gets to the targeted ground-truth vector, the larger the performance gain becomes, therefore reinforcing our assumption that better performances can be achieved if we can obtain closer latent vectors to the target latent vector.

4.6 Iterative Method

In this subsection, we seek a performance improvement by automating the query-based framework in an iterative

way, which we refer to as an iterative method. The iterative method is done as follows. First, we separate the target source using the mean vector of certain sound class $\mathbf{z}_{mean}$. Then, we re-encode the separated source into a latent space expecting the re-encoded latent vector to be closer to the target latent vector. Finally, we separate the target source using the re-encoded latent vector. We verify the effect of the proposed iterative method and show that it can be helpful under the harsh condition where the target sources are far from generic class. The results (Single step $\rightarrow$ Iterative) are as follows, 'vocals': $4.84 \rightarrow \mathbf{4.90}$, 'drums': $4.31 \rightarrow \mathbf{4.34}$, 'bass': $\mathbf{3.11} \rightarrow 3.09$, and 'other': $2.97 \rightarrow \mathbf{3.16}$. We can see the iterative method noticeably improves the performance in 'vocals' and 'other'. On the other hand, the differences are not significant in drums and bass.

We looked into the tracks which gained significant improvement in terms of SDR in vocals. 'Timboz - Pony' and 'Hollow Ground - Ill Fate' gained more than 0.5dB in SDR through the iterative method. We found the results intuitive as the vocals in the two songs feature a growling technique from heavy metal genres, which can be considered distant from the general characteristics of vocals.

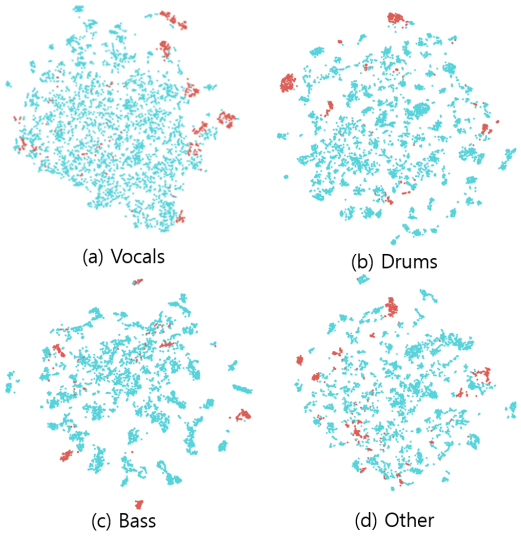


Figure 7. t-SNE visualization of encoded latent vectors from each source in the test dataset. Red points denote the vectors of the tracks which gained more than 0.4dB in terms of SDR by the iterative method.

To verify our assumptions, we divided each source of the test set into segments and converted them into latent vectors. We divided the encoded vectors into two groups, the ones which gained more than 0.4dB in terms of SDR by the iterative method and the ones did not. Then, we visualized the encoded vectors using t-SNE (results shown in Fig. 7). The red dots in Fig. 7 represent the latent vector from the group that showed significant SDR improvement more than 0.4dB. Although some vectors lie around the center, most of them are located far from the center. These vectors can be inferred as outliers and the results show that our iterative method is effective when it comes to separating the sources of distinctive characteristics.

	Vocals	Drums	Bass	Other
STL2 [16]	3.25	4.22	3.21	2.25
WK [22]	3.76	4.00	2.94	2.43
RGT1 [12]	3.85	3.44	2.70	2.63
JY3 [6]	5.74	4.66	3.67	3.40
UHL2 [20]	5.93	5.92	5.03	**4.19**
TAK1 [18]	**6.60**	**6.43**	**5.16**	4.15
Ours (mean)	4.90	4.34	3.09	3.16
Ours (GT)	5.48	4.59	3.45	3.26

Table 1. Median scores of SDR for the MUSDB18 dataset.

4.7 Algorithm Comparison

In this subsection, we compare our method to other methods with the evaluation result of the MUSDB18 dataset. As stated above, our method's output is dependent on the encoded latent vector from a query. For the comparison with other methods that do not require a query, therefore, we used the mean vector in the latent space encoded from the training samples for each source – *i.e.*, we ended up using four mean latent vectors for 'vocals', 'drums', 'bass', and 'other', respectively. Additionally, to show the upper bound of our proposed method, we used the encoded latent vector of the ground truth (GT) signal from test set. Note also that the separation is done with a single network.

Table 1 shows the median scores of SDR of methods reported in SiSEC2018 [17], including our method denoted as Ours. Although the proposed algorithm did not achieve the best performance, the results show that it is comparable to the other deep learning-based models that are dedicated to separating just four sources in the dataset. This means that our method is not limited to query-based separation, but also can be used for general music source separation just like as other conventional methods. Additionally, there is room for improvement: applying the multichannel Wiener filter and/or using other architecture for the separator besides U-net could be such an option.

5. CONCLUSION

In this study, we presented a novel framework, consisting of Query-net and Separator, for audio query-based music source separation. Experiment results showed that our method is scalable as the Query-net directly encodes audio query into a latent space. The latent space is interpretable as was shown by the t-SNE visualization and latent interpolation experiments. Furthermore, we have introduced various utilities of the proposed framework including manual and automated approach showing the promise of audio-query based source separation. As a future work, we plan to investigate more adequate conditioning method for audio and better neural architecture for performance improvement.

6. ACKNOWLEDGEMENT

This work was supported partly by Kakao and Kakao Brain corporations and partly supported by Institute for Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.2019-0-01367, Infant-Mimic Neurocognitive Developmental Machine Learning from Interaction Experience with Real World (BabyMind)).

7. REFERENCES

[1] John R Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe. Deep clustering: Discriminative embeddings for segmentation and separation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 31–35. IEEE, 2016.

[2] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[3] Andreas Jansson, Eric J. Humphrey, Nicola Montecchio, Rachel M. Bittner, Aparna Kumar, and Tillman Weyde. Singing voice separation with deep u-net convolutional networks. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pages 745–751, 2017.

[4] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018.

[5] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[6] Jen-Yu Liu and Yi-Hsuan Yang. Denoising auto-encoder with recurrent skip connections and residual regression for music source separation. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 773–778. IEEE, 2018.

[7] Antoine Liutkus, Fabian-Robert Stöter, Zafar Rafii, Daichi Kitamura, Bertrand Rivet, Nobutaka Ito, Nobutaka Ono, and Julie Fontecave. The 2016 signal separation evaluation campaign. In Petr Tichavský, Massoud Babaie-Zadeh, Olivier J.J. Michel, and Nadège Thirion-Moreau, editors, *Latent Variable Analysis and Signal Separation - 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings*, pages 323–332, Cham, 2017. Springer International Publishing.

[8] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[9] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016.

[10] Sungheon Park, Taehoon Kim, Kyogu Lee, and Nojun Kwak. Music source separation using stacked hourglass networks. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, pages 289–296, 2018.

[11] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. The MUSDB18 corpus for music separation, December 2017.

[12] Gerard Roma, Owen Green, and Pierre Alexandre Tremblay. Improving single-network single-channel separation of musical audio with convolutional layers. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 306–315. Springer, 2018.

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[14] P. Seetharaman, G. Wichern, S. Venkataramani, and J. L. Roux. Class-conditional embeddings for music source separation. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 301–305, May 2019.

[15] O. Slizovskaia, L. Kim, G. Haro, and E. Gomez. End-to-end sound source separation conditioned on instrument labels. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 306–310, May 2019.

[16] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, pages 334–340, 2018.

[17] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito. The 2018 signal separation evaluation campaign. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 293–305. Springer, 2018.

[18] Naoya Takahashi, Nabarun Goswami, and Yuki Mitsufuji. Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation. In *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, pages 106–110. IEEE, 2018.

[19] Naoya Takahashi and Yuki Mitsufuji. Multi-scale multi-band densenets for audio source separation. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 21–25. IEEE, 2017.

[20] Stefan Uhlich, Marcello Porcu, Franck Giron, Michael Enenkl, Thomas Kemp, Naoya Takahashi, and Yuki Mitsufuji. Improving music source separation based on deep neural networks through data augmentation and network blending. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 261–265. IEEE, 2017.

[21] Jun Wang, Jie Chen, Dan Su, Lianwu Chen, Meng Yu, Yanmin Qian, and Dong Yu. Deep extractor network for target speaker recovery from single channel speech mixtures. In *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018.*, pages 307–311, 2018.

[22] Felix Weninger, John R Hershey, Jonathan Le Roux, and Björn Schuller. Discriminatively trained recurrent neural networks for single-channel speech separation. In *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 577–581. IEEE, 2014.

[23] Tom White. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016.

[24] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, pages 465–476, 2017.

MOSAIC STYLE TRANSFER USING SPARSE AUTOCORRELOGRAMS

Dan MacKinlay and Zdravko Botev
School of Mathematics and Statistics, UNSW Sydney
dan@danmackinlay.name

ABSTRACT

We introduce a novel mosaic synthesis algorithm for musical style transfer using the autocorrelogram as a feature map. We decompose the autocorrelogram feature map sparsely in a decaying sinusoid basis, using that decomposition as an interpolation scheme in feature space. This efficiently provides gradient information in the mosaicing optimization, including gradients of the challenging time-scale parameters, which are usually computationally intractable for discretely sampled signals. The required calculations are straightforward to parallelize on vector-processing hardware. Our implementation of the method provides good quality output and novel musical effects in example tasks by itself and can also be integrated into alternative mosaicing methods.

1. INTRODUCTION

Mosaicing synthesis is a particular approach to the style transfer problem. As with all style transfer methods, the goal is combining two signals, a source, and a target, to produce a hybrid output signal with qualities of each, which we call a *mosaic*. A musical application of these methods would typically use the 'style' of one signal, the timbre, to express the 'content' of another, a melody. Concretely, if the target were a trumpet playing a melody, and the source a recording of a singing vocalist, the mosaic might aim to emulate the vocalist singing that melody.

There exist a variety of problem definitions of, and associated algorithms for, mosaicing synthesis; e.g. [8, 12, 13, 22, 23, 35, 41, 45], partially summarized in [32]. In mosaicing specifically, we accomplish style transfer using a dictionary-based granular synthesis method, which constructs its output by superposition of transformed short recordings, *grains*, from an audio dictionary, in the time or, more recently, spectral domains [1, 7, 16].

The granular synthesis methods in themselves are well understood and widely deployed in industrial applications. They comprise a significant proportion of the music industry market for software synthesizers, are integrated into every major Digital Audio Workstation package, and have been extensively researched – see e.g. [31] and references therein.

The extension of granular synthesis into a style-transfer problem as mosaicing is less well-understood. In this setting we choose the parameters of a granular synthesis so as to optimally approximate a desired target audio signal in the sense of optimising some measure of acoustic similarity. Typically, this implies approximating, in the sense of minimising some approximation loss, the power spectral density (PSD) of the target signal. Applications for this include musical accompaniment, creative musical effects, or user customization of speech synthesis [10].

Our sparse autocorrelogram method advances the capabilities of musical mosaicing applications, by leveraging a feature map that is related to, but more convenient than, classical PSD methods. This method is enabled by two major innovations.

Firstly, we define signal similarity through the *autocorrelogram*, a representation of the signal as covariance with delayed versions of itself. The autocorrelogram and its relationship to PSD is well-known (e.g. [44]) but our use in mosaicing synthesis appears novel. Although we use the autocorrelogram in a standalone procedure, it may be included in the feature vectors of loss functions of other mosaic techniques and is thus of independent interest.

Secondly, we decompose the high-dimensional empirical autocorrelogram into a sparse dictionary of decaying sinusoids. By interpolating discrete signals, this procedure calculates both error and gradients efficiently, enabling gradient-based optimization. The resulting technique is flexible and straightforward to parallelize on modern Single Instruction Multiple Data (SIMD) architectures such as Graphics Processing Units (GPUs).

We make our Python code [1] openly available for public use. We thereby aim to facilitate both the investigations of future researchers and the immediate application of these methods by musicians. Comparisons are made with benchmark mosaicing implementation, NiMFKS [7].

2. PRIOR WORK

Style transfer techniques, construed broadly, have a long history in signal processing research. Early work in this area begins with the channel vocoder [17], via various innovations to the modern repertoire of methods which includes innovations such as neural style transfer methods

Attribution: Dan MacKinlay and Zdravko Botev . "Mosaic style transfer using sparse autocorrelograms", 20th International Society for Music Information Retrieval Conference, Delft, The Netherlands, 2019.

[1] https://github.com/danmackinlay/mosaicing_omp_ismir_2019/

[19, 21, 43]. In the style transfer field, the mosaicing techniques form a sub field which fix a choice of synthesis to dictionary-based granular synthesis techniques.

We are concerned with the musical applications of style transfer. The archetypal task in this context is using the timbre of the 'style' signal to express the melodic 'content' of another. Concretely, if the target were a trumpet playing a melody, and the source a recording of a singing vocalist, the output should emulate the vocalist singing that melody.

In mosaicing synthesis, the task of choosing synthesis parameters to produce the desired output is non-trivial and subject of ongoing interest. Notable recent progress includes matrix factorization methods to decompose audio [1, 7, 16], various improvements in spectral matrix factorization [1, 7, 16] and optimization over features [8, 11, 36]. However, few methods can conveniently handle time-scaling of audio, so that time-scale parameters must be ignored, or selected by exhaustive search. One recent exception is Sound Retiler [1], which claims to handle time shifting via tensor decomposition. It is in this area that we make our main contribution, by the application of autocorrelogram features in this task.

While the autocorrelogram itself is not new in audio synthesis (e.g. [38]), our application to the mosaicing problem seems novel. The autocorrelogram-based analysis in combination with sparse coding induces a novel and analytically differentiable expression for the time scale parameter, and it is this we use to solve the mosaic problems.

3. PROBLEM DESCRIPTION

3.1 Audio signals and notation

We work with audio signals, a Hilbert space $\mathcal{H}$ of real L_2 functions $f : \mathbb{R} \to \mathbb{R}$ mapping time to instantaneous signal pressure level. Where the argument of the signal is clear, we abbreviate notation, writing for example, $t \mapsto f(at)$ as $f(at)$. We will handle transforms on signals $f(.)$ such as the autocorrelogram $\mathcal{A}$, and Fourier transform $\mathcal{F}$. Where not clear from context which argument of the signal with respect to which the transform is taken, we indicate it with a subscript to the transform. Thus $\mathcal{F}_t\{f(s,t)\}(\xi) := \int e^{-2\pi i t\xi} f(s,t)\mathrm{d}t$. Where we specify a weight v for the inner product or norm, we write it as a subscript, i.e. $\langle f, g\rangle_v := \int_{\mathbb{R}} v(t)f(t)g(t)\mathrm{d}t$.

In practice we do not observe continuous audio signals, but discretely sampled observations of signals. Sampling fidelity will be assumed, requiring signals are band-limited to some suitably low cutoff period Ω. We scale time so that the sample period $T = 1$ and $\Omega > 1/2$. The sampling process is a train of Dirac impulses, and inner products with a discrete signals are defined

$$\langle g, f\rangle_v := \sum_{t\in\mathbb{Z}} v(t)g(t)f(t). \tag{1}$$

We denote length-M vectors in bold, $\boldsymbol{x} = [x_1, x_2, \ldots, x_M]^\intercal$.

3.2 Mosaicing

Given a target signal f_0, we seek an approximant, the mosaic $\hat{f}_0$, as a sparse linear combination of scaled signals, called *codes*, from a source *dictionary* $\mathfrak{G} := \{g_1, \ldots, g_D\}$ subject to a maximum budget of J codes. In our earlier style transfer example, say, f_0 would be the recorded trumpet melody and $\mathfrak{G}$, recordings of the singing vocalist. For a fixed dictionary the mosaic is specified completely by the length-J parameter vectors $\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\rho}$ and written

$$\hat{f}_0(t; \boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\rho}) = \sum_{j=1}^{J} \alpha_j g_{\gamma_j}(\rho_j t). \tag{2}$$

The problem requires selecting approximately optimal values for parameter vectors

$$\{\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\rho}\} \simeq \underset{\{\boldsymbol{\alpha},\boldsymbol{\gamma},\boldsymbol{\rho}\}}{\operatorname{argmin}} d\left(\hat{f}_0(t; \boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\rho}),\ f_0(t)\right), \tag{3}$$

where $\rho_j \in \mathbb{R}^+, \alpha_j \in \mathbb{R}, \gamma_j \in \{1, \ldots, D\}$ and $d : \mathcal{H} \times \mathcal{H} \mapsto \mathbb{R}^+$ is a distance function quantifying the poorness of the approximation. In contrast to sparse coding for signal compression, $\hat{f}_0$ is an intentionally imperfect approximation of f_0, possessing qualities of both the source and target signals, hence the designation style *transfer*.

4. AUTOCORRELATION MOSAICING METHOD

The autocorrelogram mosaicing method has two stages.

1. In the pre-training stage, autocorrelogram features are computed from the source signals, and decomposed in a dictionary of decaying sinusoids.
2. In the inference stage, we search our dictionary of autocorrelogram decompositions for matches to the autocorrelogram of the target signal, and solve an inverse problem, synthesizing a corresponding mosaic from our result.

Both stages leverage convenient properties of autocorrelograms, and sparse dictionary decompositions, which we now introduce.

4.1 Properties of autocorrelograms

We now motivate the use of the autocorrelogram in our feature map. As with other style transfer methods we face the challenge that sample values of a time domain audio signal f are only indirectly indicative of how human listeners will perceive it. For audio analysis, one typically operates on a feature map $\mathcal{P}\{f\}$ which is in some sense closer to human perception of these signals. Specifically, we aim to find a feature map such that two signals are similar if some distance between their feature vectors is small, i.e. the similarity of f and $\hat{f}$ is high iff the distance $d_{\mathcal{P}}(f, \hat{f}) := \|\mathcal{P}\{f_0\} - \mathcal{P}\{\hat{f}\}\|$ is low, with some choice of norm $\|\cdot\|$. We would like $d_{\mathcal{P}}$ to approximate specifically *psychoacoustic* similarity, which is to say $d_{\mathcal{P}}(f, \hat{f})$ is small iff a typical human listener would perceive f and $\hat{f}$

as similar. Ideally the image of the feature map should also be of lower dimension than f, and $d_{\mathcal{F}}$ should be computationally efficient to manipulate.

True psychoacoustic similarity is not well-defined, so practical algorithms settle for feature maps compromising between convenience and psychoacoustic plausibility. Usually, feature maps are empirical PSDs [8, 23], or are derived from the PSD, as with the Mel-Frequency Cepstral Coefficient (MFCC) [27] or the Constant-Q transform [6]. These maps induce expensive mosaicing optimization problems [11, 12]. MFCCs for example, are suitable for low-dimensional indexing and search but are hard to invert. A raw empirical PSD is easier to invert, via, e.g. Griffin-Lim iteration, but of the same order of dimensionality as the original signal and thus difficult to search. One could ameliorate this difficulty if a computationally convenient feature map could be found which was well-behaved under operations of scaling and superposition, as in Eq. 2, so that one could conduct as much calculation as possible in the feature space.

These desiderata suggest the autocorrelogram map

$$\mathcal{A}\{f\} : \xi \mapsto (\xi \mapsto \langle f(t), f(t-\xi) \rangle) \,. \tag{4}$$

This is the deterministic covariance between $f(t)$ and $f(t-\xi)$. The autocorrelogram is an even function in ξ, so we work with one-sided autocorrelograms $\mathbb{R}^+ \to \mathbb{R}$. Autocorrelogram-like transforms are implicated in the neurological processing of harmonic audio by human listeners [3, 9, 25, 26]. For our purposes, the supposed neurological basis is a secondary consideration to the demonstrated empirical usefulness in psychoacoustic tasks, most notably in pitch-detection [30, 37, 40]. In this regard it resembles the cepstral analysis method [5], which also effectively identifies small numbers of periodic components by analysing a pointwise non-linear transformation of the power spectrogram , but unlike the cepstrum it is well-behaved under superposition.

Specifically, brief calculation shows the following useful properties: a) Multiplication by a constant $c \in \mathbb{R}$:

$$\mathcal{A}\{cf\}(\xi) = c^2 \mathcal{A}\{f\}(\xi). \tag{5}$$

b) Time scaling:

$$\mathcal{A}\{f(rt)\}(\xi) = \frac{1}{r} \mathcal{A}\{f\}\left(\frac{\xi}{r}\right) \tag{6}$$

c) Randomized addition:

$$\mathbb{E}\left[\mathcal{A}\{S_1 f + S_2 f'\}(\xi)\right] = \mathcal{A}\{f\}(\xi) + \mathcal{A}\{f'\}(\xi), \tag{7}$$

where $\{S_i\}$ are i.i.d. Rademacher variables, taking values in $\{+1, -1\}$ with equal probability.

We note two obstacles to the application of these formulae in the mosaicing problem. Firstly, Eq. 6 is not well-defined for the discrete signals that comprise the usual subject matter of digital signal processing. We will handle discrete signals by continuous interpolants, which turn out to be practically sufficient approximations. Secondly, the additive rule c) is valid only in expectation, via the contrivance of introducing Rademacher random variables. Solving for the deterministic case by accounting for phase cancellation is indeed possible, but considerably more involved, and constitutes an active area of research in its own right in, e.g. the Overlap-Add [15, 42], and phase retrieval [24, 34] literatures. As the randomised solution also turns out in practice to be already sufficient for many tasks, we defer such extensions to future work.

In order to construct these interpolants efficiently, we decompose discrete autocorrelograms using a matching pursuit, which we now introduce.

4.2 Orthogonal matching pursuit

In orthogonal matching pursuit (OMP) [14, 28], given a target signal f_0 and a dictionary of *code* signals $\mathfrak{D} = \{g_\theta\}_{\theta \in \Theta}$, one finds a decomposition $\hat{f}_0 = \mathrm{OMP}_{\mathfrak{D},K}(f_0)$ of form

$$f_0 \simeq \underset{\mathfrak{D},K}{\mathrm{OMP}}(f_0) := \sum_{i=1}^{K} \mu_i g_{\theta_i}. \tag{8}$$

A solution is a parameter vector $[\theta_1, \ldots, \theta_K] \in \Theta^k$ and code weights $[\mu_1, \ldots, \mu_K] \in \mathbb{R}^K$ which nearly minimize $\|f_0 - \hat{f}_0\|$. We require that f_0 and all codes g_θ are L_2 integrable and not null, i.e. possessing positive norm, $\|g_\theta\| > 0$.

The OMP algorithm is as follows.

1. Initialization. Let the first residual be $r_0 := f$. Set step counter $k \leftarrow 1$.
2. Find θ_k such that (possibly approximately)

$$\theta_k = \underset{\theta}{\mathrm{argmax}}\, A(r_k, g_\theta) \tag{9}$$

where A is the *normalized code product*

$$A(r_k, g_\theta) := \frac{\langle r_{k-1}, g_\theta \rangle}{\|g_\theta\|}. \tag{10}$$

3. Solve the least sum of squares problem

$$[\mu_1^k, \ldots \mu_k^k] = \underset{[\mu_1, \ldots, \mu_k]}{\mathrm{argmin}} \left\| \sum_{1 \le \ell \le k} \mu_\ell g_{\theta_\ell} - f_0 \right\| \tag{11}$$

giving kth decomposition $\hat{f}^k = \sum_{1 \le \ell \le k} \mu_\ell g_{\theta_\ell}$.

4. Update the residual $r_{k+1} = f_0 - \hat{f}^k$.
5. If $k = K$, stop, otherwise set $k \leftarrow k + 1$ and repeat from step (2).

We allow the components of θ to be either a) a discrete and finite, or b) a continuous parameter. For finitely enumerable components $\theta_{\text{finite}} \subseteq \theta$ we maximize normalized code product in Eq. 9 by enumeration. For continuous components $\theta_{\text{cts}} \subseteq \theta$ we assume that we can choose θ_{cts}

approximately by iterative optimization using the gradient $\nabla_{\theta_{\text{cts}}} A(r_k, g_\theta)$. As the objective may not attain a global maximum, we choose $I \geq 1$ different initial guesses, and select the best local optimum attained. A first order gradient ascent with fixed number of steps performs well in our examples and moreover requires no branching instructions, as suits our goal of a SIMD-compatible algorithm.

4.3 Sparse approximate autocorrelograms

In the pre-training stage, we find autocorrelograms for each of the empirical source autocorrelogram codes in $\mathfrak{G}$, decomposing them into a dictionary of sparse OMP matches, $\mathfrak{M}$. It is this dictionary which we search for mosaic matches, using matches here to identify approximately matching codes in the original space $\mathfrak{G}$.

In this section we use ξ as the free argument for signals, and restrict $\xi > 0$. For the interpolant dictionary we use decaying sinusoids

$$\mathfrak{S} := \{h(\xi; \omega, \tau, \phi) := \cos(\omega\xi + \phi)e^{-\tau\xi} : \phi, \tau, \omega \in \mathbb{R}\}. \tag{12}$$

The dictionary choice must ultimately be justified by empirical performance, which we demonstrate in the final section of the paper. It is notable that there are also *a priori* reasons for favouring this one for musical audio. Firstly, this basis will decompose an autocorrelogram into a global approximant, rather than a piecewise interpolant, as with for example polynomial splines. Evaluations of such an interpolant are tractable to parallelise without branching instructions, and therefore better suited to modern SIMD architectures.

Secondly, decaying sinusoid models are effective in compactly decomposing time-domain audio [20], and the nature of the autocorrelogram suggests that they could be similarly useful and even more compact in decomposing autocorrelograms. The space of superpositions of decaying sinusoids is, by inspection, closed under the autocorrelogram transform, so it is at just as plausible to represent autocorrelograms in a such a decaying sinusoid dictionary. The question remains how compact such a representation is. Analytic expansion of the superposition of many decaying sinusoids is a lengthy exercise in elementary calculus. However, we have reason to suspect that the amplitude coefficient of most terms in such expansions will negligible. Recall the Wiener-Khintchine theorem, which says that, for signals of finite energy, assuming all these terms are well-defined,

$$\mathcal{F}_\xi\{\mathcal{A}\{f\}(\xi)\}(s) = |\mathcal{F}_t\{f(t)\}(s)|^2$$

where $\mathcal{F}_\xi\{f(\xi)\}$ is the Fourier transform of signal $\xi \mapsto f(\xi)$. This tells us that the magnitude of sinusoidal components of the autocorrelogram are squared with respect to the magnitude of sinusoidal components of the PSD, and thus relatively sparser. This indicates that for autocorrelograms of musical signals, which are well approximated by a superposition of sinusoidal signals, the autocorrelogram could often be approximated with comparable relative error by a yet smaller number of sinusoidal signals, as can be seen in Fig. 1. Moreover, we know that the envelope of musical audio spectral content decays eventually super-exponentially with frequency [18] and thus high frequency content of an autocorrelogram will in general be proportionally even lower. This latter fact additionally implies that the autocorrelogram calculations might even be downsampled with little loss in information content, and some computational saving.

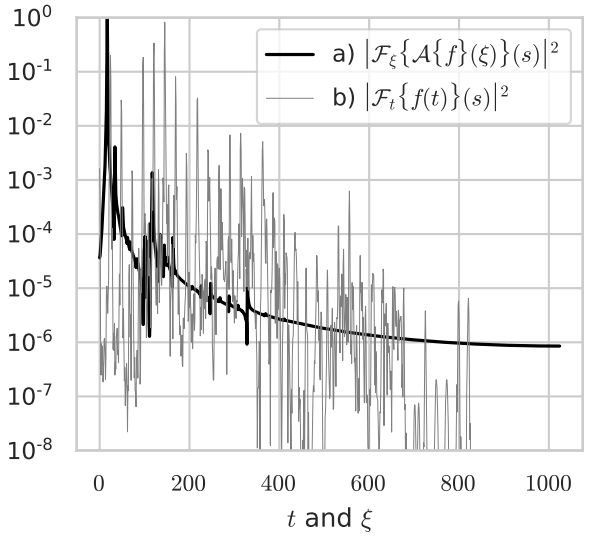


Figure 1. The relatively simple form of (a) the PSD of the autocorrelogram versus (b) the PSD of the signal itself. Signal is a length 2048 recording of a trumpet note onset. The scale of the vertical axis is arbitrary, and signals have been normalized for comparison. Sample period is $1/44100s$.

Implementing the decomposition is straightforward. For each code $g \in \mathfrak{G}$ we perform the following calculation: First, we find the empirical autocorrelogram $\mathcal{A}\{g\}$ at L points $\xi = 0, 1, \ldots, (L-1)$ with Eq. 4.

Next, we decompose each $\hat{G} = \text{OMP}_{\mathfrak{S},C}(\mathcal{A}\{g\})$ over the decaying sinusoid dictionary, as defined in Eq. 12. There are many methods of fitting decaying sinusoids to time series [2, 29, 33], but OMP is convenient in the current application [20] as we may re-use the same algorithm in the reconstruction stage of this algorithm. Autocorrelograms of musical audio in our experiments are highly sparse with respect to this decaying sinusoid dictionary, typically achieving negligible residual error with number of components $C \leq 4$.

We will apply the OMP with product $\langle \cdot, \cdot \rangle_v$ weighted by $v(\xi) := \mathbb{I}\{[0, L)\}(\xi)/L$, returning parameters $\{\tau_i, \omega_i, \phi_i\}$ and code weights μ_i. We first find the normalized code product (Eq. 10) in closed form. Substituting in Eq. 12 gives

$$A(r(\xi), h(\xi;\omega,\tau,\phi)\}) = \frac{\langle r_i(\xi), \cos(\omega\xi+\phi)e^{\tau\xi}\rangle_v}{\|\cos(\omega\xi+\phi)e^{\tau\xi}\|_v}. \tag{13}$$

The numerator is simply Eq. 1. Applying Euler identities gives the denominator

$$\begin{aligned}&\|\cos(\omega\xi+\phi)e^{-\tau\xi}\|_v^2\\ &= \frac{1}{2}\int_0^L (1+\cos(2\omega\xi+2\phi))e^{-2\tau\xi}\mathrm{d}\xi\\ &= \left.\frac{e^{-2\xi\tau}}{2}\frac{(\omega\sin(2\xi\omega+2\phi)-\tau\cos(2\xi\omega+2\phi))}{4\tau^2+4\omega^2}\right|_{\xi=0}^{\xi=L} + \frac{1-e^{-2L\tau}}{4\tau}\end{aligned} \tag{14}$$

Combining Eq. 1 and Eq. 14 gives a closed form normalized code product (Eq. 13), from which we can explicitly calculate gradients in τ, ω, ϕ as desired. Note that although the original signal is discrete, our decomposition is a continuous near-interpolant for it.

From these decompositions we construct the dictionary

$$\mathfrak{M} := \{\hat{G}_\gamma(\rho\xi) : \gamma \in (1,\ldots,D), \rho \in \mathbb{R}^+\}. \tag{15}$$

4.4 Synthesizing the mosaic

In the second, inference, stage we construct a mosaic $\hat{f}_0$ given a target f_0. Here we match the discrete autocorrelogram $F_0 := \mathcal{A}\{f_0\}$ by a second OMP decomposition $\hat{F}_0 := \mathrm{OMP}_{\mathfrak{M},J}(F_0)$, into

$$\hat{F}_0(\xi) := \sum_{j=1}^{J} \kappa_j \hat{G}_{\gamma_j}(\rho_j\xi) \tag{16}$$

for index parameters $\{\gamma_i, \rho_i\}$ and weights κ_i. The OMP has already been introduced, but we pause to verify that it may be applied to this new context. Since each $\hat{G}_{\gamma_j}$ is a linear combination of decaying sinusoids (Eq. 12), the normalizing denominator of the code product (Eq. 10) is again a linear combination of decaying sinusoids, so its integral has a (lengthy) closed form as a linear combination of integrals (Eq. 14), and we can find an explicit gradient $\nabla_\rho A(r_k, \rho)$. Thus we may find $\hat{F}_0$ as required.

Now we wish to construct $\hat{f}_0$ (Eq. 2) such that

$$\mathbb{E}[\mathcal{A}\{\hat{f}_0\}] = \hat{F}_0. \tag{17}$$

Choosing $\hat{f}_0 := \sum_j S_j\alpha_j g_{\gamma_j}(\rho_j t)$ by matching pursuit, simulating S_j independent Rademacher variates, and applying Eqns. 5, 6, 7 to Eq. 2, we find

$$\begin{aligned}\mathbb{E}[\mathcal{A}\{\hat{f}_0\}] &= \mathbb{E}\left[\mathcal{A}\left\{\sum_j S_j\alpha_j g_{\gamma_j}(\rho_j t)\right\}(\xi)\right]\\ &= \sum_j \frac{\alpha_j^2}{\rho_j}\mathcal{A}\left\{g_{\gamma_j}(t)\right\}(\rho_j\xi)\\ &\simeq \sum_j \frac{\alpha_j^2}{\rho_j}\hat{G}_{\gamma_j}(\rho_j\xi).\end{aligned} \tag{18}$$

By inspection,

$$\alpha_j = S_j\sqrt{|\rho_j||\kappa_j|} \tag{19}$$

satisfies Eq. 17. We resample the original discrete dictionary codes to target time scale ρ_i by band-limited sinc interpolation [39]. Finally, we substitute the resulting α_j into Eq. 2 and superpose grains to realize the desired mosaic.

4.5 Localized matching

So far we have discussed entire signals, implicitly assuming them to be brief. The autocorrelogram, taken globally over a long signal such as an entire musical piece, no longer estimates the local, stylistic characteristics. Just as one adapts the discrete Fourier transform for long signals into the Short-Time Fourier Transform (STFT) [4], so do we adapt the autocorrelogram mosaic method, applying it locally. A simple localization is to slice signals into short frames of fixed duration M, which are called *grains* by convention. As in the STFT, we multiply each frame point-wise with real window function w, supported on $[0, M]$ with $\|w\| = 1$. Hereafter, we assume a *sine window*, $w(t) := 2\sin(\pi t/M)\mathbb{I}[0, M]/M$. We fix hop length $H < M$. Next, we localize $\mathfrak{G}$ into a new dictionary whose codes are precisely these time-shifted grains (disallowing zero-energy grains).

$$\mathfrak{G}^{w,H} := \{w(t)g(t-\phi) : g \in \mathfrak{G}, \phi/H \in \mathbb{Z}, \|g'\| > 0\}. \tag{20}$$

In musical material a localized dictionary tends to high redundancy and marginal return on search effort decreases. Rather than proceeding exhaustively, we keep the search tractable by searching a pseudorandom subset of fixed size, where the size of this pseudorandom subset is a user selectable parameter.

In the synthesis stage, we localize the target signal, $f_0^w(t;\phi) := w(t)f_0(t-\phi)$, constructing a local mosaic $\hat{f}_0^w(t;\phi)$ from $\mathfrak{G}^{w,H}$ for $\phi = 0, H, 2H, \ldots$ Finally, we superpose the local mosaics into a global one,

$$\hat{f}_0(t) = \sum_{\ell\in\mathbb{Z}} \hat{f}_0^w(t + H\ell; H\ell). \tag{21}$$

5. EXPERIMENTS

As an initial example we transfer style with target f_0 trumpet solo [2] and source audio a vocal recording. [3] Audio is sampled with a period of $1/44100$s. We fix $M = 8192$, $H = M/2$, $L = 1024$, $C = 4$, $J = 1$, $I = 12$ and reasonable default parameters for the optimization routines. Examining the spectrogram Fig. 2 illustrates phenomena compatible with our claims: In the mosaic we observe local features of the source with the larger structure of the target, to wit, the pitch contours of the trumpet solo with a spectral distribution somewhat like the human voice.

[2] credit Mihai Sorohan
[3] credit Emm Collins

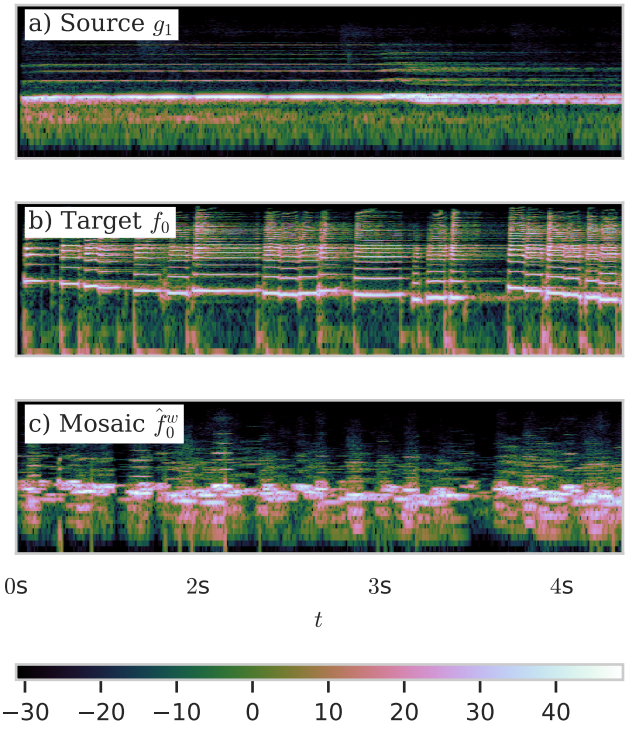


Figure 2. Power spectral density of signals a) source vocal recording b) target trumpet recording and c) resulting mosaic. Frequency increases up vertical axis, intensity in dB with arbitrary normalization.

We next apply the algorithm across a small corpus and compare our results against the mosaicing algorithm NiMFKS [7]. [4] NiMFKS is a useful benchmark for mosaicing synthesis, incorporating many different user-selectable loss functions and decompositions methods from elsewhere in the literature, and possessing openly available code. [5] Their method generalises classical mosaicing by using a non negative PSD factorization to further decompose grains into a sparse product of activations and responses. Unlike our method it does not infer optimal time scaling of audio.

Performance evaluation of mosaicing methods is subjective. In the following, we will nevertheless attempt to describe the behaviors of the two algorithms as objectively as we are able. In order to challenge the NiMFKS model, our corpus samples are tuned to a variety of different root notes, scales and audio ranges, including Indonesian, western and centerless tunings. Style transfer is applied to every pairing of samples. Parameters are left at default values in each algorithm. These may be heard in the supplemental material. Subjectively, neither method seems to produce naturalistic outputs for all pairs of source and target audio. NiMFKS seems ascendant where the source audio is polyphonic and the factorization succeeds at decomposing different notes where our method cannot. On the other hand, where the target tuning is not spanned by the source, the sparse autocorrelogram method is able to produce smoother and better related mosaics by transposing source grains to match the target. Occasionally the sparse autocorrelogram mosaics sound rough during rapid articulations; the method could possibly be improved in these cases by adaptive selection of grain size, or tuning of the free hyperparameters in the model, or extension with non-randomised reconstruction methods. Even in these cases, however, simultaneous playback of the target and the mosaic reveals that we maintain harmonic relationships with the target audio. As such, even this imperfect reconstruction can be regarded as an exotic musical effect. In summary, even at this early stage, our method succeeds in extending mosaic methods to previously intractable tasks, and produces musically interesting output.

[4] It would be instructive to compare against mosaicing method Music Retiler [1], which claims to handle time scaling of audio via a different method, should the source code become available.

[5] `https://code.soundsoftware.ac.uk/projects/nimfks`

6. CONCLUSION

By combining autocorrelogram feature maps and interpolating matching pursuit, we have extended the library of methods of audio mosaicing style transfer. Our method in isolation produces interesting results on the sample data with little tuning. Work remains to be done in analysing the robustness and generality of the method, and selecting optimal tradeoff of cost and quality of different style transfer tasks under different choices of user parameters. More work also remains to be done in integrating this method with existing ones. The flexible loss function of, for example, NiMFKS could be augmented to include autocorrelogram features, and the autocorrelogram approach can be applied to spectrally decomposed signals, which are still audio signals. However, the ease with which we produce good results suggests that further extensions and refinements are worthy of pursuit.

7. ACKNOWLEDGEMENTS

We are indebted to Graham Coleman, James Nichols, the Dadabots and MTF Stockholm for illuminating conversations that greatly improved this work.

8. REFERENCES

[1] H. F. Aarabi and G. Peeters. Music Retiler: Using NMF2D Source Separation for Audio Mosaicing. In *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion*, AM'18, pages 27:1–27:7, New York, NY, USA, 2018. ACM.

[2] H. Barkhuijsen, R. de Beer, W. M. J. Bovée, and D. van Ormondt. Retrieval of frequencies, amplitudes, damping factors, and phases from time-domain signals using a linear least-squares procedure. *Journal of Magnetic Resonance (1969)*, 61(3):465–481, Feb. 1985.

[3] G. M. Bidelman and A. Krishnan. Neural correlates of consonance, dissonance, and the hierarchy of musical

pitch in the human brainstem. *Journal of Neuroscience*, 29(42):13165–13171, Oct. 2009.

[4] R. B. Blackman and J. W. Tukey. *The Measurement of Power Spectra from the Point of View of Communications Engineering*. Dover Publications, New York, 1959.

[5] B. P. Bogert, M. J. R. Healy, and J. W. Tukey. The quefrency alanysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking. In *Symposium on Time Series Analysis*, pages 209–243, 1963.

[6] J. C. Brown. Calculation of a constant Q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, Jan. 1991.

[7] M. Buch, E. Quinton, and B. L. Sturm. NichtnegativeMatrixFaktorisierungnutzendesKlangsynthesenSystem (NiMFKS): Extensions of NMF-based concatenative sound synthesis. In *Proceedings of the 20th International Conference on Digital Audio Effects*, page 7, Edinburgh, 2017.

[8] M. Caetano and X. Rodet. Musical Instrument Sound Morphing Guided by Perceptually Motivated Features. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(8):1666–1675, Aug. 2013.

[9] P. A. Cariani and B. Delgutte. Neural correlates of the pitch of complex tones. I. Pitch and pitch salience. *Journal of neurophysiology*, 76(3):1698–1716, Sept. 1996.

[10] D. Chazan and R. Hoory. Feature-domain concatenative speech synthesis, Apr. 2006.

[11] G. Coleman and J. Bonada. Sound transformation by descriptor using an analytic domain. In *Proceedings of the 11th Int. Conference on Digital Audio Effects (DAFx-08), Espoo, Finland, September 1-4, 2008*, page 7, 2008.

[12] G. Coleman, E. Maestre, J. Bonada, E. Maestre, and J. Bonada. Augmenting sound mosaicing with descriptor-driven transformation. In *Proceedings of DAFx-10*, page 4, 2010.

[13] N. Collins and B. L. Sturm. Sound cross-synthesis and morphing using dictionary-based methods. In *International Computer Music Conference*, 2011.

[14] G. M. Davis, S. G. Mallat, and Z. Zhang. Adaptive time-frequency decompositions. *Optical Engineering*, 33(7):2183–2191, 1994.

[15] J. Driedger and M. Müller. A Review of Time-Scale Modification of Music Signals. *Applied Sciences*, 6(2):57, Feb. 2016.

[16] J. Driedger and T. Pratzlich. Let It Bee – Towards NMF-Inspired Audio Mosaicing. In *Proceedings of ISMIR*, page 7, Malaga, 2015.

[17] H. Dudley. Thirty Years of Vocoder Research. *The Journal of the Acoustical Society of America*, 36(5):1021–1021, May 1964.

[18] A. Elowsson and A. Friberg. Long-term average spectrum in popular music and its relation to the level of the percussion. In *Audio Engineering Society Convention 142*, page 13. Audio Engineering Society, 2017.

[19] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. In *PMLR*, July 2017.

[20] M. Goodwin. Matching pursuit with damped sinusoids. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 2037–2040, Munich, Germany, 1997. IEEE.

[21] E. Grinstein, N. Duong, A. Ozerov, and P. Perez. Audio style transfer. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 586–590, Oct. 2017.

[22] M. Hoffman and P. R. Cook. Feature-Based Synthesis: A Tool for Evaluating, Designing, and Interacting with Music IR Systems. In *Proceedings of ISMIR*, page 2, 2006.

[23] M. D. Hoffman, P. R. Cook, and D. M. Blei. Bayesian spectral matching: Turning Young MC into MC Hammer via MCMC sampling. In *ICMC*, 2009.

[24] K. Jaganathan, Y. C. Eldar, and B. Hassibi. Phase Retrieval: An Overview of Recent Developments. *arXiv:1510.07713 [cs, math]*, Oct. 2015.

[25] G. Langner. Periodicity coding in the auditory system. *Hearing Research*, 60(2):115–142, July 1992.

[26] J. C. R. Licklider. A duplex theory of pitch perception. *Experientia*, 7(4):128–134, Apr. 1951.

[27] P. Mermelstein and C. Chen. Distance measures for speech recognition: Psychological and instrumental. In *Pattern Recognition and Artificial Intelligence,*, volume 101, pages 374–388. Academic Press, 1976.

[28] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, pages 40–44 vol.1, Nov. 1993.

[29] R. Prony. Essai éxperimental et analytique: Sur les lois de la dilatabilité de fluides élastique et sur celles de la force expansive de la vapeur de l'alkool, à différentes températures. *Journal de l'École Polytechnique Floréal et Plairial*, 2, 1795.

[30] L. Rabiner. On the use of autocorrelation analysis for pitch detection. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 25(1):24–33, Feb. 1977.

[31] C. Roads. *Microsound*. The MIT Press, Cambridge, Mass., Aug. 2004.

[32] D. Schwarz. State of the art in sound texture synthesis. In *Proceedings of DAFx-11*, pages 221–231, 2011.

[33] X. Serra and J. Smith. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):12–24, 1990.

[34] Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, and M. Segev. Phase Retrieval with Application to Optical Imaging: A contemporary overview. *IEEE Signal Processing Magazine*, 32(3):87–109, May 2015.

[35] I. Simon, S. Basu, D. Salesin, and M. Agrawala. Audio analogies: Creating new music from an existing performance by concatenative synthesis. In *Proceedings of the 2005 International Computer Music Conference*, pages 65–72, 2005.

[36] M. Slaney, M. Covell, and B. Lassiter. Automatic Audio Morphing. In *Proceedings of the Acoustics, Speech, and Signal Processing, 1996. On Conference Proceedings., 1996 IEEE International Conference - Volume 02*, volume 2 of *ICASSP '96*, pages 1001–1004, Washington, DC, USA, 1996. IEEE Computer Society.

[37] M. Slaney and R. F. Lyon. A perceptual pitch detector. In *Proceedings of ICASSP*, pages 357–360 vol.1, Apr. 1990.

[38] M. Slaney, D. Naar, and R. Lyon. Auditory model inversion for sound separation. In *Proceedings of ICASSP '94.*, volume ii, pages II/77–II/80, Adelaide, SA, Australia, 1994. IEEE.

[39] J. O. Smith. Digital audio resampling home page. Technical report, Center for Computer Research in Music and Acoustics (CCRMA), Stanford University, Jan. 2018.

[40] M. Sondhi. New methods of pitch extraction. *IEEE Transactions on Audio and Electroacoustics*, 16(2):262–266, June 1968.

[41] B. L. Sturm, C. Roads, A. McLeran, and J. J. Shynk. Analysis, visualization, and transformation of audio signals using dictionary-based methods. *Journal of New Music Research*, 38(4):325–341, Dec. 2009.

[42] W. Verhelst and M. Roelands. An Overlap-add Technique Based on Waveform Similarity (WSOLA) for High Quality Time-scale Modification of Speech. In *Proceedings of ICASSP*, ICASSP'93, pages 554–557, Washington, DC, USA, 1993. IEEE Computer Society.

[43] P. Verma and J. O. Smith. Neural style transfer for audio spectograms. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Jan. 2018.

[44] N. Wiener. Generalized harmonic analysis. *Acta Mathematica*, 55:117–258, 1930.

[45] A. Zils and F. Pachet. Musical mosaicing. In *Proceedings of DAFx-01*, volume 2, page 135, Limerick, Ireland, 2001.

AUTOMATIC CHOREOGRAPHY GENERATION WITH CONVOLUTIONAL ENCODER-DECODER NETWORK

Juheon Lee **Seohyun Kim** **Kyogu Lee**
Music & Audio Research Group, Seoul National University , Korea
{juheon2, shzkim, kglee}@snu.ac.kr

ABSTRACT

Automatic choreography generation is a challenging task because it often requires an understanding of two abstract concepts - music and dance - which are realized in the two different modalities, namely audio and video, respectively. In this paper, we propose a music-driven choreography generation system using an auto-regressive encoder-decoder network. To this end, we first collected a set of multimedia clips that include both music and corresponding dance motion. We then extract the joint coordinates of the dancer from video and the mel-spectrogram of music from audio and train our network using music-choreography pairs as input. Finally, a novel dance motion is generated at the inference time when only music is given as an input. We performed a user study for a qualitative evaluation of the proposed method, and the results show that the proposed model is able to generate musically meaningful and natural dance movements given an unheard song. We also revealed through quantitative evaluation that the network has created a movement that correlates with the beat of music.

1. INTRODUCTION

Choreography is a kind of art that designs a series of movements. In particular, in performing art, choreography extends to the use of human bodies to express movements, and these are often performed with music. The choreography suitable for music has significance in that it is not only an artwork itself, but also maximizes the expression of music [4, 7]. For this reason, choreography has become an essential element in many pop music works in recent years. Therefore, the process of creating choreography for music is also considered to be important, and research on a system capable of automatically generating choreography is actively conducted. However, automatic choreography generation is a challenging task because both music and dance are abstract art concepts, and the clear relationship between the two concepts is also not defined by established rules.

In this paper, we proposed a music-driven choreography generation system. In order to model the relationship between music and movement, two concepts in different domains, we firstly designed an autoregressive sequence to sequence model based on neural network that has been actively studied recently. Sequence used as input is time series data with strong correlation between adjacent timestep. Therefore, we designed a network of causal-dilation convolutional layers to fully reflect the information in the adjacent frame. We also applied local conditioning methods to the network to ensure that information related to music is effectively conditioned in the process of creating choreography movements. To evaluate whether a trained network actually produces a dance motion that matches music, we conducted a user study that evaluated naturalness by comparing video that matched random choreography with music and video generated by the proposed network. We also proposed a comparison of the two sequences' auto-correlations to analyze whether the choreography actually reflects music. As a result, we confirmed that the proposed network produced choreography that better reflected music than randomly matched videos, and that the joint movements of the generated choreography had a periodicity similar to the tempo of the music.

The contribution of this paper is as follows: First, we designed a music driven choreography generation network trained by an end-to-end method. Second, to generate choreography reflecting music, we successfully applied a local conditioning method used in speech synthesis field. Third, for the task of creating a choreography that is relatively difficult to assess quantitatively, we proposed the evaluation method using auto-correlation and user evaluation.

The rest of the paper is organized as follows. Studies related to this paper are introduced in Section 2. In Section 3, we explain in detail our proposed method for choreography generation based on the encoder-decoder network. We describe the dataset for experiments and the training process in Section 4. The evaluation scheme and the results are presented in Section 5, followed by conclusions and directions for future work in Section 6.

2. RELATED WORK

Recent advances in machine learning and deep learning techniques have led to a variety of attempts to study the relationship between dance and music. Lee et al. proposed

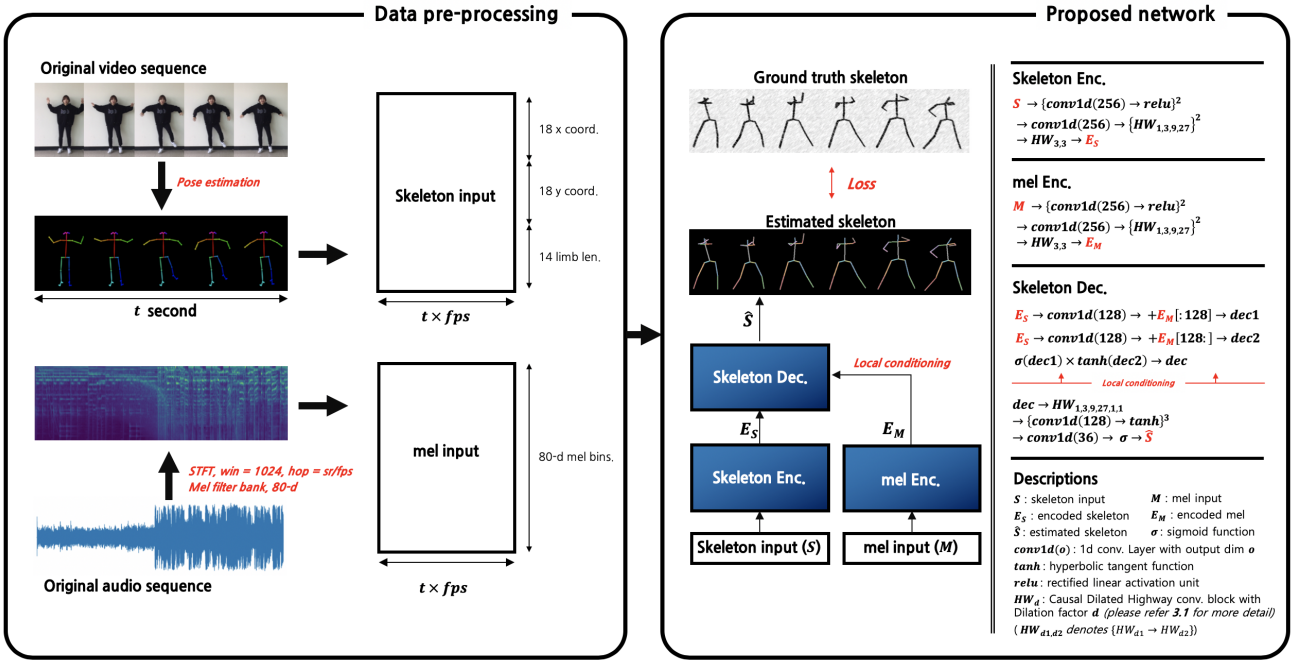


Figure 1. A schematic diagram of the proposed music-driven choreography generation system.

a choreography generation algorithm that retrieves the motions corresponding to the most similar pieces of music in the predefined motion-music-paired database for a given new music segment. [8]. This method selects dance motion from a predefined database, so choreography retrieved with high correlation with music is guaranteed. However, it has limitations in that it can not create novel dance movements that are not included in the database. Ofil et al. proposed a HMM-based model that categorizes the genre of music based on the Mel-Frequency Cepstrum Coefficients (MFCC) [10] feature and generates matching choreography based on the results [11]. But since the choreography is determined by the categorical value obtained through the genre classifier, there is a limit to generate a novel choreography. Omid et al. proposed a music-driven choreography model named Groovenet [1]. They used pairs of music and three-dimensional motion data to train the Factored Conditional Restricted Boltzmann Machines (FCRBM) [14]. They attempted to directly train the relationship between music and dance by using the mel-spectrogram in the training process. However, they reported that their model created awkward dance moves for unheard song, so they conclude that the model was overfitted and the dance moves according to music were not generalized enough.

Lee et al.'s and Ofil et al.'s studies have a limitation in that they can not create novel choreography because the former synthesizes motion by reusing the choreographic samples in a predefined database, and the latter creates choreography only for music input categorized by its genre. Omid et al. did succeed to create novel dance motions, but failed to yield good results mainly due to insufficient training data of merely 23 minutes.

In this study, we proposed a music-driven choreography generation system that can produce novel and natural choreography. [1] In order to secure the novelty of choreography, we used the method of creating choreography with frame by frame generation, not the method of retrieve in the pre-defined dataset. Also, to train the network with sufficient data, we also proposed a way to use the choreography-music data pairs that can be easily obtained from online video sharing community as training data. Finally, in order to conduct effective conditioning of music information, we have applied the methods used in other conditional sequence generation tasks effectively to our task.

3. PROPOSED APPROACH

In this chapter we explain the detailed structure of the proposed network. An overview of the proposed system is illustrated in Figure 1.

In order to learn the relationship between the time-series data of two different modalities, i.e., music and dance, we need a model that performs multi-modal sequence-to-sequence transformations. Also, since the choreographic movement at a certain time-step has a strong correlation with the information at the previous time-step, we should consider a system that provides sufficient reference to the information at the adjacent time-step. From this point of view, we have noted a text-to-speech system that shows reliable performance in a similar environment to these conditions, and then designed our system, inspired by the DCTTS [13] model, which is known to be capable of efficient text-to-speech training.

Our proposed model takes skeleton input S and mel input M as input, to predict skeleton $\hat{S}$ in the next time step:

[1] The generated result can be found at: listentodance.strikingly.com.

$$\hat{S}_{1:T} = CG(S_{0:T-1}, M_{1:T}) \quad (1)$$

where CG denotes our proposed model, choreography generator. For this purpose, each input is encoded via two encoder first. Then the encoded skeleton E_s passes through the decoder and predicts the next time step's skeleton $\hat{S}$, which utilizes the encoded E_M as conditioned information.

3.1 Causal Dilated Highway Conv. Block

In this section we explain the Causal Dilated Highway Convolution Block, one of the core structures of the proposed network. The choreography, which is basically the object that we should create, has a strong correlation with the information of the adjacent time-step. Therefore, in order to predict movement in the next time-step, information from previous time-steps should be fully consulted. Also, for choreography, a wide range of historical information should be referred to because it has relatively long-term dependency. To this end, we use the Causal Dilated Convolution. *Causal* means that only the input data from time 0 to $t-1$ can be referred to when calculating the output at time t. We used a causal convolution layer because our network must be an auto-regressive model to generate the next frame that is not yet known from the preceding frames. In addition, we used the *dilated convolution* proposed in the Wavenet [16] to ensure that the model has a wider receptive field. Finally, to enable efficient training even in deep model structures, we used a *highway network* architecture [12] where gated function could be trained. That is, the output of the CDHC block is calculated as:

$$\textbf{output} = tanh(\text{H1}) \cdot relu(\text{H2}) + (1 - tanh(\text{H1})) \cdot \textbf{input} \quad (2)$$

where [H1, H2] is the tensor calculated through the causal dilated convolution layer of the input tensor. The output channel of this convolution layer is twice the input channel, and the kernel size is 3.

3.2 Encoder & Decoder structure

To predict the next time-step skeleton information from the given input information, we used a method of effectively encoding input information and then combining them to decode. To this end, we designed two encoder and one decoder with CDHC block. Both the skeleton encoders and the audio encoders all consist of three convolution layers and 10 CDHC blocks. The first convolution layer of each encoder increases the input channel to 256 dimensions, and the other two layers perform 1x1 convolution. Thereafter, the output values from last convolutional layer are connected in sequence to 10 CDHC blocks with a dilation factor of (1,3,9,27,1,3,9,27,3,3), and the corresponding operations result in audio and skeleton data are encoded to have a sufficiently wide receptive field to reflect sufficient past information.

A decoder is a network that generates skeleton data for the next frame from an encoded skeleton and an encoded audio. To do this, the encoded skeleton input to the decoder is combined with the encoded audio in the following:

$$Dec1 = conv1d(E_S) + E_M[:128] \quad (3)$$

$$Dec2 = conv1d(E_S) + E_M[128:] \quad (4)$$

$$Dec = \sigma(Dec1) \times tanh(Dec2) \quad (5)$$

Where E_S and E_M refer to the encoded skeleton and encoded audio, respectively, and $conv1d$ means the convolution layer with an output channel of 128 and a kernel size of 1. The combined Dec tensor then goes through six CDHC blocks with a dilation factor of (1,3,9,27,3,3) and then through three 128-channel convolutional layers with a $tanh$ activation function. Finally, after passing through a convolution layer with the same output channel as the dimension of the target, the final decoder output is obtained via $sigmoid$ activation.

3.3 Proposed network

This network receives skeleton and music data from time 0 to $t-1$ as input. Both data are encoded via encoders and combined at the beginning of the decoder. The final output of the decoder is compared with the ground truth motion data at time 1 to t and we used it as a $L1$ loss. Since all convolution operations included in the network are with kernel size 1 or causal operations, the k-th value of output refers to only the 0 to $k-1$ time step of the input during the operation. Therefore, the model satisfies the causal condition.

4. EXPERIMENT

4.1 Data

We have collected 100 YouTube choreography videos and corresponding audios. The genre was selected mainly for K-pop dance, and the total length of collected data was 6.26 hours. We divided 85 songs into train sets, 5 songs into valid sets, and 10 other songs into test sets, to train and evaluate the proposed network.

4.1.1 Skeleton data

We extracted the x, y coordinates of 15 human body joints from each frame using the Openpose algorithm [3] from the collected video as shown in Fig. 2. Next, we min-max normalize the extracted coordinate values for each video, and use the linear-interpolation for the unrecognized coordinate values.

Since we can not measure the exact 3d angle between the human body limbs using the 2d joint coordinate, we used the absolute coordinates values of each point as the training target. However, in this case, the length of each limb in the projected skeleton can vary, and awkward motion can be generated if the model learns it incorrectly. So we additionally calculated the lengths of the 14 main limbs together and added a loss to compare with the limb length of the skeleton that the model generated. Therefore, the x, y coordinates of the total 15 joints, and the total of 14 main limb length are used as skeleton data.

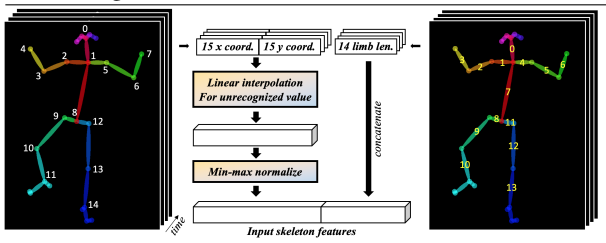


Figure 2. The process of extracting skeleton data from video frames.

4.1.2 Music data

We separated the audio contained in the collected video and used it as music data. The mel-spectrogram was extracted from the audio waveform with the window size of 1024 samples, and 80 mel-frequency bins. Because we need time-aligned audio-video pairs for training, we adjusted the hop size when extracting the mel-spectrogram so that audio and video data end up with the same frame rate.

4.2 Training

We have trained the proposed network that creates the next skeleton coordinate for a given previous skeleton sequence and music sequence. To do this, we first input skeleton data and music data from 0 to t-1 frames. Then, the output of the network is compared with the ground truth choreographic data corresponding to 1 to t frame by use L1 loss as a cost function. In addition, we calculated the length of each limb from the skeleton data of the generated frame, and compared with the actual ground truth length through the L1 loss.

We used the adam optimizer [6], with $\beta_1 = 0.5$, $\beta_2 = 0.9$, for training and set the learning rate to 0.0002. At every iteration, we used a video-audio pair that was cut in 500 frames for training, and it contains about 20 seconds of choreography and music information. The length of the input sample was set to 500 frame because we decided that the sequence of lengths, which fully reflected meaningful levels of behavior in the choreography, should be used for training. We set the batch size to 16, and then we finished the training after proceeding with a total of 30,000 iterations. We trained our network with one GEFORCE GTX 1080 ti GPU for three days.

4.3 Inference

The choreography inference process is performed in an auto-regressive manner different from training. That is, the initial position of each joint is given as an input skeleton frame, and at the same time, the first frame of mel-spectrogram is input to the trained model. When inference is performed once, estimated skeleton at $t = 1$ is output. Then we concatenate skeleton at $t = 0$ and $t = 1$, then input them back into the model with mel-spectrogram at $t = 0$ and $t = 1$. After than, we get estimated skeleton at $t = 1$ and $t = 2$. Therefore, we can generate the choreography by repeating the above process for the length of music input, and used it to evaluate the generated choreography.

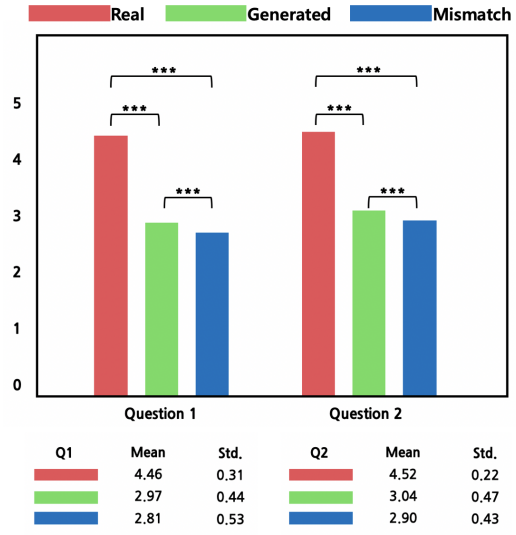


Figure 3. Average Likert-scale user scores on two questions (Q1: Is choreography natural? / Q2: Does choreography fit well with music?). The table below the graph indicates the mean and variance of responses by model for each question. The p-values for pairwise comparisons between the groups are also shown at the top. ***: $p < 0.001$; **: $p < 0.01$.

5. EVALUATION & RESULTS

5.1 User study

We conducted a user study to evaluate whether the generated choreography was natural and whether it was produced in accordance with the music. First, we generated 20 videos for each of the three groups: *Real*, *Generated*, and *Mismatch*. Group *Real* consists of music A_i and actual choreography for music A_i. Group *Generated* consists of music B_i and novel choreography generated by our model given music B_i. Finally, the group *Mismatch* consists of music C_i and novel choreography generated by our model but with randomly selected music rather than C_i. Music A_i, B_i, and C_i were randomly selected among the songs included in the validation dataset that was not used in training, and the length of each audio/video was 16 seconds.

After mixing the three groups of videos in a random order, we asked the participants whether each video's choreography is natural (Question 1) and whether it fits well with music (Question 2), and to give a score in a Likert scale [2]. After collecting the responses, we performed isoquantity and normality tests using data averaging 20 responses from each group, to see if there was a difference in the mean of the responses of the groups. After evaluating significance through repeated-measure ANOVA test , further post-hoc paired t-test analysis was performed to calculate the p-value, and the difference between the groups was examined [5].

A total of 33 participants answered the questionnaire

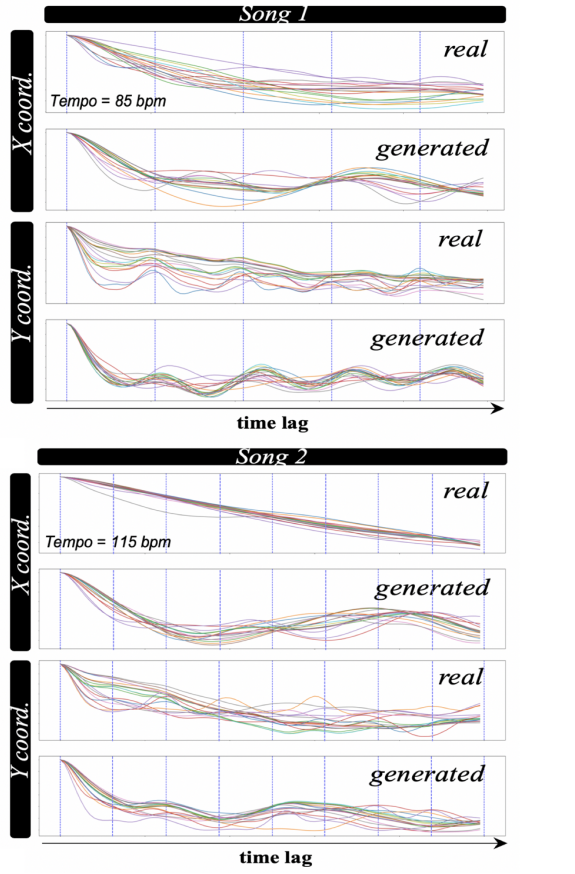


Figure 4. Autocorrelation of the x,y coordinates of each joint from real and generated choreography for two songs. The x-axis of each graph represents the time lag, y-axis of each graph represents the autocorrelation, and the blue vertical lines represent the beat positions of each song.

and the results are shown in the Figure 3. The results of the statistical tests confirmed that the mean scores between the three groups were significantly different for both questions. Average user score for both questions were highest in *Real* group and lowest in *Mismatch* group. It is clear that the *Real* group score is the highest, because it is made up of the choreography created by the human. The average score of the *Generated* group surpassed the *Mismatch* group in both questions. If the proposed model generates choreography that is not associated with music, participants will have a similar response, regardless of what music is played with the generated choreography. However, from the fact that the video received a significantly higher score when played with the music used in choreography generation, we judged that the proposed model produced choreography that listen and reflects the music.

5.2 Autocorrelation Analysis

We also performed an autocorrelation analysis to further investigate the differences between the generated choreography and the actual choreography. Autocorrelation is a correlation between a given sequence with itself, reflecting the periodic properties of the sequence. We can identify the periodic component of a given sequence through the location of the peaks observed in the autocorrelation results. Using this, we analyzed the motion by calculating the autocorrelation on the x, y coordinates of the choreography movement and compared it with the tempo of corresponding music. Our hypothesis was that if the model can produce dance by listening to the music, the autocorrelation peak position of the motion will appear at the same point as the beat of the music.

Fig. 4 shows the autocorrelation results of two choreography samples along with the tempo of corresponding music. In actual choreography, a clear peak is observed in y-direction movement, but not in x-direction movement. This tendency is also observed in the generated choreography. From this we can determine that the proposed network has learned the periodic tendency of the real choreography used in training. Also, In actual choreography, the first or second peak of the y-direction auto-correlation appears at the same position as the music beat. This means that music and choreography have similar periodic properties. This tendency can be confirmed also in the case of the generated sample. From this, it is judged that the proposed model has generated the choreography that listen the music and reflects its periodic nature.

6. CONCLUSION

In this study, we proposed an auto-regressive encoder-decoder network that generates matching choreography for a given music input. We used audio-video pairs data obtained from YouTube for training. As a result, it was found that motions matching with the music were generated through comparison of user study and autocorrelation analysis. This study has a significance in that it shows a significant performance in the area of learning-based choreography generation, in which sufficient performance has not been secured yet. Also, it is meaningful not only to learn the movement of dance but also to use the relationship with music together for generation.

Although we found in this study that the choreography generated compared to the mismatch group has a higher correlation with music at a significant level, we still have the limitation of having a large difference score from the real group. To overcome this, we will further model the correlation between movement and music more elaborately and carry out follow-up studies that reflect it in the network architecture. Also, this research has limitations that generated choreography reflects only the periodicity among various properties of music. Ultimately, it is necessary to create appropriate choreography according to various genres, moods, and contexts of music as well as periodicity. In order to do this, we plan to establish data sets that satisfy various conditions and carry out further research. In addition, we use 2-d skeleton position for training, and it is difficult to use this type of data in case of needing actual implementation such as a robot. Therefore, the extension of the model to 3-d choreography generation using the improved 3-d pose estimation algorithm [9, 15, 17] is also a future research topic.

7. ACKNOWLEDGEMENTS

This work was supported partly by LG Electronics and partly by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2017M3C4A7078548).

8. REFERENCES

[1] Omid Alemi, Jules Françoise, and Philippe Pasquier. Groovenet: Real-time music-driven dance movement generation using artificial neural networks. *networks*, 8(17):26, 2017.

[2] I Elaine Allen and Christopher A Seaman. Likert scales and data analyses. *Quality progress*, 40(7):64–65, 2007.

[3] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[4] Sommer Gentry and Eric Feron. Modeling musically meaningful choreography. In *Systems, man and cybernetics, 2004 IEEE international conference on*, volume 4, pages 3880–3885. IEEE, 2004.

[5] Ellen R Girden. *ANOVA: Repeated measures*. Number 84. Sage, 1992.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Carol L Krumhansl and Diana Lynn Schenck. Can dance reflect the structural and expressive qualities of music? a perceptual experiment on balanchine's choreography of mozart's divertimento no. 15. *Musicae Scientiae*, 1(1):63–85, 1997.

[8] Minho Lee, Kyogu Lee, and Jaeheung Park. Music similarity-based approach to generating dance motion sequence. *Multimedia tools and applications*, 62(3):895–912, 2013.

[9] Diogo C Luvizon, David Picard, and Hedi Tabia. 2d/3d pose estimation and action recognition using multitask deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5137–5146, 2018.

[10] Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *arXiv preprint arXiv:1003.4083*, 2010.

[11] Ferda Ofli, Yasemin Demir, Yücel Yemez, Engin Erzin, A Murat Tekalp, Koray Balcı, İdil Kızoğlu, Lale Akarun, Cristian Canton-Ferrer, Joëlle Tilmanne, et al. An audio-driven dancing avatar. *Journal on Multimodal User Interfaces*, 2(2):93–103, 2008.

[12] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[13] Hideyuki Tachibana, Katsuya Uenoyama, and Shunsuke Aihara. Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. *arXiv preprint arXiv:1710.08969*, 2017.

[14] Graham W Taylor and Geoffrey E Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *Proceedings of the 26th annual international conference on machine learning*, pages 1025–1032. ACM, 2009.

[15] Denis Tome, Chris Russell, and Lourdes Agapito. Lifting from the deep: Convolutional 3d pose estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2500–2509, 2017.

[16] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *SSW*, page 125, 2016.

[17] Wei Yang, Wanli Ouyang, Xiaolong Wang, Jimmy Ren, Hongsheng Li, and Xiaogang Wang. 3d human pose estimation in the wild by adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5255–5264, 2018.

HIERARCHICAL CLASSIFICATION NETWORKS FOR SINGING VOICE SEGMENTATION AND TRANSCRIPTION

Zih-Sing Fu
Dept. EE, National Taiwan University, Taiwan
b04901015@ntu.edu.tw

Li Su
IIS, Academia Sinica, Taiwan
lisu@iis.sinica.edu.tw

ABSTRACT

Identifying the onset and offset time of a note is a challenging step in singing voice transcription, as the soft onset/offset, portamento, and vibrato phenomena are rich in singing voice signals. In this work, we utilize various types of signal representations with deep learning for onset and offset detection of monophonic singing voice. We consider onset and offset detection as a hierarchical classification problem, where every input segment is classified into one of all the possible states in monophonic singing, namely the silence, activation, and transition states,where the transition state is further classified into the onset and offset states. An objective function based on this hierarchical taxonomy nicely guides the model to capture complicated temporal dynamics of note sequences. Multiple input signal representations containing spectral differences and pitch saliency are employed to jointly enhance such temporal patterns. The proposed method implemented with residual networks provides improved performance over prior art in onset and offset detection. Moreover, by integrating with a pitch detection framework, the proposed method also outperforms previous singing voice transcription methods. This result emphasizes the importance of note segmentation in singing voice transcription.

1. INTRODUCTION

Note-level automatic music transcription (AMT) refers to converting a recorded music piece into its symbolic form containing the onset, offset, and pitch of every note [4, 22]. Note-level AMT is still a challenging problem, particularly in the case of singing voice transcription. The soft onset/offset and portamento patterns of singing voice hinder the positioning of onset and offset time in both the detection [8, 29] and the annotation process [10, 15, 19]. However, solving the onset and offset detection problem, or equivalently the *note segmentation* problem, [1] is mandatory in a note-level AMT system. How to improve a note segmentation model efficiently with limited scope of data, and how to incorporate the outcomes of detection into note-level AMT, are both important issues in developing a complete AMT system.

Previous note segmentation works on singing voice usually employ state-space machines such as the hidden Markov models (HMM), which consistently detect onset and offset by characterizing the temporal dynamics among the *states* (attack, sustain, and silence, etc.) of note events [16, 20, 24, 29]. Recently, deep neural networks with objective functions optimized for onset and offset detection have demonstrated excellent performance in note-level AMT [1, 12]. Some architectures such as the convolutional neural network (CNN) do achieve a great advance in modeling note transition by their compelling performance in pattern recognition on a local scale. One example is the CNN-based onset detection method in [25], where the local feature segments with CNN outperforms the temporal models based on the recurrent neural network (RNN) [9].

In this paper, we propose novel signal representations and objective functions in neural network-based singing voice segmentation. we regard onset and offset detection as a hierarchical classification problem that maps input segments/sequences onto our proposed state space, where a generalized hierarchical taxonomy of the states in a note sequence is specified to guide the learning process. Multiple data representations are also used to enhance signal-level expressivity of note transition events. Experiments using either the residual network (ResNet) [13] or the RNN with attention [2] demonstrate the effectiveness of hierarchical classification in note segmentation. Finally, a straightforward integration of the proposed note segmentation method and pitch detection provides improved note transcription performance over prior art.

2. RELATED WORK

The most challenging case of onset detection is arguably singing voice. According to the results from MIREX 2018 audio onset detection task, the best F1-score of singing voice onset detection among all submissions is 61.94%, lower than the best results of other instrument classes by at least 10%. [2] The state-of-the-art onset detection algorithms are based on either RNN [5, 11] or CNN [25]. In [25], the onset detection task is to classify whether the

[1] We refer to note segmentation as temporal segmentation of note objects, which is therefore equivalent to onset and offset detection [7].

[2] More details can be found in: https://nema.lis.illinois.edu/nema_out/mirex2018/results/aod/resultsperclass.html

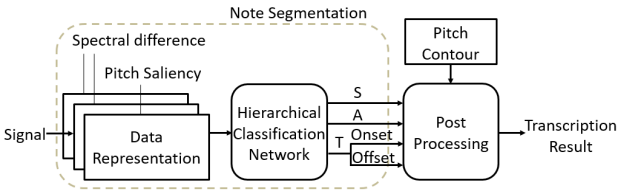


Figure 1: System overview of the proposed note segmentation and transcription framework.

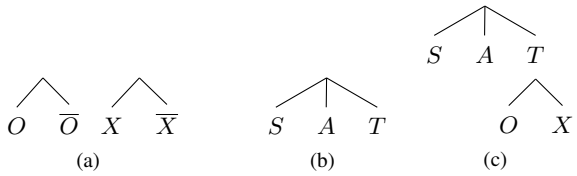


Figure 2: The taxonomy of the proposed models. Every tree represents an objective function, every siblings form a regularization term of the objective function, and every leaf of the tree represents a state label; S, A, O, $\overline{O}$, X, $\overline{X}$, and T represent silence, activation, onset, non-onset, offset, non-offset, and transition, respectively. Different trees therefore represent different optimization approaches: (a) On-Off model. (b) Tri-state model. (c) Hierarchical classification model. See Section 3.1 for more details.

middle of the input is at the onset time, where the inputs are short segments of spectrogram with various resolutions, each as one channel of the CNN. Besides spectrogram, other feature representations such as spectral difference, spectral flux and group-delay function are also widely-used in general-purpose onset detection [14].

Unlike onset detection, offset detection is seldom treated independently and is more often discussed in the context of note-level AMT [1, 3, 12]. The study carried out in [15] focuses on different playing styles of string instruments and summarizes several relevant features, including spectral difference, signal RMS energy, pitch confidence values, and pitch change, etc.

Previous methods in singing voice transcription widely adopt state-space machines to accomplish onset detection, pitch tracking, and offset detection in a single workflow. For example, the Tony software [16] uses an HMM containing three states, namely attack, stable, and silent, to characterize the temporal dynamics of a note sequence. The only allowed transition rules between these states are: 1) from attack to stable, 2) from stable to silent, and 3) from silent to attack of another note. However, these rules are oversimplified from real cases; for instance, an offset event is not always equivalent to a transition into the silent state. Rather, some offset events are followed immediately by the attack state of another consecutive note, which sometimes has the same pitch as the previous one. As a result, consecutive notes are merged and needs to be resolved by post-processing.

Recently developed note-level AMT methods utilizing deep learning has gained tremendous improvement, especially in offset detection. It is notable that in these methods, offset or onset detection sub-modules are optimized with more than one objective functions. Elowsson used two separate networks to learn 1) the offset curve, which outputs one at the instance of note offset, 2) the offset detection activation, which turns from zero to one when a note offset event turns into silence, and combined the results to describe offset events [1]. Hawthorne *et al.* used time-dependent object functions to infer the attack and decay of a musical note. These methods shed light on the note tracking of singing voice [12].

The above discussion inspires us two ways for improving singing voice segmentation. First, the objective functions can be designed to rely not merely on the onset and offset labels, but on an state space that describes all possible state transitions in a note sequence. Second, given the flexibility of neural network models, one may augment all the data representations related to onset/offset into the network to enhance the optimization process. The two ideas will be discussed in Section 3.1 and 3.2 respectively.

3. METHOD

Following previous discussion, we discuss the frame-wise onset and offset detection framework shown in Figure 1: for every time instance t, the hierarchical classifier predicts a set of labels y_t containing onset and offset information from a local feature representation $\mathbf{R}_t$. Note transcription is done by integrating pitch contour information.

3.1 Hierarchical classification for note segmentation

We consider the following states in a note sequence: silence (S), activation (A), and transition (T), where transition is further divided into two states, onset (O) and offset (X). When a transition (i.e. onset or offset) occurs, there are three possible *transition behaviors* of state evolution: $S \to T \to A$ where T represents an onset (O), $A \to T \to S$ where T represents an offset (X), and $A \to T \to A$ where T in this case contains an offset followed immediately by the onset of another note (XO). In other words, there is an important case that *an onset and an offset are presumably overlapped*. This fact motivates us to define such a state space that can encompass more general cases. As a result, there is a hierarchical taxonomy of these states, as shown in Figure 2 (c). See the caption of Figure 2 for more detailed information.

To investigate the behavior of this state space, we introduce several baselines and the proposed hierarchical classification model altogether to highlight the advantage of the proposed model in onset and offset classification.

1) First, we consider the note segmentation model consisting of two independent classifiers, one for onset detection and the other for offset detection. The 2-D onset label $y_{\text{on}} := [O, \overline{O}]$ is one-hot, where O represents the onset state while $\overline{O}$ represents the non-onset state. That means, $y_{\text{on}} = [1, 0]$ for onset and $y_{\text{on}} = [0, 1]$ for non-onset. Similarly, we have the offset label $y_{\text{off}} := [X, \overline{X}]$. Let the prediction of the two networks be $\hat{y}_{\text{on}}$ and $\hat{y}_{\text{off}}$, the model

is optimized by the following two objective functions:

$$L_{\text{on}}(y_{\text{on}}, \hat{y}_{\text{on}}) = \text{BCE}(y_{\text{on}}, \hat{y}_{\text{on}})\,, \quad (1)$$

$$L_{\text{off}}(y_{\text{off}}, \hat{y}_{\text{off}}) = \text{BCE}(y_{\text{off}}, \hat{y}_{\text{off}})\,. \quad (2)$$

where BCE is the binary crossentropy. This model is denoted as the on-off network (OON) model, and its taxonomy is illustrated in Figure 2 (a). Note that one tree represents one objective function, and every siblings form a regularization term in an objective function.

2) The onset and offset detection tasks share the same network, but with two task-specific layers, one for onset and the other for offset. The output label $y := [y_{\text{on}}, y_{\text{off}}]$ therefore has four dimensions. The total loss function is

$$L_{\text{M-OON}}(y, \hat{y}) := \text{BCE}(y_{\text{on}}, \hat{y}_{\text{on}}) + \text{BCE}(y_{\text{off}}, \hat{y}_{\text{off}}) \quad (3)$$

This model is denoted as the merged on-off network (M-OON) model hereafter.

3) The onset and offset are described implicitly by the three output states S, A, and T from a shared network. That means, the network outputs a multi-hot 3-D vector $y_{\text{tri}} := [S, A, T]$, where S, A and T are values between 0 and 1. The total loss function is

$$L_{\text{TSN}}(y, \hat{y}) := \text{BCE}(y_{\text{tri}}, \hat{y}_{\text{tri}}) \quad (4)$$

After obtaining the likelihood of S, T, A at every time instance t, we may follow the *transition behaviors* mentioned above to determine a T state to be an onset or an offset; the details can be found in Section 3.4. This model will be denoted as the tri-state notwork (TSN) model, and its taxonomy tree is constructed following Figure 2 (b).

Note that it is also possible to use categorical crossentropy rather than BCE in (4). However, using BCE allows possible overlapping of different states and therefore more flexibility for the model. Our pilot study also shows that using BCE achieves better performance.

4) We further consider the hierarchical structure that T can be onset, offset or an overlap of onset and offset. The output label is then a six-dimension space $y := [S, A, O, \overline{O}, X, \overline{X}]$, and the total objective function is:

$$L_{\text{HCN1}}(y, \hat{y}) := \text{BCE}(y_{\text{tri}}, \hat{y}_{\text{tri}}) + \text{BCE}(y_{\text{on}}, \hat{y}_{\text{on}}) + \text{BCE}(y_{\text{off}}, \hat{y}_{\text{off}}) \quad (5)$$

where we define the likelihood of the transition state as $T := \max(O, X)$. That means, if one of O or X is higher than a threshold (0.5 in the logistic regression case), then the state will be also predicted as T. The taxonomy tree of this case is illustrated in Figure 2 (c).

Finally, since T is in minority, optimizing the term $\text{BCE}(y_{\text{tri}}, \hat{y}_{\text{tri}})$ would suffer from data imbalance. To mitigate this issue, we enhance the *activity* classification between S and A by adding a new set of labels $y_{\text{act}} := [S, A]$, to enforce the output that only one of S and A would have high likelihood. The total objective function is then

$$L_{\text{HCN2}}(y, \hat{y}) := \text{BCE}(y_{\text{tri}}, \hat{y}_{\text{tri}}) + \text{BCE}(y_{\text{act}}, \hat{y}_{\text{act}}) + \text{BCE}(y_{\text{on}}, \hat{y}_{\text{on}}) + \text{BCE}(y_{\text{off}}, \hat{y}_{\text{off}}) \quad (6)$$

For clarity, (5) is denoted as the hierarchical classification network 1 (HCN1) model and (6) is denoted as the the hierarchical classification network 2 (HCN2) model.

3.2 Data representations

Based on the discussion in [15], we consider the spectral differences and the pitch salience representation in as the input of the proposed model. Given the input audio signal $\mathbf{x} := \mathbf{x}[n]$, where n is the time index. Let the amplitude part of the short-time Fourier transform (STFT) of $\mathbf{x}$ be $\mathbf{X}$. The forward spectral difference $\mathbf{S}^+$ and the backward spectral difference $\mathbf{S}^-$ are the time-forward and the time-backward differences of two neighbouring spectra in $\mathbf{X}$, as shown in the followings:

$$\mathbf{S}^+ = \text{ReLU}\left(\mathbf{X}[k, n+1] - \mathbf{X}[k, n-1]\right), \quad (7)$$

$$\mathbf{S}^- = \text{ReLU}\left(\mathbf{X}[k, n-1] - \mathbf{X}[k, n+1]\right), \quad (8)$$

where $\text{ReLU}(\cdot)$ represents the element-wise rectified linear unit: $\text{ReLU}(x) = x$ if $x > 0$, and 0 otherwise. That means, we split the first-order temporal difference of the spectrogram $\mathbf{X}$ into two channels, one is the part with positive temporal difference, and the other one is with negative temporal difference.

For the pitch saliency feature of $\mathbf{x}$, we adopt the one proposed in the combined frequency and periodicity (CFP) approach, which combines a frequency-domain feature indicating its fundamental frequency (f_0) and harmonics (nf_0), in a time-domain feature revealing its f_0 and sub-harmonics (f_0/n) to form a succinct, localized pitch feature with suppressed harmonic and sub-harmonic peaks [21, 28]. The feature is computed with the following process. Given a DFT matrix $\mathbf{F}$, high-pass filters $\mathbf{W}_f$ and $\mathbf{W}_t$, and activation functions σ_i, we consider three features, namely, spectrogram $\mathbf{Z}_0$, generalized cepstrum (GC) $\mathbf{Z}_1$, and generalized cepstrum of spectrum (GCoS) $\mathbf{Z}_2$:

$$\mathbf{Z}_0[k, n] := \sigma_0\left(\mathbf{W}_f \mathbf{X}\right), \quad (9)$$

$$\mathbf{Z}_1[q, n] := \sigma_1\left(\mathbf{W}_t \mathbf{F}^{-1} \mathbf{Z}_0\right), \quad (10)$$

$$\mathbf{Z}_2[k, n] := \sigma_2\left(\mathbf{W}_f \mathbf{F} \mathbf{Z}_1\right). \quad (11)$$

The index k in $\mathbf{Z}_0$ and $\mathbf{Z}_2$ is frequency, while the index q in $\mathbf{Z}_1$ is called *quefrency*, which has the same unit as time. The nonlinear activation function is defined as a rectified and root-power function $\sigma_i(\mathbf{Z}) = |\text{ReLU}(\mathbf{Z})|^{\gamma_i}$, where $i = 0, 1, 2\cdots$, $0 < \gamma_i \leq 1$, and $|\cdot|^{\gamma_0}$ is an element-wise root function. $\mathbf{W}_f$ and $\mathbf{W}_t$ are two high-pass filters designed as diagonal matrices used to remove slow-varying portions, where $\mathbf{W}_f$ applies cutoff frequency k_c and $\mathbf{W}_t$ applies cutoff quefrency q_c. In this paper we set $k_c = 80$ Hz and $q_c = 1/800$ sec. Based on the CFP approach, unwanted harmonics and sub-harmonics can be suppressed by merging $\mathbf{Z}_1$ and $\mathbf{Z}_2$ together. Note that $\mathbf{Z}_1$ should be mapped into the frequency domain because it is in the quefrency domain. Hence, we apply two sets of filter banks, both of which contain 174 triangular filters ranging from 80 Hz to 1000 Hz and with 48 bands per octave, respectively in the time and frequency domains. More specifically, the mth filter in frequency (or time) takes the weighted sum of the components whose frequency (or period) is between 0.25 semitones above and below the frequency at $f_m = 80 \times 2^{(m-1)/48}$ Hz (or the period at $1/f_m$

seconds). The filtered representations $\tilde{\mathbf{Z}}_1$ and $\tilde{\mathbf{Z}}_2$ are then both in the time-pitch scale. The CFP representation $\mathbf{Z}$ is

$$\mathbf{Z}[p,n] = \tilde{\mathbf{Z}}_1[p,n]\tilde{\mathbf{Z}}_2[p,n]\,, \tag{12}$$

where p is the pitch index. Details and source codes of computing the CFP representations can be found in [27].

In this work, the audio recordings are resampled to 16 kHz and are merged into mono-channel. Following [5], the input features are of multiple resolution.We compute $\mathbf{S}^+$, $\mathbf{S}^-$, and $\mathbf{Z}$ using the Hann window with 3 different sizes of 186, 372, and 743 samples (i.e. 11.61, 23.22, and 46.44 ms), resulting in nine data representation. The hop size is 320 samples (i.e. 20 ms). In CNN, $\mathbf{S}^+$, $\mathbf{S}^-$, and $\mathbf{Z}$ form the three input channels, and in each channel the data representations with three different window sizes are concatenated together. In RNN, all the nine data representations are concatenated as the input.

3.3 Model

We investigate two networks that stand for two strategies in modeling note sequences: ResNet for image classification [13] and RNN with attention for sequence classification [2]. Denote the frame-level feature at the time instance t as $\mathbf{r}_t$. For every t, we take the sequence $\mathbf{R}_t := [\mathbf{r}_{t-k}, \mathbf{r}_{t-k+1}, \cdots, \mathbf{r}_t \cdots, \mathbf{r}_{t+k}]$ as the input of the model to predict the presence of onset and offset at t. We set $k = 9$ according to the optimal loss on the validation set. That means, the dimension of every input $\mathbf{R}_t$ is $(c, 174, 19)$ (for ResNet) or $(c * 174, 19)$ (for RNN with attention mechanism), where c represents the number of channels: if $\mathbf{S}^+$, $\mathbf{S}^-$, and $\mathbf{Z}$ are stacked as the input, then $c = 3$.

Our implementation of the ResNet model basically follows the ResNet-18 architecture in [13]. The network is composed of eight sub-networks, each of which has two convolutional layers. The convolutional layers mostly have kernel of size $(3, 3)$. Batch normalization is used after each convolutional layer. The spatial pooling process is done by using convolutional layers with stride of two. Shortcut paths link the feature maps by skipping every two convolutional layers. After the convolution stages, the feature maps are pooled by averaging, and then are mapped to the output space through fully connected layers. See [13] for the implementation details. The output format and the objective functions follow the discussion in Section 3.1.

The RNN with attention is composed of a three bidirectional long-short-term memory (BLSTM) [26] layers, an attention layer, and two fully connected layers. For the three-layer BLSTM, the dimension of every hidden unit is 150. The outputs of the BLSTM are weighted and summed by the $2k + 1$ attention weights derived from the hidden units of the last BLSTM layer [2]. Layer normalization is used to stabilize training and inference processes. The results are then fed into the two-layer fully-connected network, each with a dimension of 150 and 6. The output format and the objective functions of the model also follows the discussion in Section 3.1.

Each data representation is normalized to zero mean before fed into the model. The manual labels in the dataset are not always exact since the exact time of an onset/offset event is hard to determine [5]. To solve this issue, we extend the labels to a *tolerance window* δ that can allow uncertainty in the onset/offset time labels: if a frame is within $\delta = \pm$50ms to the true label, the label is also set to 1. This δ value is chosen according to the evaluation convention of onset detection in MIREX. This can mitigate the issue of data imbalance. In this work, all the models are obtained after 80 epochs of training on an Nvidia TITAN Xp GPU, using the Adam optimizer with the learning rate of 0.001. The source code, supplementary materials, and listening examples are available at: `https://github.com/Itachi6912110/Hierarchical-Note-Segmentation`.

3.4 Post-processing and note segmentation

We employ a linear filter with impulse response $h(n) = [0.25, 0.5, 1, 0.5, 0.25]$ to smooth the predicted onset and offset sequences. Then we apply a threshold at 0.5 and a peak picking process on the sequences to determine possible onset and offset positions. At this stage, minor mismatches between the predicted onset and offset positions still remain. To ensure that every onset is followed by exactly one offset, additional procedures are used.

For the OON and the M-OON models, the procedure includes: 1) if there are two onsets having no offset between them, we insert an offset at the time when the second onset occurs; 2) if there are two offsets without any onset between them, we directly discard the second one.

For the TSN model, consistent segmentation results can be derived directly from the relationship among S, A and T, so there is no issue on onset/offset mismatching. Onsets and offsets are determined by the following steps: 1) obtain the peak positions of the predicted sequence of T; 2) sum over the likelihood values of S and A in every interval separated by those peaks obtained in 1). If the sum of S is higher than the sum of A, then the interval is determined to be S. Otherwise, the interval is determined to be A; 3) for every selected T in 1), if its left-side interval is S and its right-side interval is A, a $S{\rightarrow}T{\rightarrow}A$ pattern is detected and the transition is determined as an onset. Conversely, if we detect a $A{\rightarrow}T{\rightarrow}S$ pattern, the transition is determined as an offset; 4) if we detect an $A{\rightarrow}T{\rightarrow}A$ pattern, the transition is determined as an offset and an onset; 5) if we detect a S$\rightarrow$T$\rightarrow$S pattern, the transition is directly discarded.

For HCN1 and HCN2, the procedure is a combination of the two strategies above: 1) if there are two onsets having no offset between them, we insert an offset specified to the time when S firstly surpasses A at that interval; 2) similarly, if there are two offsets having no onset between them, the inserted onset is specified to the time when A firstly surpasses S at that interval; 3) any detection violating the rules of 1) and 2) is deleted.

3.5 Note-level transcription

We combine the note segmentation method with a simple pitch estimation process for note-level singing voice transcription. This is implemented by: 1) obtain the onset and

offset times of each note with the note segmentation model, and 2) use the vocal melody extraction method in [27] to obtain the pitch contour of every note, and 3) the final pitch value is simply determined by the median of the pitch contour of that note.

4. EXPERIMENTS

4.1 Data and evaluation metrics

To test the robustness of our model, we set a cross-dataset scenario for the experiments on note segmentation. We use TONAS [10, 19], a dataset of 71 flamenco a cappella sung melody, as our training dataset. In addition, we evaluate our proposed method on the ISMIR2014 sung melody dataset [17]. It contains singing data from 11 female adults, 13 male adults and 14 children.

Section 4.2 first compares the results using different input features. Section 4.3 further compares the results of training with five different objective functions mentioned in Section 3.1. Section 4.4 then compares the ResNet-18 model, the RNN model with attention and the onset detector in the `MADMOM` library [6]. The latter is known as the state of the art for general-purpose onset detection.

For the evaluation metrics, we report the F1-scores of onset detection, offset detection and note transcription and the average overlap ratio (AOR) by using the utilities in the `mir_eval` library with default parameters [23]. To quantify the mismatch between the detected onsets and offsets in note segmentation results, we further compare their *conflict ratio* (CFR), which is defined as the ratio between the number of unpaired detection and the number of all predicted transitions (i.e. onsets plus offsets):

$$\text{CFR} := \frac{\text{\# of unpaired transitions}}{\text{\# of predicted transitions}} \tag{13}$$

The unpaired transition is defined as the onset/offset that cannot be derived from, or that violates the relationship of the states used in the model. For example, in the OON model, if there are two consecutive onsets having no offset in between, the second offset violates the relationship between onset and offset and is accounted as an unpaired detection. On the other hand, the TSN model produces zero unpaired transition and therefore has zero CFR, as discussed in Section 3.4. CFR can be seen as a criterion of systematic consistency for a note segmentation model.

4.2 Comparison of input features

The first five rows of Table 1 lists the results of both onset and offset detection with various inputs: $\mathbf{X}$, $\mathbf{S}^+$, $[\mathbf{S}^+, \mathbf{S}^-]$, $[\mathbf{S}^+, \mathbf{Z}]$, and $[\mathbf{S}^+, \mathbf{S}^-, \mathbf{Z}]$. In comparison to others, using only the spectrogram ($\mathbf{X}$) with less feature engineering gives competitive result, which indicates the power of ResNet in pattern recognition. However, it should be emphasized that using a detailed set of features relevant to onset and offset such as $[\mathbf{S}^+, \mathbf{S}^-, \mathbf{Z}]$ achieves the best note transcription F1-score at 59.5%, which is better than the case using only $\mathbf{X}$ by 3.9%. Such improvement can be seen from other interesting comparisons. For example, adding either $\mathbf{S}^-$ or $\mathbf{Z}$ to $\mathbf{S}^+$ greatly improves the F1-scores of both the onset and offset. Adding $\mathbf{S}^-$ to $\mathbf{S}^+$ also results in 14.5% improvement on onset F1-score, meaning that the backward spectral difference may also be relevant to an onset event. These observations can all be explained by the fact that an onset event can be highly overlapped by an offset event of another notes, and the feature set revealing different aspects of the signal characteristics helps resolve such ambiguity. For simplicity, we adopt $[\mathbf{S}^+, \mathbf{S}^-, \mathbf{Z}]$ in the following experiments.

4.3 Comparison of objective functions

The lower part of Table 1 compares the results of models trained by four baseline objective functions, including OON, M-OON, TSN, and HCN1. Comparing the F1-scores of OON and M-OON, we observe that M-OON slightly degrades onset detection but greatly improves offset detection by 29.2%. This indicates the importance of joint training: incorporating onset information in a shared network can help offset detection.

Although the F1-score of TSN is worse than the one of M-OON, TSN achieves zero CFR as all onsets/offsets can be completely inferred from the rule mentioned in section 3.1 and 3.4. This shows that training on S, A, T and the temporal constraints make highly consistent prediction. However, the poor performance on onset and offset detection implies that using a single T state is not sufficient to describe the behavior of both onset and offset.

HCN1 and HCN2 therefore combine the advantage of both the M-OON model and TSN model. Result shows that the HCN1 model enhances the segmentation quality (reducing CFR to half) compared to the M-OON model and improves the onset and offset detection F1-score compared to the TSN model, then achieves the F1-score of 56.7% on note transcription. In addition, HCN2 model outperforms the HCN1 model in almost all evaluation metrics, where a 2.7% improvement on note transcription F1-score is obtained. Such advancement indicates the importance of regularizing activation/silence detection in note segmentation and transcription tasks.

4.4 Comparison of models

Table 1 also compares two implementations of HCN2 using different modules for the hierarchical classifier: ResNet-18, and the RNN with attention (denoted as RNN-attn) as a sequence classification network for comparison.

Results show that ResNet-18 outperforms RNN-attn in every performance metrics, probably because that an image-based classification network can extract more detailed features considering local information where sequential dependency is not that significant. These findings are partly in line with that in [25], where a CNN outperforms sequence models such as RNN.

4.5 Singing Voice Note Transcription

Table 2 shows the results of singing voice transcription compared with five previous methods: Ryynänen *et al.*

Objective	Classifier	Feature	F1 (onset)	F1 (offset)	CFR	AOR	P (note)	R (note)	F1 (note)
HCN2	ResNet-18	$\mathbf{S}^+$	0.599	0.409	0.078	0.862	0.430	0.394	0.409
		$\mathbf{X}$	0.757	0.740	0.050	0.873	0.576	0.538	0.555
		$[\mathbf{S}^+, \mathbf{S}^-]$	0.744	0.715	0.057	0.870	0.532	0.506	0.517
		$[\mathbf{S}^+, \mathbf{Z}]$	0.745	0.713	0.050	0.870	0.553	0.506	0.527
		$[\mathbf{S}^+, \mathbf{S}^-, \mathbf{Z}]$	0.786	**0.759**	0.043	0.869	**0.625**	**0.569**	**0.594**
	RNN-attn	$[\mathbf{S}^+, \mathbf{S}^-, \mathbf{Z}]$	0.699	0.722	0.050	0.840	0.520	0.502	0.510
HCN1	ResNet-18	$[\mathbf{S}^+, \mathbf{S}^-, \mathbf{Z}]$	0.751	0.739	0.051	0.872	0.608	0.535	0.567
TSN			0.691	0.705	**0.000**	0.864	0.472	0.480	0.474
M-OON			0.778	0.707	0.129	**0.874**	0.574	0.526	0.547
OON			**0.790**	0.415	0.210	0.846	0.313	0.305	0.308

Table 1: Evaluation results for various input features objective functions, and classifier models.

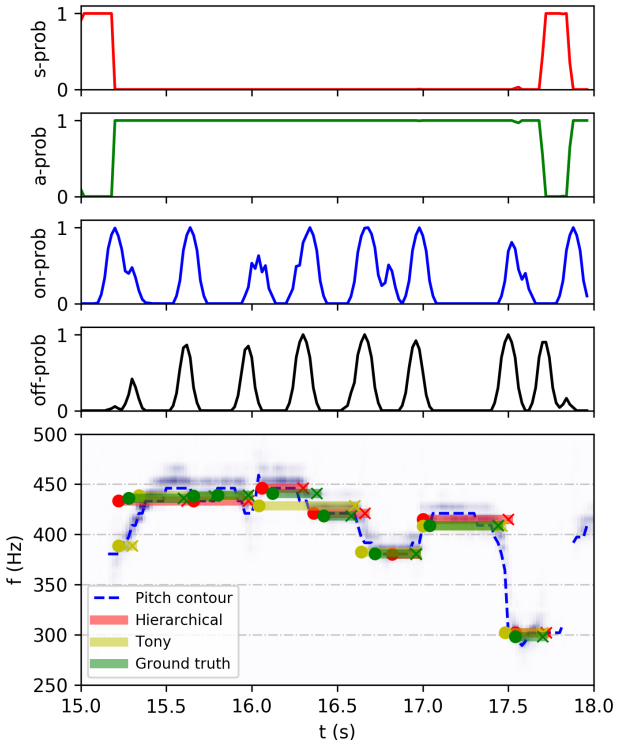


Figure 3: Transcription results from the 15th to the 18th second of 'child10.wav' in the ISMIR 2014 dataset. From top to bottom: predicted likelihood for S, A, O, X, and transcription results. Background of the bottom subfigure: the pitch saliency function $\mathbf{Z}$. Blue dashed lines: estimated pitch contour. Bullet: onset time. X mark: offset time.

Method	Precision	Recall	F
Ryynänen [24]	0.304	0.315	0.308
Gómez & Bonada [10]	0.430	0.373	0.398
SiPTH [18]	0.397	0.440	0.415
Yang [29]	0.409	0.436	0.421
Tony [16]	0.510	0.534	0.520
Proposed	**0.625**	**0.569**	**0.594**

Table 2: Comparison of singing transcription results.

[24] , Gómez & Bonada [10], SiPTH [18], Yang *et al.* [29], and Tony [16]. The results for these five methods are reported in [29]. Our proposed method outperforms all the previous methods by more than 7.4% in terms of the F1-measure. It is important to note that although our model is trained on a dataset with the singing style (flamenco singing) quite different from the testing data, the model still outperforms the Tony software, which performance is actually based on a parametric grid search on the testing dataset [16]. This fact indicates that our method is potentially generalizable over various data modalities. Besides, since we do not directly deal with issues such as vibrato, unstable pitches and tuning shift [29], our model actually benefits more from a stable note segmentation method. This highlights the importance of note segmentation in note transcription.

Fig. 3 illustrates an example of the predicted silence, activation, onset, offset likelihood curves and note transcription results of a clip in the testing dataset. The transcription result from the Tony software is also provided for comparison. It can be shown that Tony tends to miss onsets for consecutive notes, while the proposed model successfully captures almost all the note transitions except the onset at 16.71 sec, which is a challenging case due to the bent pitch contour around the onset event and a relatively short note duration.

5. CONCLUSION

We have presented the effectiveness of the proposed hierarchical classification networks in note segmentation and transcription in singing voice. By unfolding the structure of the state evolution patterns in note sequences and by applying multi-channel data representations to modeling note transitions, the general, robust, and consistent note segmentation procedure plays a vital role in achieving state-of-the-art performance. One important aspect omitted in our discussion is using temporal modeling (e.g., HMM) over the hierarchical state space rather than using post-processing rules to complete the note transcription process. Based on the positive result of this study, this direction is with high potential and will be left as future work.

6. ACKNOWLEDGEMENT

This work is partially supported by the MOST of Taiwan under Grant No. 106-2218-E-001-003-MY3.

7. REFERENCES

[1] E. Anders. *Modeling Music: Studies of Music Transcription, Music Perception and Music Production*. PhD thesis, KTH Royal Institute of Technology, 2018.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.

[3] E. Benetos and S. Dixon. Polyphonic music transcription using note onset and offset detection. In *Proc. IEEE ICASSP*, pages 37–40. IEEE, 2011.

[4] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri. Automatic music transcription: challenges and future directions. *J. Intelligent Information Systems*, 41(3):407–434, 2013.

[5] S. Böck, A. Arzt, F. Krebs, and M. Schedl. Online real-time onset detection with recurrent neural networks. In *Proc. DAFx*, 2012.

[6] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer. Madmom: A new python audio and music signal processing library. In *Proc. ACM MM*, pages 1174–1178, 2016.

[7] P. Brossier, J. P. Bello, and M. D Plumbley. Real-time temporal segmentation of note objects in music signals. In *Proceedings of ICMC 2004, the 30th Annual International Computer Music Conference*, 2004.

[8] S. Chang and K. Lee. A pairwise approach to simultaneous onset/offset detection for singing voice using correntropy. In *Proc. IEEE ICASSP*, pages 629–633, 2014.

[9] F. Eyben, S. Böck, B. Schuller, and A. Graves. Universal onset detection with bidirectional long-short term memory neural networks. In *ISMIR*, pages 589–594, 2010.

[10] E. Gómez and J. Bonada. Towards computer-assisted flamenco transcription: An experimental comparison of automatic transcription algorithms as applied to a cappella singing. *Computer Music Journal*, 37(2):73–90, 2013.

[11] R. Gong and X. Serra. Singing voice phoneme segmentation by hierarchically inferring syllable and phoneme onset positions. In *Interspeech*, pages 716–720, 2018.

[12] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck. Onsets and frames: Dual-objective piano transcription. In *ISMIR*, pages 50–57, 2018.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[14] A. Holzapfel, Y. Stylianou, A. C Gedik, and B. Bozkurt. Three dimensions of pitched instrument onset detection. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1517–1527, 2010.

[15] C.-Y. Liang, L. Su, Y.-H. Yang, and H.-M. Lin. Musical offset detection of pitched instruments: The case of violin. In *ISMIR*, pages 281–287, 2015.

[16] M. Mauch, C. Cannam, R. Bittner, G. Fazekas, J. Salamon, J. Dai, J. Bello, and S. Dixon. Computer-aided melody note transcription using the tony software: Accuracy and efficiency. In *Proc. SMC*, 2015.

[17] E. Molina, A. M. Barbancho-Perez, L. J. Tardón, I. Barbancho-Perez, et al. Evaluation framework for automatic singing transcription. 2014.

[18] E. Molina, L. J. Tardón, A. M. Barbancho, and I. Barbancho. Sipth: Singing transcription based on hysteresis defined on the pitch-time curve. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 23(2):252–263, 2015.

[19] J. Mora, F. Gómez, E. Gómez, F. Escobar-Borrego, and J. M. Díaz-Báñez. Characterization and melodic similarity of a cappella flamenco cantes. In *ISMIR*, pages 351–356, 2010.

[20] R. Nishikimi, E. Nakamura, K. Itoyama, and K. Yoshii. Musical note estimation for F0 trajectories of singing voices based on a bayesian semi-beat-synchronous hmm. In *ISMIR*, pages 461–467, 2016.

[21] G. Peeters. Music pitch representation by periodicity measures based on combined temporal and spectral representations. In *Proc. IEEE ICASSP*, pages 53–56, 2006.

[22] M. Pesek, A. Leonardis, and M. Marolt. Robust real-time music transcription with a compositional hierarchical model. *PloS one*, 12(1), 2017.

[23] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel. mir_eval: A transparent implementation of common mir metrics. In *ISMIR*, pages 367–372, 2014.

[24] M. P. Ryynänen and A. P. Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32(3):72–86, 2008.

[25] J. Schlüter and S. Böck. Improved musical onset detection with convolutional neural networks. In *Proc. ICASSP*, pages 6979–6983, 2014.

[26] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[27] L. Su. Vocal melody extraction using patch-based cnn. In *Proc. IEEE ICASSP*, pages 371–375, 2018.

[28] L. Su and Y.-H. Yang. Combining spectral and temporal representations for multipitch estimation of polyphonic music. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 23(10):1600–1612, 2015.

[29] L. Yang, A. Maezawa, J. B. Smith, and E. Chew. Probabilistic transcription of sung melody using a pitch dynamic model. In *Proc. IEEE ICASSP*, pages 301–305, 2017.

VirtuosoNet: A HIERARCHICAL RNN-BASED SYSTEM FOR MODELING EXPRESSIVE PIANO PERFORMANCE

Dasaem Jeong[1] **Taegyun Kwon**[1] **Yoojin Kim**[1]
Kyogu Lee[2] **Juhan Nam**[1]
[1] Graduate School of Culture Technology, KAIST , Korea
[2] Graduate School of Convergence Science and Technology, Seoul National University, Korea
{jdasam, ilcobo2, luciaicul}@kaist.ac.kr, kglee@snu.ac.kr, juhannam@kaist.ac.kr

ABSTRACT

In this paper, we present our application of deep neural network to modeling piano performance, which imitates the expressive control of tempo, dynamics, articulations and pedaling from pianists. Our model consists of recurrent neural networks with hierarchical attention and conditional variational autoencoder. The model takes a sequence of note-level score features extracted from MusicXML as input and predicts piano performance features of the corresponding notes. To render musical expressions consistently over long-term sections, we first predict tempo and dynamics in measure-level and, based on the result, refine them in note-level. The evaluation through listening test shows that our model achieves a more human-like expressiveness compared to previous models. We also share the dataset we used for the experiment.

1. INTRODUCTION

Music performance is one of the most essential activities in music. Good performance requires not only translating notes in the score into physical actions with precise timing and right pitch on an instrument but also delivering emotions and messages through subtle controls of tempo, dynamics, articulations and other expressive elements.

There have been research interests in modeling expressive performance using a computational method. A recent review paper comprehensively summarized the history [4]. While some of previous work exploited computational modeling as a tool for understanding how humans perform [3], or listen to music [10], others focused on automatically generating expressive performances. The previous methods include rule-based approaches [2, 8], or probabilistic models [17, 29], and an artificial neural network [5, 11]. The instrument is mainly limited to piano because it is relatively easy to quantify the performances.

Recent approaches have attempted to apply deep learning to modeling expressive piano performance, such as rendering note velocity and deviation of note onset with vanilla recurrent neural network (RNN) [20], or predicting note velocity with a long short-term memory (LSTM) RNN [22]. Others introduced DNN models for generating polyphonic music with expressive timing and dynamics [13, 24]. While these models can generate performance MIDI notes, they are more like music composition models rather than expressive performance models that take music scores as input. Besides piano performance, a recent work presented DNN-based system for modeling expressive drum performance [9].

One of the bottlenecks in the DNN-based approach is the lack of dataset [4]. Since the task is rendering expressive performances from score inputs, the dataset should consist of music scores and their corresponding performances by human musicians. Furthermore, the pair of score and performance should be aligned in note-level to effectively train the model. Also, ideally, the list of music score and performance should cover various composers and performance styles.

In this paper, we present a hierarchical RNN-based model for expressive piano performance along with a dataset that we organized. The model takes MusicXML as input and generates performance MIDI with expressive tempo, dynamics, articulation and pedaling. The model consists of RNN with hierarchical attention network and conditional variational autoencoder (CVAE). In particular, the model predicts the performance features using a multi-scale approach; it first predicts tempo and dynamics in measure-level and, based on the result, fine-tunes them in note-level. A listening test with professional pianists shows that our model achieves a more human-like expressiveness compared to previous models.

2. DATASET

2.1 Performance and Score Data

As aforementioned, we need a dataset of human performances with their corresponding music scores to train a neural network model. A list of expressive performance datasets are summarized in [4]. Among others, Yamaha

Signature MIDI collection [1], which are recorded during Yamaha e-Competitions with computer-controlled pianos, is the largest public dataset that provides a substantial amount of expressive performance MIDI of professional pianists. Some of the pianists performed the same piece more than once in different rounds of the competition in different years. The Yamaha collection has been employed in automatic performance generation [24] and automatic music transcription and audio synthesis [12] as well.

While the Yamaha collection provides high-quality piano performance data in MIDI, it does not contain the corresponding music scores of the pieces. Thus, we collected the score files from another source. Specifically, we downloaded them from *MuseScore*, a community-based web platform of music score [2]. The scores were transcribed voluntarily by the community users and can be exported in MusicXML format. We also included our own transcriptions of scores to the dataset. While MIDI is suitable for representing performance, MusicXML aims to represent the Western music notation in its entirety. Therefore, MusicXML can contain various types of musical symbols such as rest, slur, beam, barline, key and time signature, articulation, ornament markings and so on, which are excluded in MIDI format.

2.2 Data Matching and Refinement

Since we collected the performance and score data from different sources, we had to match and refine them. In particular, transcription styles in the crowdsourced MusicXML files are not consistent. For example, some of the transcribers add extra expressions such as dynamics markings or tempo change to make the score sounds more expressive. They usually set them to "invisible objects" to make the transcribed score appear as the reference score. We deleted such extra markings added by transcribers. Also, we manually checked whether the performances followed the repetitions in the scores. If a performance skipped the repetition, we omitted the repetition from the score so that the performance and the score can be aligned.

To train a model with note-level score features and performance features, each note in the score should be matched to that in the performance. We employed a score-to-performance alignment algorithm proposed by Nakamura et al. [23]. The algorithm automatically handles asynchronously performed notes as well as missing and extra notes in the performance, and returns a list of note-to-note matches. Although the algorithm showed high accuracy in our test, a small amount of alignment errors can be critical in extracting performance features such as tempo or onset deviation. Since the dataset is too large to make manual corrections, we filtered out some erroneous matches based on simple rules and excluded them in training the performance model. For example, if a matched performance note is too close or even earlier than the previous note in the score, we regarded it as an alignment error. Also, if multiple notes have the same onset time in the score (e.g., chord notes) but one is too far from other notes in performance, we regarded it as an alignment error as well.

We found that this additional refinement made severe improvement on the training result, especially on onset deviation, or micro-timing, of individual notes. The standard deviation of onset deviation decreases from 7.369 to 0.053 after the refinement, where the unit is quarter-notes. Without the refinement, the prediction of onset deviation became too noisy that one could not perceive correct rhythm.

As a result, we collected music scores of 226 pieces by 16 composers in MusicXML and 1,052 piano performances in MIDI. After the matching and refinement, the score and performance data contain a total of 666,918 notes and 3,547,683 notes, respectively. Among the performance notes, 131,095 notes were failed to be aligned with score notes, and additional 114,914 notes were excluded by our refinement algorithm. The number of valid performance notes is ten times larger than the Magaloff corpus [7], which is the largest existing dataset for classical piano music [4].

3. SYSTEM ARCHITECTURE

3.1 Background

3.1.1 Input and Output Features

Designing input and output features is an important issue in performance modeling because it defines the characteristics of the computational task [4]. We followed the scheme we previously proposed in [15], which covers a wide range of score and performance features. The score features include pitch, duration, articulation marking, slur and beam status, tempo marking, dynamic markings, and so on. The performance features include absolute tempo, velocity, onset deviation, articulation and pedal usages. All the features are encoded in the note-level so that each note had the same dimension of score features and performance features.

3.1.2 Hierarchical Attention Network

Recent research has shown that a hierarchical approach can improve the performance of RNN model in modeling sequential data [6, 30]. It was also demonstrated that the hierarchical approach has advantages in generating symbolic music data [25]. In this paper, we employ a hierarchical attention network (HAN) to predict a sequence of performance features from a sequence of score features.

The HAN composes higher-level representations by summarizing lower-level representations in pre-defined hierarchical boundaries using a weighted sum. In our case, we set beat and measure as the hierarchical boundaries so that beat-level attention and measure-level attention summarize note-level and beat-level representations, respectively. Instead of directly implementing the HAN in [30], we combined it with the idea of multi-head attention [28] which splits the dimension into several heads and applies different weights of attention for each split.

Composing nodes through the attention layers can be described as follows. For each hierarchical boundary,

[1] `http://www.yamahaden.com/midi-files`
[2] `https://musescore.com`

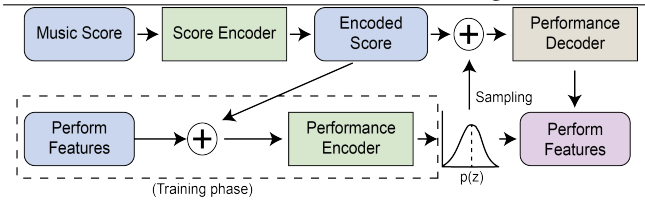


Figure 1. The overview of the proposed system.

which can be a beat or a measure in music score, the notes in the boundary can be indexed with $t \in [B_f, B_l]$, where B_f and B_l represent the index of the first and last notes in the selected boundary B. The lower-level hidden states $\mathbf{h}_t$ for t in the boundary B are summarized by context attention to compose a higher-level node $\mathbf{m}$. There are a total I number of attention heads indexed with i.

$$
\begin{aligned}
\mathbf{u}_t &= \tanh(\mathbf{W}_a \mathbf{h}_t + \mathbf{b}_a) \\
\mathbf{u}_t^i &= \mathbf{u}_{t,i:(i+1)d} \\
\mathbf{h}_t^i &= \mathbf{h}_{t,i:(i+1)d} \\
\boldsymbol{\alpha}_v^i &= \frac{\exp(\mathbf{u}_t^{i\intercal} \mathbf{u}_c^i)}{\sum_t \exp(\mathbf{u}_t^{i\intercal} \mathbf{u}_c^i)} \\
\mathbf{m}^i &= \sum_t \boldsymbol{\alpha}_t^i * \mathbf{h}_t^i \\
\mathbf{m} &= \text{Concat}(\mathbf{m}^0, ..., \mathbf{m}^I)
\end{aligned}
\tag{1}
$$

where $\mathbf{W}_a$ and $\mathbf{b}_a$ denote weight and bias parameters of attention, and $\mathbf{u}_c$ denotes a context vector representing query for importance, which are trainable parameters. The sequence of summarized nodes are fed into a new layer of LSTM.

3.1.3 Conditional VAE

A music score can be interpreted and performed in various styles, i.e. with a different tempo or phrasing. Therefore it is important to enable the performance modeling system to generate different types of performance. On the other hand, the variation of performance can be an obstacle for training the model, because it has to generate different outputs from the same input. To solve this problem we employed a conditional variational autoencoder (CVAE), which we proposed in our previous work [14].

VAE is a widely used generative models based on deep neural networks [19]. It is a type of autoencoder, which compresses input information into a lower dimensional latent vector and decodes the original information from the compressed latent vector. The main difference is that VAE constrains its latent vector to be sampled from a probability distribution. VAE consists of an encoder that models $q(z|x)$ and decoder to model $p(x|z)$. VAE also models the probability of latent vector $p(z)$, which usually has a normal distribution. The training loss of VAE can be define as follows:

$$
\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{rec}} + \beta D_{KL}[(q(z|x)||p(z)] \tag{2}
$$

where $\mathcal{L}_{\text{rec}}$ is the reconstruction error from AE, D_{KL} is Kullback-Leibler divergence (KLD), and β is a weight for the KLD.

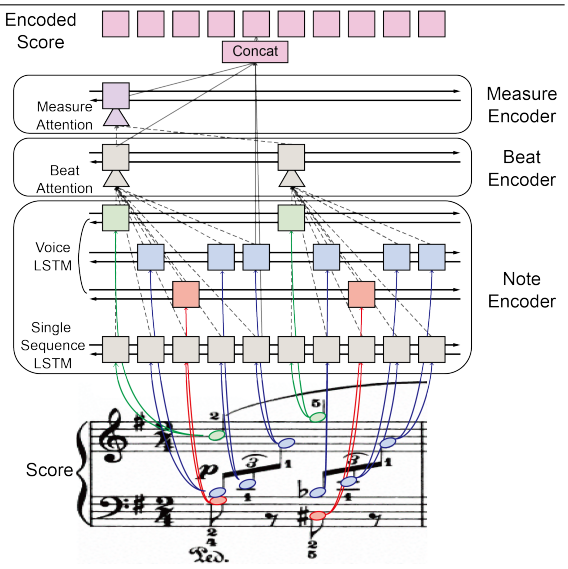


Figure 2. Diagram for Score Encoder with HAN and RNN

A conditional VAE (CVAE) provides an additional condition so that the output satisfies the given condition [27]. In our system, the condition is the learned score representation, and the target output are the performance features. The idea of employing CVAE for expressive performance modeling was first proposed in [21]. While the previous work encoded the latent vector in note-level, our idea is to encode the performance style in a longer-level, such as an entire piece.

3.2 Proposed System

Our proposed system consists of three parts: score encoder, performance encoder, and performance decoder as depicted in Figure 1.

The role of score encoder is to learn score representations C from an input sequence of notes. It consists of three hierarchical-levels: note, beat, and measure. Each level has a corresponding bidirectional LSTM unit with a different hidden size and number of layers. The note-level layer consists of two different LSTM units, one taking the input as a single sequence, and the other taking the input as voice-separated sequences. The "voice" means the voice index in MusicXML that represents an independent stream of music as depicted with different colors of notes in Figure 2. The hidden representations of the lower-level are summarized through the HAN to compose higher-level nodes. The output of the note-level LSTM is summarized to beat-level nodes and then they are fed into the beat-level LSTM. Similarly, we compose the measure-level LSTM. We concatenate the outputs of all the three layers in a note-level as depicted in Figure 2. The output of score encoder is a sequence with the same length as the input. Since we use multi-head attention instead of single-head attention, each attention head focuses on the different type of notes as illustrated in Figure 3.

We implemented the performance encoder using CVAE that models $q(z|C, y)$ to summarize the given performance y in score condition C to a probability distribution of the

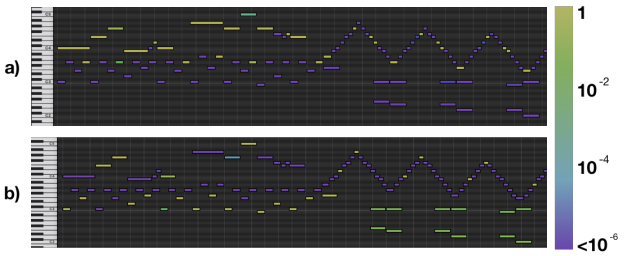


Figure 3. Visualization of attention weights from different attention heads. a) focuses more on the melody notes while b) focuses more on the bass or harmonic notes.

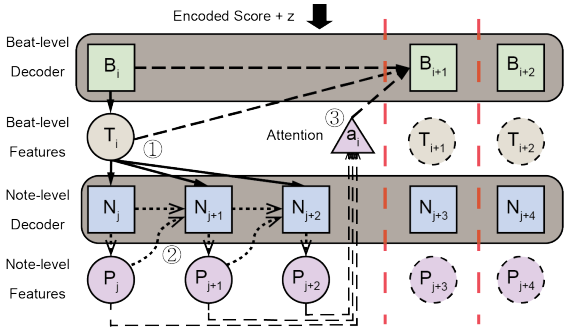


Figure 4. The figure shows how the beat-level decoder and the note-level decoder feed its results to the other. The dashed lines in red indicate the edge of beats.

latent vector z, which can be regarded as a performance style vector. C and y are concatenated and fed into a single dense layer that contracts the feature dimension. We use uni-directional note-level LSTM and measure-level HAN-LSTM to process the contracted input. The last output of the sequence from the measure-level LSTM is used to infer μ and σ of $q(z|C, y)$ by a dense layer.

During the actual performance generation from a given score, the performance encoding is bypassed, and the system randomly samples the style vector z from a normal distribution or exploits a pre-encoded z from other performances.

The performance decoder uses LSTMs to generate a sequence of performance features $\hat{y}$ for the given condition C and the style vector z. Since the tempo is always estimated in beat level, we have two different LSTM units, one in the beat-level and the other in the note-level. Both LSTMs are in auto-regressive, i.e., take their own output from the previous step as an input, and the outputs of the note-level decoder is fed into the beat-level decoder, and vice versa, as presented in Figure 4.

3.3 Measure-level Module

One of the main difficulties in expressive performance modeling is achieving long-term expression such as gradual change of tempo or contrast between loud and quiet sections. To solve this problem, we propose an optional measure-level module that predicts measure-level tempo and dynamics as presented in Figure 5. The main idea is to make our system predict overall progress of the performance in measure-level and then refine it in note-level. A similar idea achieved a successive result in image generation using GAN, which started training in a low resolution and progressively in higher resolutions [16].

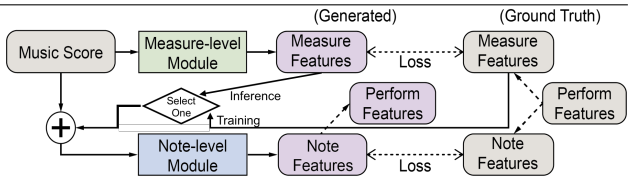


Figure 5. Diagram for Measure-level modules

To train the measure-level module, we have to define measure-level performance features. The measure-level tempo is defined by elapsed time to play the measure divided by the length of the measure in quarter-notes. We used average velocities of notes in the measure for a measure-level dynamics. The measure-level module has almost the same architecture with the note-level modules except that the output of the score encoder is the measure-level states instead of concatenated result of note, beat and measure hidden states. The performance encoder and decoder are also in measure level.

In this hierarchical approach, the note-level module takes not only the score data but also the output of the measure-level module as a concatenated input. It is possible to combine two modules as a single model or in a single training process, but we made two modules independent and trained them separately. Therefore, the note-level module is trained with ground-truth measure-level outputs.

4. EXPERIMENTS

4.1 Training

We split the dataset into training, validation, and test sets so that each set has a size of approximately 8:1:1 in the number of piece, performance, and notes, while considering the distribution of composers in each set. A single piece was included only in either of one of the splits. For the training set, we sliced the input sequences at the measure boundaries with the least size of 500 notes. When training the measure-level module, the sequence has at least 2000 notes or entire notes if the piece is short. The note is ordered by its appearance order and pitch. The features with continuous value was normalized to have zero mean and unit standard deviation.

We calculated the loss in mean square error (MSE) between each feature. The loss was calculated for each note and each output features, except the tempo, whose loss was calculated in beat level. During the training, the input sequences included all the notes that have non-matching performance notes, because missing notes in the input data can change the context of the other notes in the score. However, these notes were excluded in the loss calculation because we could not extract performance features for the notes. Since the articulation is largely affected by the sustain pedal, we reduced the weight for the articulation loss to 0.1 for notes with the sustain pedal pressed at the offset.

Model	Tempo	Vel	Dev	Artc	Pedal
Baseline	0.400	0.673	0.773	0.721	0.843
HAN-S	0.269	0.607	0.753	**0.688**	0.820
HAN-M	**0.220**	**0.532**	**0.747**	0.754	**0.810**

Table 1. Reconstruction loss of each model on the test set in MSE. Vel, Dev and Artc denote velocity, onset deviation and articulation, respectively.

We used the ADAM optimizer [18] with an initial learning rate of 0.0003 and dropout ratio of 0.1. To avoid that the system bypasses z during the decoding, we use the KLD weight annealing as proposed in [1], so that the KLD weight started from zero at the beginning of training, and increased to 0.02 gradually.

4.2 Model Configuration

The score encoder of our proposed system has three-layer dense network of size 128 with ReLU activation as an embedding layer, two-layer bidirectional(Bi)-LSTMs of size 128 for note-level and voice-level, two-layer Bi-LSTM of size 64 for beat-level, and one-layer Bi-LSTM of size 64 for measure-level. The performance encoder has two-layer unidirectional(Uni)-LSTM of size 16 for note-level and one-layer Uni-LSTM of size 16 for measure-level. The size of latent vector z in CVAE is 16. The performance decoder consists of one-layer Uni-LSTMs for beat-level and note-level both of size 64. The measure-level module has almost the same setting except that every hidden size of the network in the performance encoder is 8 including the latent vector z.

To compare our approach with HAN architecture and measure-level modules (HAN-M), we also trained two other models. The first model is a baseline model that uses only three-layers LSTM in note-level with hidden size of 256. The other model, which will be denoted as HAN-S, is a model that excludes the measure-level module. In HAN-S, the hidden size of beat-level layer in the score encoder and performance decoder was 128.

5. RESULTS

5.1 Reconstruction Error

Quantitative evaluation of modeling expressive performance is a not trivial issue. One of the frequently used quantitative evaluation method is calculating MSE of output features [4]. Comparing the predicted outputs with "target" performance can be arbitrary, because there can be various ways to perform the score. In our system, however, there is a performance encoder and a latent style vector z that, ideally, makes the output in a style of the target performance. Therefore, comparing output features with the target performances is more reasonable. Also, as a learning model, it is fair to check the test loss with the same criteria used for training.

Table 1 shows the reconstruction loss of each model on the test set which includes 21 pieces and 109 performances. The two HAN models achieved much less reconstruction error than the baseline model. This indicates that the hierarchical approach helps the model to generalize to unseen data. Between the two HAN models, HAN-M is slightly better than HAN-S. We have tested different parameter sizes for HAN-S so that HAN-S has a similar number of parameters with the sum of two modules in HAN-M, but the result was not much different.

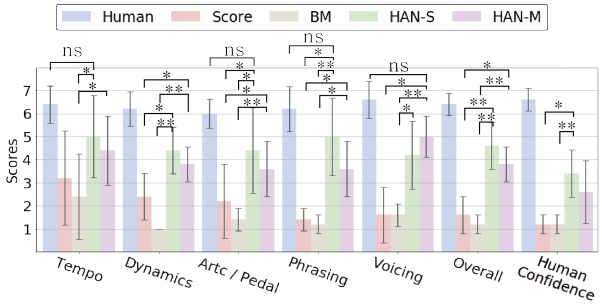


Figure 6. Average score of the listening test for Schubert Sonata in 7-point Likert scale. The t-test results between our models (HAN-S,M) and Human are marked with `ns` only if it is not significant. The results between our models and the others models are marked only if it is significant. "*" and "**" denote "p≤0.05" and "p≤0.01", respectively.

5.2 Listening Test

We also conducted a listening test to evaluate our model qualitatively. We asked five students, who are majoring piano at a college of music, to listen to the rendered performances and evaluate them with criteria presented in Figure 6 in 7-point Likert scale (1 - very bad, 7 - very good) with additional comments on the performance. We chose three pieces of different styles from our test set: the first movement from Beethoven's Piano Sonata No. 5 (cut before recapitulation), Chopin's Etude op. 10 No. 2 (entire piece), and the first movement from Schubert Piano Sonata D.664 (cut before development).

We prepared five different performances MIDI per piece: a human performance from Yamaha e-competition, a direct export from MusicXML score to MIDI by a notation program (MuseScore), each of rendered performances from HAN-S and HAN-M, and Basis Mixer (BM). The result exported from MuseScore had no tempo change but the velocities of notes were changed by a simple rule-based conversion of dynamic markings in the score. BM is the only publicly available model that does not require additional notation among previous expressive performance models [5]. It also achieved a highest score among other computational methods in previous research [26]. We included the BM model in the listening test and generated performances with the same MusicXML file we used for our model [3]. Since the recording and playback in audio systems can affect the quality of performance, we invited the participants to our studio and played the prepared MIDI files with a Yamaha Disklavier piano. Each performance was played once in a random order.

The result of evaluation on Schubert is presented in Figure 6 As expected, all participants gave highest scores in

[3] `https://basismixer.cp.jku.at/static/app.html`

every criteria for the human performance. Our proposed model, HAN-S and HAN-M, achieved higher scores in all seven criteria compared to other models. Two among five participants gave more than five out of seven points as the human confidence for the performance by HAN-S, and one gave five points to HAN-M. The t-test result showed that HAN-S and HAN-M showed statistically significant differences ($p \leq 0.05$) compared to the score and the BM model in overall ratings of the performance of Schubert.

The positive comments on our models were: *"the interpretation was interesting"* (Beethoven, HAN-M), *"felt that the flow of performance was humane"* (Beethoven, HAN-S), *"voicing was too good so that it felt like performed by multiple performers"* (Chopin, HAN-S), *"sounded like a performance by human with strong characteristics"* (Chopin, HAN-S), *"voicing was fine except some faults"* (Schubert, HAN-M), and *"sounded like machine-generated performance with fine pedaling"* (Schubert, HAN-S) .

There were also negative comments criticizing our models, such as *"used too much pedal"*, *"pedal points were unnatural"*, *"lack of color"*, *"too short articulation"*, *"some tempo or dynamic changes were unnatural"*, *"touch was too light"*, and *"it did not seem that the performer was listening to the performance"*. Although our models had predicted the pedal usage, the pedaling was often too deep and "dirty" or too shallow. The result showed that the note-level pedal embedding needs improvement.

The responses of our participants for performances by BM, which is a data-driven model based on RNN, were negative regardless of the piece. The comments from the participants said that *"although there was a clear intention to express phrasing, it was unnatural and sounded like a mechanical interpretation"* (Schubert), *"inaccurate and limping rhythm"* (Beethoven and Schubert),and *"the temporal gaps at measure boundary were unnatural"* (Chopin and Schubert). Unlike the Score MIDI, the performance by BM included clear change in tempo for phrasing. However, most of the participants gave almost the same level of negative response to its phrasing quality compared to the Score MIDI. This shows how difficult it is to model phrasing of the music.

The results were also largely differed by the characteristics of the piece. For example, Schubert's Sonata has a song-like melody with arpeggio accompaniments. Hence, it was important to model the natural phrasing, e.g., subtle change of tempo and velocity according to the melody. On the other hand, the fast chromatic scale in Chopin's Etude demands a stable tempo. Therefore, Score MIDI received six out of seven points for overall quality from three participants because the performance was in perfectly constant tempo with strict following of dynamic markings. The flexibility of tempo generated by our model was not favored by the participants in case of Chopin's Etude.

In summary, the results of listening test shows that our models have achieved more natural expressions compared to the other models, especially in a piece with song-like melodies. Modeling the pedal usage and a human-like stable tempo are issues to further investigate.

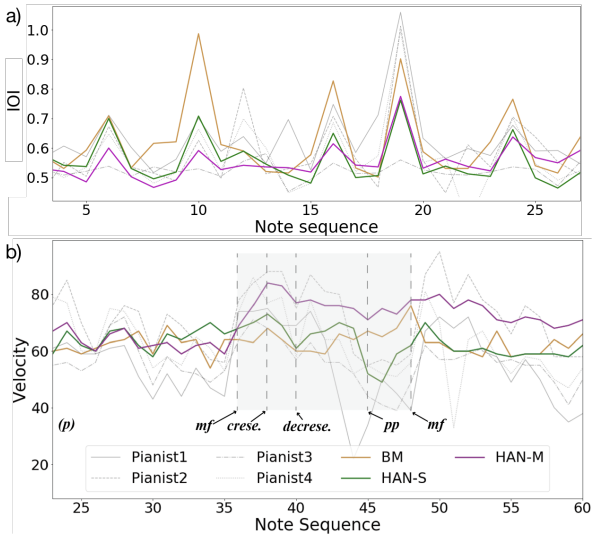


Figure 7. a) Local tempo changes and b) Dynamics change in different performances of Schubert's Piano Sonata

5.3 Case Study: Comparison in Tempo and Dynamics

The quality of phrasing can be also observed from examples. Figure 7-a) compares local tempo changes in difference performances of Schubert's Sonata. The local tempo is represented with inter-onset-interval (IOI) which is computed by dividing seconds into quarter-note. BM has an evident peak at around the 10th note, which was exaggerated than any other human pianists. In terms of Pearson correlation, HAN-S and HAN-M have a strong positive correlation with the pianists ($0.7<r<1.0$) while the BM model has a less positive correlation ($0.3<r<0.5$).

Figure 7-b) compares dynamics changes of melody notes in different performances of the same piece. The dynamics is represented with MIDI note velocity. Increasing and decreasing timings of HAN-M and HAN-S are generally similar to pianists. For example, *decrescendo* starts at note sequence 40 which follows *crescendo*, then *pp* starts at 45 and comes back to *mf* at 48. Both HAN-M and HAN-S show similar downward and upward curves with pianists while the BM model shows just slight upward curve. These trend can be proved by correlation coefficients among pianists and generated models. HAN-S and HAN-M have significant positive correlation with pianists ($0.3<r<0.7$) while BM has almost no correlation ($r<0.1$).

6. CONCLUSIONS

We introduced a hierarchical RNN-based system for modeling expressive piano performance and a dataset for training the model. Our listening test demonstrated that our model achieved more human-like musical expression compared to the previous model [5]. The source code and dataset are available in `https://github.com/jdasam/virtuosoNet`.

7. ACKNOWLEDGEMENT

This research was funded by Samsung Research Funding Center under the project number SRFC-IT1702-12.

8. REFERENCES

[1] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proc. of the 20th Conf. on Computational Natural Language Learning*, 2016.

[2] Sergio Canazza, Giovanni De Poli, and Antonio Rodà. Caro 2.0: an interactive system for expressive music rendering. *Advances in Human-Computer Interaction*, 2015:2, 2015.

[3] Carlos Eduardo Cancino-Chacón, Thassilo Gadermaier, Gerhard Widmer, and Maarten Grachten. An evaluation of linear and non-linear models of expressive dynamics in classical piano and symphonic music. *Machine Learning*, 106(6):887–909, Jun 2017.

[4] Carlos Eduardo Cancino-Chacón, Maarten Grachten, Werner Goebl, and Gerhard Widmer. Computational models of expressive music performance: A comprehensive and critical review. *Frontiers in Digital Humanities*, 5:25, 2018.

[5] Carlos Eduardo Cancino Chacón and Maarten Grachten. The basis mixer: a computational romantic pianist. In *Late-Breaking Demos of the 17th International Society for Music Information Retrieval Conf. (ISMIR)*, 2016.

[6] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2017.

[7] Sebastian Flossmann, Werner Goebl, Maarten Grachten, Bernhard Niedermayer, and Gerhard Widmer. The magaloff project: An interim report. *Journal of New Music Research*, 39(4):363–377, 2010.

[8] Anders Friberg, Roberto Bresin, and Johan Sundberg. Overview of the kth rule system for musical performance. *Advances in Cognitive Psychology*, 2(2-3):145–161, 2006.

[9] Jon Gillick, Adam Roberts, Jesse Engel, Douglas Eck, and David Bamman. Learning to groove with inverse sequence transformations. In *Proc. the 36th International Conf. on Machine Learning (ICML)*, volume 97, pages 2269–2279, 2019.

[10] Bruno Gingras, Marcus T Pearce, Meghan Goodchild, Roger T Dean, Geraint Wiggins, and Stephen McAdams. Linking melodic expectation to expressive performance timing and perceived musical tension. *Journal of Experimental Psychology: Human Perception and Performance*, 42(4):594, 2016.

[11] Sergio Giraldo and Rafael Ramirez. A machine learning approach to ornamentation modeling and synthesis in jazz guitar. *Journal of Mathematics and Music*, 10(2):107–126, 2016.

[12] Curtis Hawthorne, Andrew Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019.

[13] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019.

[14] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. Graph neural network for music score data and modeling expressive piano performance. In *Proc. the 36th International Conf. on Machine Learning (ICML)*, volume 97, pages 3060–3070, 2019.

[15] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. Score and performance features for rendering expressive music performances. In *Proc. of Music Encoding Conf.*, 2019.

[16] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conf. on Learning Representations (ICLR)*, 2018.

[17] Tae Hun Kim, Satoru Fukayama, Takuya Nishimoto, and Shigeki Sagayama. Statistical approach to automatic expressive rendition of polyphonic piano music. In Alexis Kirke and Eduardo R. Miranda, editors, *Guide to Computing for Expressive Music Performance*, pages 145–179. London, 2013.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[20] Stanislas Lauly. Modélisation de l'interprétation des pianistes & applications d'auto-encodeurs sur des modèles temporels. Master's thesis, University of Montréal, 2010.

[21] Akira Maezawa. Deep piano performance rendering with conditional VAE. In *Late-Breaking Demos of the 19th International Society for Music Information Retrieval Conf. (ISMIR)*, 2018.

[22] Iman Malik and Carl Henrik Ek. Neural translation of musical style. *CoRR*, abs/1708.03535, 2017.

[23] Eita Nakamura, Kazuyoshi Yoshii, and Haruhiro Katayose. Performance error detection and post-processing for fast and accurate symbolic music alignment. In *Proc. of 18th International Society for Music Information Retrieval Conf. (ISMIR)*, 2017.

[24] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. This time with feeling: learning expressive musical performance. *Neural Computing and Applications*, Nov 2018.

[25] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. In *Proc. of the 35th International Conf. on Machine Learning (ICML)*, pages 4364–4373, 2018.

[26] Emery Schubert, Sergio Canazza, Giovanni De Poli, and Antonio Rodà. Algorithms can mimic human piano performance: the deep blues of music. *Journal of New Music Research*, 46(2):175–186, 2017.

[27] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3483–3491, 2015.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017.

[29] Gerhard Widmer, Sebastian Flossmann, and Maarten Grachten. YQX plays chopin. *AI magazine*, 30(3):35, 2009.

[30] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proc. of the 2016 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.

MIDI PASSAGE RETRIEVAL USING CELL PHONE PICTURES OF SHEET MUSIC

Daniel Yang* **Thitaree Tanprasert*** **Teerapat Jenrungrot** **Mengyi Shan** **TJ Tsai**
Harvey Mudd College, Claremont, CA USA
{dhyang, ttanprasert, mjenrungrot, mshan, ttsai}@hmc.edu

ABSTRACT

This paper investigates a cross-modal retrieval problem in which a user would like to retrieve a passage of music from a MIDI file by taking a cell phone picture of a physical page of sheet music. While audio–sheet music retrieval has been explored by a number of works, this scenario is novel in that the query is a cell phone picture rather than a digital scan. To solve this problem, we introduce a mid-level feature representation called a bootleg score which explicitly encodes the rules of Western musical notation. We convert both the MIDI and the sheet music into bootleg scores using deterministic rules of music and classical computer vision techniques for detecting simple geometric shapes. Once the MIDI and cell phone image have been converted into bootleg scores, we estimate the alignment using dynamic programming. The most notable characteristic of our system is that it does test-time adaptation and has no trainable weights at all—only a set of about 30 hyperparameters. On a dataset containing 1000 cell phone pictures taken of 100 scores of classical piano music, our system achieves an F measure score of .869 and outperforms baseline systems based on commercial optical music recognition software.

1. INTRODUCTION

Consider this scenario: A person is practicing at the piano, and would like to know what a particular passage of music sounds like. She takes a cell phone picture of a portion of the physical sheet music in front of her, and is immediately able to hear what those lines of music sound like.

In this paper, we explore the feasibility of such an application where we assume that the piece is known and a MIDI file of the piece is available. Our goal is to retrieve a passage of music from a MIDI file using a cell phone image as a query. This is a cross-modal retrieval scenario.

Several works have investigated the correspondence between audio and sheet music images. There are two general approaches to the problem. The first approach is to use an existing optical music recognition (OMR) system to convert the sheet music into a symbolic (MIDI-like) representation, to compute chroma-like features, and then to compare the resulting sequences to chroma features extracted from the audio. This approach has been applied to synchronizing audio and sheet music [4,5,15,16,21], identifying audio recordings that correspond to a given sheet music representation [14], and finding the audio segment corresponding to a fragment of sheet music [13]. A different approach has been explored in recent years: convolutional neural networks (CNNs). This approach attempts to learn a multimodal CNN that can embed a short segment of sheet music and a short segment of audio into the same feature space, where similarity can be computed directly. This approach has been explored in the context of online sheet music score following [7], sheet music retrieval given an audio query [8,9,11,12], and offline alignment of sheet music and audio [9]. Dorfer et al. [10] have also recently shown promising results formulating the score following problem as a reinforcement learning game. See [18] for a recent overview of work in this area.

The key novelty in our scenario is the fact that the queries are cell phone images. All of the works described above assume that the sheet music is either a synthetically rendered image or a digital scan of printed sheet music. In recent years, a few works have begun to explore optical music recognition (OMR) on camera-based musical scores [1–3,22,23]. Here, we explore the use of cell phone images of sheet music for *retrieval*. Cell phone images provide a natural and convenient way to retrieve music-related information, and this motivates our current study.

The main conceptual contribution of this paper is to introduce a mid-level feature representation called a bootleg score which explicitly encodes the conventions of Western musical notation. As we will show, it is possible to convert MIDI into a bootleg score using the rules of musical notation, and to convert the cell phone image into a bootleg score using classical computer vision techniques for detecting simple geometrical shapes. Once we have converted the MIDI and cell phone image into bootleg feature space, we can estimate the alignment using subsequence DTW. The most notable characteristic of our system is that it does test-time adaptation and *contains no trainable weights at all*—only a set of approximately 30 hyperparameters. In the remainder of this paper, we will describe the system and present our experimental results.

*The first two authors had equal contribution.

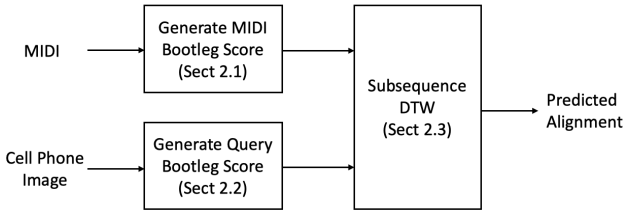


Figure 1. Block diagram of the proposed system.

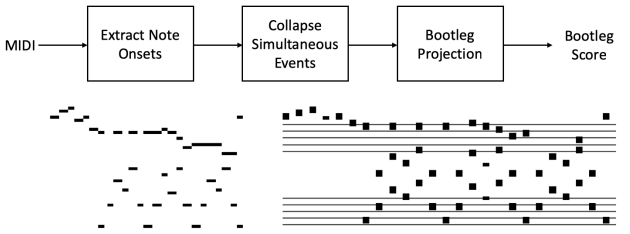


Figure 2. Overview of generating the MIDI bootleg score (Section 2.1). Below the block diagram, a short MIDI passage (left) and its corresponding bootleg score are shown.

2. SYSTEM DESCRIPTION

Our system takes two inputs: a cell phone picture of a page of sheet music and a MIDI file of the corresponding piece. The output of the system is a prediction of the time segment in the MIDI file that matches the lines of sheet music shown in the cell phone picture. Note that in this problem formulation, we assume that the piece is known, and that we are trying to identify the matching passage of music in the piece. In our study, we focus on piano music.

Our approach has three main components, which are shown in Figure 1. The first two components convert the MIDI and cell phone image into a representation which we call a bootleg score. A bootleg score is a very low-dimensional representation of music which is a hybrid between sheet music and MIDI. It is a manually designed feature space that explicitly encodes the rules of Western musical notation. The third component is to temporally align the two bootleg scores using subsequence DTW. These three components will be discussed in the next three subsections. [1]

2.1 Generating MIDI Bootleg Score

Generating the MIDI bootleg score consists of the three steps shown in Figure 2. The first step is to extract a list of all individual note onsets. The second step is to cluster the note onsets into groups of simultaneous note events. After this second step, we have a list of note events, where each note event consists of one or more simultaneous note onsets. The third step is to project this list of note events into the bootleg feature space.

The bootleg feature representation can be thought of as a very crude version of sheet music (thus the name "bootleg score"). It asks the question, "If I were to look at the sheet music corresponding to this MIDI file, where would the notehead for each note onset appear among the staff lines?" Note that there is ambiguity when mapping from a MIDI note value to a position in a staff line system. For example, a note onset with note value 60 (C4) could appear in the sheet music as a C natural or a B sharp, [2] and it could also appear in the right hand staff (i.e. one ledger line below a staff with treble clef) or the left hand staff (i.e. one ledger line above a staff with bass clef). The bootleg feature representation handles ambiguity by simply placing a notehead at all possible locations. The bootleg score is a binary image containing only these floating noteheads.

The bootleg score is a very low dimensional representation. Along the vertical dimension, it represents each staff line location as a single bootleg pixel (which we will refer to as a "bixel" to differentiate between high-dimensional raw image pixels and low-dimensional bootleg score pixels). For example, two adjacent staff lines would span three bixels: two bixels for the staff lines and one bixel for the position in between. The bootleg score contains both right hand and left hand staves, similar to printed piano sheet music. In total, the bootleg score is 62 bixels tall. Along the horizontal dimension, we represent each simultaneous note event as a single bixel column. We found through experimentation that a simple modification improves performance in the alignment stage (Section 2.3): we simply repeat each bixel column twice and insert an empty bixel column between each simultaneous note event. This gives the system more flexibility to deal with noisy bixel columns in the alignment stage.

The resulting MIDI bootleg score is a $62 \times 3N$ binary matrix, where N is the number of simultaneous note events in the MIDI file. [3] Figure 2 shows an example bootleg score. The staff lines are included as a visualization aid, but are not present in the bootleg feature representation.

2.2 Generating Query Bootleg Score

The second main component of our system (Figure 1) is to convert the cell phone image into a bootleg score representation. Unlike the MIDI representation, the image does not explicitly encode any information about the notes, so we will have to estimate this information from the raw image.

Our general approach rests on two key insights. The first key insight is that we can identify where we are in the piece if we can detect just three things: filled noteheads, staff lines, and bar lines. Because these objects are simple geometrical shapes, classical computer vision tools are sufficient to detect them (Section 2.2.2–2.2.4). The second key insight is that we know a priori that these three objects will occur many times in the image. This opens up the possibility of test-time adaptation, where we can use a very simple notehead detector to identify some of the noteheads in the image, and then use those detected instances to learn a more accurate notehead template at test time. This is

[1] Our code is available at `https://github.com/tjtsai/SheetMidiRetrieval`

[2] It could also appear as a D double flat, but we do not consider double sharps or double flats since they occur relatively infrequently.

[3] The factor of 3 comes from the filler and repetitions.

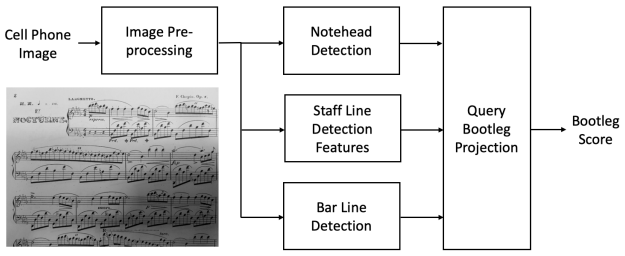


Figure 3. Overview of generating the query bootleg score (Sections 2.2.1–2.2.5). The cell phone image shown will serve as a running example throughout the paper.

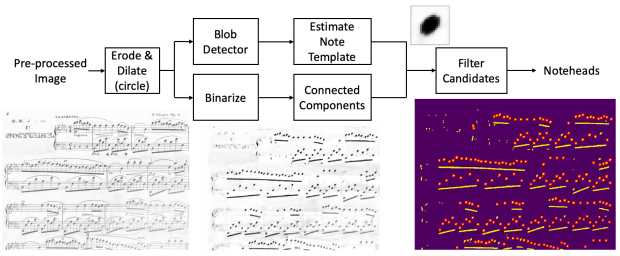


Figure 4. Overview of notehead detection (Section 2.2.2). The images at bottom show the pre-processed image before (left) and after (center) erosion & dilation, and the detected noteheads (right).

generally not possible with large object detection and classification scenarios like the ImageNet competition [6, 20].

Our method for generating the cell phone image bootleg score has five parts, which are shown in Figure 3. These will be described in the next five subsections.

2.2.1 Image Pre-processing

The preprocessing consists of three operations: (1) converting the image to grayscale, (2) resizing the image to a maximum dimension of 1000 pixels while retaining the same aspect ratio, and (3) removing background lighting by blurring the image and then subtracting the blurred image from the non-blurred image.

2.2.2 Notehead Detection

The goal of the notehead detection stage in Figure 3 is to predict a bounding box around every filled notehead in the cell phone image. Note that we do not attempt to detect non-filled noteheads (i.e. half-notes, dotted half notes, whole notes). The basic premise of our approach is that filled noteheads are much easier to detect, and they also generally occur much more frequently than half or whole notes. The notehead detection consists of the steps shown in Figure 4. We will explain these steps in the next four paragraphs.

The first step is to perform erosion and dilation of the pre-processed image with a circular morphological filter. The erosion replaces each pixel with the whitest pixel in a circular region centered around the pixel. This operation removes any objects that consist of thin lines, and it only passes through contiguous dense regions of black pixels. The dilation takes the resulting image and replaces each pixel with the blackest pixel in a circular region center around the pixel. This operation restores any objects that survived the erosion back to their original size. Figure 4 shows an example of an image after erosion and dilation (center image).

Next, we describe the processing in the upper path of Figure 4. We take the eroded and dilated image and apply simple blob detection. We use the simple blob detector in OpenCV with default parameter settings, except that we specify a minimum and maximum area in order to specify the rough size of object we expect. We then take crops of the (eroded and dilated) image around the detected keypoints, and we compute the average of the cropped regions. This average gives us an estimate of what a filled notehead looks like in this image. Figure 4 shows an example of an estimated template (upper right).

Now we describe the processing in the lower path of Figure 4. We take the eroded and dilated image and binarize it using Otsu binarization [19]. We then extract a list of connected component regions from the binary image, which gives us a list of candidate regions, some of which are noteheads.

The last step in notehead detection is to filter the list of candidates using our estimated notehead template. We filter the list of candidates to only contain those regions whose height, width, height-width ratio, and area all roughly match the notehead template (within some tolerance). We also filter the list of candidates to identify chord blocks, which often appear as a single connected component region. When a chord block is identified, we estimate the number of notes in the chord based on its area relative to the notehead template and then perform k-means clustering to estimate individual notehead locations.

At the end of these steps, we have a list of bounding boxes around the detected notes in the cell phone image. Figure 4 (bottom right) shows an example of the predicted notehead locations in an image.

2.2.3 Staff Line Detection Features

The goal of the staff line detection features stage in Figure 3 is to compute a tensor of features that can be used to predict staff line locations in the bootleg projection stage (Section 2.2.5). In a cell phone picture, staff lines may not be straight lines or have equal spacing throughout the image due to the camera angle or camera lens distortions. For these reasons, we estimate staff line locations locally rather than globally. In other words, for every detected notehead, we make a local estimate of the staff line location and spacing in its vicinity.

The staff line detection features are computed in three steps as shown in Figure 5. The first step is to perform erosion and dilation on the image with a short (1 pixel tall), fat morphological filter. This removes everything except for horizontal lines. In practice, we find that there are two types of objects that survive this operation: staff lines and horizontal note beams (e.g. the beam connecting a sequence of sixteenth notes). The second step is

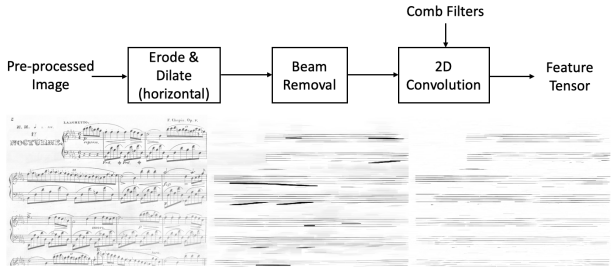


Figure 5. Overview of staff line features computation (Section 2.2.3). The images at bottom show the pre-processed image before (left) and after (middle) erosion & dilation, and the result after removing note beams (right). The actual feature tensor is not shown.

to remove the note beams, as they can throw off the staff line location estimates. Because the note beams are much thicker than staff lines, we can isolate the note beams based on their thickness and subtract them away from the image. The third step is to convolve the resulting image with a set of comb filters. We construct a set of tall, skinny (1 pixel wide) comb filters, where each comb filter corresponds to a particular staff line spacing. The set of comb filters is selected to span a range of possible staff line sizes. We then convolve the image (after beam removal) with each of the comb filters and stack the filtered images into a tensor. This feature tensor $\boldsymbol{T}_{global}$ has dimension $H_{image} \times W_{image} \times N_{comb}$, where H_{image} and W_{image} specify the dimensions of the image and N_{comb} is the number of comb filters. Note that the third dimension corresponds to different staff line spacings.

2.2.4 *Bar Line Detection*

The goal of the bar line detection stage (Figure 3) is to predict a bounding box around the barlines in the cell phone image. The bar lines are needed to correctly cluster staff lines into grand staff systems, where each grand staff consists of a right hand staff and a left hand staff.

The bar line detection consists of the five steps shown in Figure 6. The first step is to perform erosion and dilation of the image with a tall, skinny morphological filter. This filters out everything except vertical lines. In practice, we find that there are three types of objects that survive this operation: bar lines, note stems, and background pixels (e.g. music stand at edges of image). The second step is to binarize the eroded and dilated image using Otsu binarization. The third step is to extract a list of connected component regions from the binary image. The fourth step is to filter this list of candidates to identify bar lines. This can be done by first filtering out regions that are too wide (e.g. background pixel regions), and then distinguishing between note stems and bar lines by finding the threshold on height that minimizes intra-class variance (which is equivalent to Otsu binarization but applied to the heights). The fifth step is to cluster the detected bar lines into lines of music. We do this by simply clustering any bar lines that have any vertical overlap. Figure 6 shows this process for an example image.

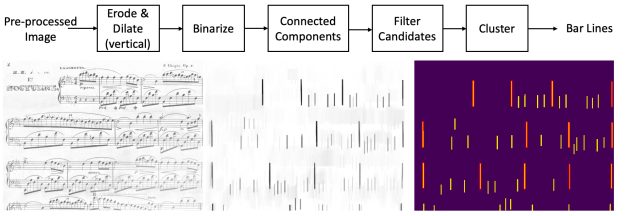


Figure 6. Overview of bar line detection (Section 2.2.4). The images at bottom show the pre-processed image before (left) and after (center) erosion & dilation, and the detected bar lines (right).

At the end of the bar line detection stage, we have a prediction of the number of lines of music in the cell phone image, along with the vertical pixel range associated with each line. Figure 6 shows an example of an image at the various stages of processing in the bar line detection.

2.2.5 *Query Bootleg Projection*

The last step in Figure 3 is to combine the notehead, staff line, and bar line information in order to synthesize a bootleg score for the cell phone image. This bootleg score synthesis consists of the three steps shown in Figure 7.

The first step is to locally estimate the staff line location and spacing for each notehead. We do this by selecting a subset $\boldsymbol{T}_{local}$ of the staff line feature tensor $\boldsymbol{T}_{global}$ which only contains a rectangular context region around the notehead's (x, y) location in the image. This gives us a three-dimensional feature tensor $\boldsymbol{T}_{local}$ with dimension $H_{context} \times W_{context} \times N_{comb}$, where $H_{context}$ and $W_{context}$ specify the size of the context region and N_{comb} specifies the number of comb filters. We calculate the sum of features across the rows of $\boldsymbol{T}_{local}$, and then identify the maximum element in the resulting $H_{context} \times N_{comb}$ matrix. The location of the maximum element specifies the vertical offset of the staff lines, along with the staff line size. Figure 8 shows a visualization of the estimated local staff line predictions for a line of music. Yellow dots correspond to estimated notehead locations, and the red and blue dots are predictions of the top and bottom staff lines.

The second step is to label and cluster detected noteheads. We estimate each notehead's discrete staff line location by applying simple linear regression on its local staff line coordinate system followed by quantization. This is necessary to determine where the notehead should be placed in the bootleg score. We also need to associate each notehead with an upper or lower staff in a specific line of music. To do this, we first check to see if the predicted staff line system is within the vertical range of a valid line of music (see Section 2.2.4) and, if so, if it falls in the upper or lower half of the region. If the predicted staff line system does not fall within a valid line of music, the notehead is ignored and will not appear in the bootleg score. The latter tends to happen with noteheads at the very top or bottom of the image, where a portion of a staff shows up but is cut off by the image boundaries.

The third step is to actually place the noteheads into

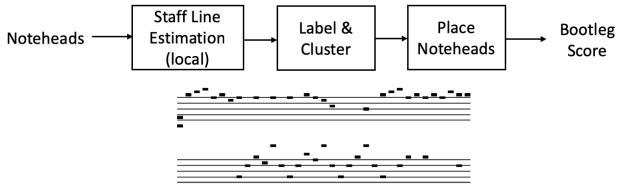


Figure 7. Overview of query bootleg projection (Section 2.2.5). The image at bottom shows part of the generated bootleg score for the cell phone image in Figure 3.

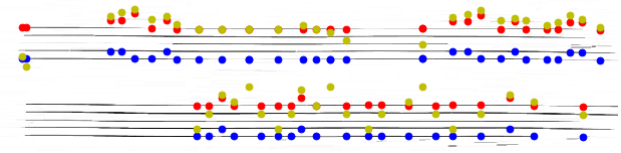


Figure 8. A visualization of local staff line estimation. Each yellow dot corresponds to a detected notehead, and the red and blue dots correspond to the predicted top and bottom staff lines.

the bootleg score. We collapse the noteheads within each valid bar line region into a sequence of simultaneous note events, and then construct the bootleg score as a sequence of simultaneous note events. Similar to the MIDI bootleg score, we repeat each simultaneous note event twice and insert a filler column between each simultaneous note event. Figure 7 shows part of the bootleg score generated from the cell phone image in Figure 3.

2.3 Subsequence DTW

The third main component of our system (Figure 1) is to align the two bootleg scores using subsequence dynamic time warping (DTW). DTW is a well-established dynamic programming technique for determining the alignment between two feature sequences. Subsequence DTW is a variant of DTW that finds the optimal match between a shorter query segment and a subsequence of a (longer) reference segment. For details on DTW and its variants, the reader is referred to [17]. Our cost metric computes the negative inner product between two bixel columns and then normalizes the result by the maximum of (a) the number of simultaneous noteheads in the sheet music and (b) the number of simultaneous note onsets in the MIDI. The inner product counts how many overlapping black bixels there are between the two columns, and the normalization factor ensures that the actual cost is not biased by the number of simultaneous notes. At the end of this stage, we have a prediction of the segment in the MIDI file that best matches the lines of sheet music shown in the cell phone image. This is the final output of our proposed system.

3. EXPERIMENTAL SETUP

The experimental setup will be described in three parts: the data, the annotations, and the evaluation metric.

The data was collected in the following manner. We first download 100 piano scores in PDF format from IMSLP. [4] These piano scores come from 25 well-known composers and span a range of eras and genres within the classical piano literature. To simplify the evaluation, we select scores that do not have any repeats or structural jumps. For each score, we then find a corresponding MIDI file from various online websites. This gives us a total of 100 MIDI-PDF matching pairs. Next, we printed out the PDF scores onto physical paper, placed the sheet music pages in various locations, and took 10 cell phone pictures of each score, spaced throughout the length of the piece. The pictures were taken in various ambient lighting conditions (some of which triggered the flash and some of which didn't), various perspectives, and varying levels of zoom. The pictures capture between 1 and 4 lines of music on a page. We collected the data with two cell phones (iPhone 8, Galaxy S10), and all pictures were taken in landscape orientation. As much as possible, we tried to emulate typical conditions of the application scenario. In total, the data contains 100 MIDI files, 100 scores, and 1000 cell phone images.

The data was manually annotated at the measure level. For the MIDI files, we used `pretty_midi` to programatically estimate the timestamps of the downbeats in each measure, which were then manually verified and corrected. For the cell phone images, we annotated which measures in the score were captured. Since the images would often capture a fragment of a line of music (at the top or bottom), we adopted the convention of only annotating measures on lines of music that are fully captured in the image. For each image, we can use these annotations to determine the matching time segment in the MIDI file.

The metric we use to evaluate our system performance is precision, recall, and F measure. Precision is the temporal duration of overlap between the hypotheses and ground truth segments divided by the total duration of hypothesis segments. Recall is the amount of overlap divided by the total duration of ground truth segments. F measure is then computed as the harmonic mean of precision and recall. In a few situations, the query perfectly matches two different sections in the score. In these situations, we consider any perfectly matching sections of the score to be correct.

4. RESULTS

We evaluate our system in the following manner. We first randomly select 10 out of the 100 scores and set apart their corresponding $10 \times 10 = 100$ cell phone images as the training set. The remaining 900 cell phone images are set apart for testing. Note that this train-test split has an unusually large emphasis on the test data. The reason that we do this is because our system has no trainable weights—only hyperparameters—so the training data is really only used to determine the hyperparameter settings. After doing iterative design on the training data and determining reasonable hyperparameter settings, we froze the system and evaluated it on the 900 test images coming from the 90 unseen music scores.

[4] `https://imslp.org`

System	Data	P	R	F
Random	Test	.152	.189	.169
SharpEye	Test	.413	.091	.150
Photoscore	Test	.692	.681	.687
Bootleg	Test	.900	.840	.869
Bootleg	Train	.872	.869	.871

Table 1. Experimental results. The three rightmost columns show precision (P), recall (R), and F measure (F).

We compare our system to three baselines. The first two baseline systems are Photoscore and SharpEye, which are both commercially available OMR software. We use the software to convert the cell phone image to a (predicted) MIDI representation and then perform subsequence DTW with chroma features. Note that Photoscore and SharpEye were not designed to handle cell phone images, so they would sometimes fail to process the image (i.e. would throw an error). In these situations, we simply mapped errors to a predicted time interval with 0 duration. The third baseline is random (informed) guessing. We calculate the average number (N) of sheet music measures showing in the training images. At test time we randomly select a time interval in the reference MIDI file spanning N measures.

Table 1 shows the performance of our system and the three baseline systems. There are three things to notice about these results. First, the baseline systems all perform poorly. This is not a surprise, since Photoscore and SharpEye were designed to handle sheet music scans, not cell phone images. We would expect that other OMR-based approaches that are trained on scanned sheet music would likewise perform poorly on cell phone images. Second, the bootleg approach far outperforms the baselines. The proposed system achieves an F measure score of .869 on the test set, which is far better than the highest F measure score (.687) among the baseline systems. Third, the proposed system generalizes very well from the training data to the testing data. After iterating and optimizing the system on the training data, the F measure score only fell from .871 (on the training data) to .869 (on the test data). The reason that our system generalizes so well with such a small training data set is that our system has no trainable weights and only about 30 hyperparameters. Even then, many of these hyper parameters are dictated by conventions of Western musical notation for piano music. With such a small number of parameters, we don't expect the system to suffer severely from overfitting, and indeed this is what we observe in our experiments.

5. ANALYSIS

In this section we gain deeper insight into our system through two different analyses.

The first analysis is to manually investigate all of the test queries that were failures. Here, we define a failure as having no overlap at all between the predicted time interval and the ground truth time interval. These are instances where the system simply failed to find a reasonable match. There were two common causes of failure. The biggest cause of failure came from notehead detection mistakes. The notehead detector will obviously fail on half notes and whole notes, since we only try to detect filled noteheads. When the sheet music contains a high fraction of these notes, the system will perform poorly. Also, the system often failed to detect chord blocks where multiple noteheads were located in close proximity to one another. This problem is primarily due to poor hyperparameter settings, and could be mitigated by optimizing the hyperparameters over a larger, more diverse training data set. The second cause of failure were symbols that cause the noteheads to appear in a different place than expected. These include clef changes, octave markings, and trills. Clef changes and octave markings could be incorporated into the MIDI bootleg score by considering all possible clef and octave changes in both right and left hand staves, but there is no immediately obvious way to address the problem of trills.

The second analysis is to characterize run time. Because our application is an online search, the run time is an important consideration. Accordingly, we profiled our system to determine how long it takes to process each query, and to identify the parts of the system that are bottlenecks to improve runtime. Note that our entire system is implemented in python with OpenCV and a custom cython-accelerated subsequence DTW function. When each query is processed by a single 2.1 GHz Intel Xeon processor, the average runtime is 7.6 seconds. When we analyze the breakdown of runtime across the major components of the system, we find that the major bottleneck is the staff line detection features stage (92% of total runtime), which primarily consists of 2-D convolutions with the set of comb filters. This suggests one way to improve runtime: rather than using a large set of comb filters to handle a wide range of possible staff line spacings, we could explicitly estimate the staff line size and consider a much smaller set of comb filters. If we could reduce the set of comb filters by a factor of 10, the average time per query would be 1.3 seconds.

6. CONCLUSION

We explore an application in which a user would like to retrieve a passage of music from a MIDI file by taking a cell phone picture of a physical page of printed sheet music. We develop a proof-of-concept prototype and evaluate its performance on a dataset containing 1000 cell phone pictures of 100 different scores of classical piano music. Our system projects both the MIDI file and the cell phone images into a low-dimensional feature representation called a bootleg score, which explicitly encodes the rules of Western musical notation. We then align the two bootleg scores using subsequence DTW. The most notable characteristic of our system is that it has no trainable weights at all—only a small set of hyperparameters that can be easily tuned on a small training set. Our system generalizes very well from training to testing, and it achieves a test F measure score of .869. We hope that this work serves as an entry point to exploring new ways to retrieve various forms of music using cell phone images as a query.

7. ACKNOWLEDGMENTS

We would like to thank the Class of 1989 Summer Experiential Learning Fund, the Vandiver Summer Experiential Learning Fund, and the Norman F. Sprague III, M.D. Experiential Learning Fund established by the Jean Perkins Foundation for their generous support.

8. REFERENCES

[1] Adrià Rico Blanes and Alicia Fornés Bisquerra. Camera-based optical music recognition using a convolutional neural network. In *IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 27–28. IEEE, 2017.

[2] Hoang-Nam Bui, In-Seop Na, and Soo-Hyung Kim. Staff line removal using line adjacency graph and staff line skeleton for camera-based printed music scores. In *International Conference on Pattern Recognition*, pages 2787–2789. IEEE, 2014.

[3] Jorge Calvo-Zaragoza and David Rizo. Camera-primus: Neural end-to-end optical music recognition on realistic monophonic scores. In *Proc. of the International Conference on Music Information Retrieval (ISMIR*, pages 23–27, 2018.

[4] David Damm, Christian Fremerey, Frank Kurth, Meinard Müller, and Michael Clausen. Multimodal presentation and browsing of music. In *Proc. of the International Conference on Multimodal Interfaces (ICMI)*, pages 205–208, Chania, Crete, Greece, October 2008.

[5] David Damm, Christian Fremerey, Verena Thomas, Michael Clausen, Frank Kurth, and Meinard Müller. A digital library framework for heterogeneous music collections: From document acquisition to cross-modal interaction. *International Journal on Digital Libraries*, 12(2-3):53–71, 2012.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

[7] Matthias Dorfer, Andreas Arzt, Sebastian Böck, Amaury Durand, and Gerhard Widmer. Live score following on sheet music images. In *Late Breaking Demo at the International Conference on Music Information Retrieval (ISMIR)*, 2016.

[8] Matthias Dorfer, Andreas Arzt, and Gerhard Widmer. Towards score following in sheet music images. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 789–795, New York City, New York, USA, 2016.

[9] Matthias Dorfer, Andreas Arzt, and Gerhard Widmer. Learning audio-sheet music correspondences for score identification and offline alignment. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 115–122, Suzhou, China, 2017.

[10] Matthias Dorfer, Florian Henkel, and Gerhard Widmer. Learning to listen, read, and follow: Score following as a reinforcement learning game. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 784–791, 2018.

[11] Matthias Dorfer, Jan Hajič Jr., Andreas Arzt, Harald Frostel, and Gerhard Widmer. Learning audio-sheet music correspondences for cross-modal retrieval and piece identification. *Transactions of the International Society for Music Information Retrieval*, 1(1):22–33, 2018.

[12] Matthias Dorfer, Jan Schlüter, Andreu Vall, Filip Korzeniowski, and Gerhard Widmer. End-to-end cross-modality retrieval with cca projections and pairwise ranking loss. *International Journal of Multimedia Information Retrieval*, 7(2):117–128, 2018.

[13] Christian Fremerey, Michael Clausen, Sebastian Ewert, and Meinard Müller. Sheet music-audio identification. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 645–650, Kobe, Japan, October 2009.

[14] Christian Fremerey, Meinard Müller, Frank Kurth, and Michael Clausen. Automatic mapping of scanned sheet music to audio recordings. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 413–418, Philadelphia, USA, September 2008.

[15] Özgür İzmirli and Gyanendra Sharma. Bridging printed music and audio through alignment using a mid-level score representation. In *Proc. of the International Conference on Music Information Retrieval (ISMIR*, pages 61–66, 2012.

[16] Frank Kurth, Meinard Müller, Christian Fremerey, Yoon-Ha Chang, and Michael Clausen. Automated synchronization of scanned sheet music with audio recordings. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 261–266, Vienna, Austria, September 2007.

[17] Meinard Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer, 2015.

[18] Meinard Müller, Andreas Arzt, Stefan Balke, Matthias Dorfer, and Gerhard Widmer. Cross-modal music retrieval and applications: An overview of key methodologies. *IEEE Signal Processing Magazine*, 36(1):52–62, 2019.

[19] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[21] Verena Thomas, Christian Fremerey, Meinard Müller, and Michael Clausen. Linking sheet music and audio – challenges and new approaches. In Meinard Müller, Masataka Goto, and Markus Schedl, editors, *Multimodal Music Processing*, volume 3 of *Dagstuhl Follow-Ups*, pages 1–22. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2012.

[22] Quang Nhat Vo, Soo Hyung Kim, Hyung Jeong Yang, and Gueesang Lee. An mrf model for binarization of music scores with complex background. *Pattern Recognition Letters*, 69:88–95, 2016.

[23] Quang Nhat Vo, Tam Nguyen, Soo-Hyung Kim, Hyung-Jeong Yang, and Guee-Sang Lee. Distorted music score recognition without staffline removal. In *International Conference on Pattern Recognition*, pages 2956–2960. IEEE, 2014.

A CONVOLUTIONAL APPROACH TO MELODY LINE IDENTIFICATION IN SYMBOLIC SCORES

Federico Simonetta[1] **Carlos Cancino-Chacón**[2] **Stavros Ntalampiras**[1]
Gerhard Widmer[3,2]
[1] Music Informatics Laboratory, Dept. of Computer Science, University of Milano, Italy
[2] Austrian Research Institute for Artificial Intelligence, Vienna, Austria
[3] Dept. of Computational Perception, Johannes Kepler University Linz, Austria
federico.simonetta@unimi.it, carlos.cancino@ofai.at

ABSTRACT

In many musical traditions, the melody line is of primary significance in a piece. Human listeners can readily distinguish melodies from accompaniment; however, making this distinction given only the written score – i.e. without listening to the music performed – can be a difficult task. Solving this task is of great importance for both Music Information Retrieval and musicological applications. In this paper, we propose an automated approach to identifying the most salient melody line in a symbolic score. The backbone of the method consists of a convolutional neural network (CNN) estimating the probability that each note in the score (more precisely: each pixel in a piano roll encoding of the score) belongs to the melody line. We train and evaluate the method on various datasets, using manual annotations where available and solo instrument parts where not. We also propose a method to inspect the CNN and to analyze the influence exerted by notes on the prediction of other notes; this method can be applied whenever the output of a neural network has the same size as the input.

1. INTRODUCTION

Many musical traditions make use of melody-accompaniment structures. Generally, the melody line carries the most significant meaning, while the accompaniment provides harmonic and rhythmic support.

In Western art music – which, unlike music in some other traditions, is typically notated – special attention is paid to the construction of melodies during composition. Ideally, melodies in Western art music styles should involve an intervallic structure that is dependent on the specific tonal hierarchy defined by the piece [24, 26]. Musicians typically accentuate melody lines during performance as a way of clarifying the piece structure for listeners: for example, melody lines may be played louder and with more flexible timing than accompaniment [9, 10].

Most listeners readily distinguish melody lines from accompaniment. In contrast, identifying the melody line through visual inspection of a musical score – without hearing the piece – can be a difficult task, even for trained musicians [3]. In this paper, we propose a convolutional approach for identifying the melody line of a piece using a piano roll representation of the score. A solution for this task has potential implications for music information retrieval and musicology [27]. An effective algorithm could be applied to music retrieval tasks such as query-by-humming, searching a database of MIDI files for melodies, developing performance models that account for melody in predicting musical expression, etc. Our focus is on music of the common practice period that uses melody-dominated homophonic textures (i.e., a single melody line plus accompaniment lines), rather than equal-voice polyphony (i.e., multiple independent melody lines) or monophony (i.e., unison melody shared by all voices). However, we provide extensive tests of the proposed method in styles other than common practice era, such as pop, baroque and contemporary art music.

The rest of this paper is structured as follows: In Section 2, we discuss related work on voice separation and streaming. Section 3 briefly describes the baseline methods that we used for comparison against our model. Section 4 presents a description of the proposed method. Section 5 describes the three datasets used in this work. Section 6 describes the experimental evaluation of the proposed method. Section 7 discusses the results of the experimental evaluation. Finally, Section 8 concludes this paper and proposes some future research directions. A companion website was also created to show additional material for the sake of reproducibility. [1]

2. RELATED WORK

2.1 Voices and Streams

Music perception research has investigated listeners' abilities to distinguish between voices in homo- and poly-

[1] https://limunimi.github.io/Symbolic-Melody-Identification/

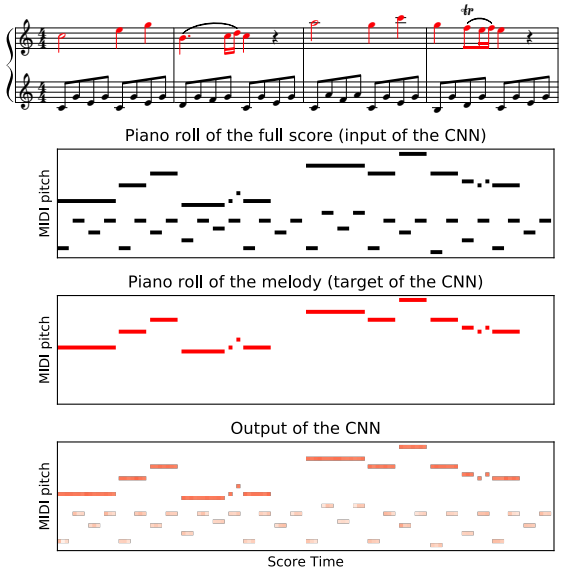


Figure 1. Top: Excerpt of Mozart's Sonata K. 545 (melody highlighted in red). Middle: Piano roll representation of the score (melody is highlighted in red). Bottom: Prediction of the CNN for this excerpt. In this piano roll, the intensity of the color of each pixel represents its probability of belonging to the melody.

phonic music, and has shown that the theoretical rules of voice leading are motivated by listeners' abilities to follow voices [15]. Cambouropoulos [4] proposed three ways of defining musical "voices": (1) for multi-instrument music, each instrument can be said to constitute a separate voice; this would allow for the possibility of non-monophonic voices in instruments that produce chords; (2) voices can be assigned to melodic streams as they are perceived and segmented by listeners, following cognitive grouping principles; and (3) in monophonic music, the harmonic content of the piece may imply a horizontal organization of polyphonic voices that unfold over time (i.e., implied polyphony), e.g., multiple temporally-overlapping voices could be assigned to passages of Bach's Cello Suites. In this work, we use the second definition, and we define the melody line as the most salient voice.

In the music information retrieval literature, three corresponding tasks have been addressed: 1) voice separation from symbolic scores [6, 11, 13, 21]; 2) main track identification (from MIDI files with multiple tracks) [8, 14, 19, 20]; and 3) main melody identification from audio [1, 2, 25]. The latter is a different problem than that addressed here: it deals with the complex task of identifying notes from an audio file, but can use performance cues (e.g., contrasts in timbre and dynamics, which are not present in MIDI data) to facilitate melody identification.

Most relevant to the current study is the task of voice separation from symbolic scores. Some of the proposed methods are computational implementations that attempt to capture perceptual rules of segmentation [4, 6, 11–13] – in particular those rules codified by Huron [15]. For a more in-depth discussion on voice separation algorithms from symbolic scores, we refer the reader to [7, 12, 17, 29].

3. BASELINE METHODS

3.1 Skyline Algorithm

The skyline algorithm is a heuristic that takes the highest note at each point in time [5, 28]. In Western art music, pop and many folk traditions from around the world, melodies are often carried by the highest voice. After the submission of this paper, we discovered that a new method was being submitted for this same task [18], confirming the relevance of this topic.

3.2 VoSA

Proposed by Chew and Wu [6], VoSA is a successful voice separation method. In this approach, a piece is split into segments based on voice entry and exit points, so that the number of sounding notes is constant within each segment. The segment with the highest number of sounding notes defines the number of voices in the piece. Notes are then connected into voices using connection weights, equal to the absolute size of the interval between one note and the next. Like most voice separation methods, VoSA was designed to work with polyphonic rather than homophonic music. In spite of its apparent simplicity, VoSA has been favorably compared against more sophisticated computational models of voice separation [12, 13, 21].

4. METHOD

4.1 Music Score Modeling Using CNNs

A schematic representation of our method is given in Figure 2. The backbone of the method consists of a fully convolutional neural network (shown in Figure 3), which takes as input segments of a music score, represented as a piano roll, and estimates the probability that each note in the score (more precisely: each pixel in the piano roll encoding) belongs to the melody line.

A piano roll can be described as a 2D representation of a musical score; the x-axis indicates score time and the y-axis indicates pitch. The piano rolls used in this study are constructed with a temporal resolution of 8 pixels/beat (i.e., a pixel represents a 32nd note in $\frac{4}{4}$). The piano roll of each piece is divided into overlapping fixed-length windows of 64 pixels (i.e., 8 beats). The length of the window was determined using hyper-parameter optimization, (see Section 6.2). The overlap between windows is 50% (i.e., 2 beats), and windows shorter than this size are padded with zeros. An output piano roll for each full piece is constructed by averaging probabilities for the pixels located in overlapping windows. Afterwards, we apply a mask on the output piano roll by multiplying it by the (binary) input piano roll, so that areas with no notes take values of zero, and non-zero probabilities only remain where there are notes. The probability of each note belonging to the melody is then calculated as the median across the output values of its pixels. In the following discussion, we will use *note*

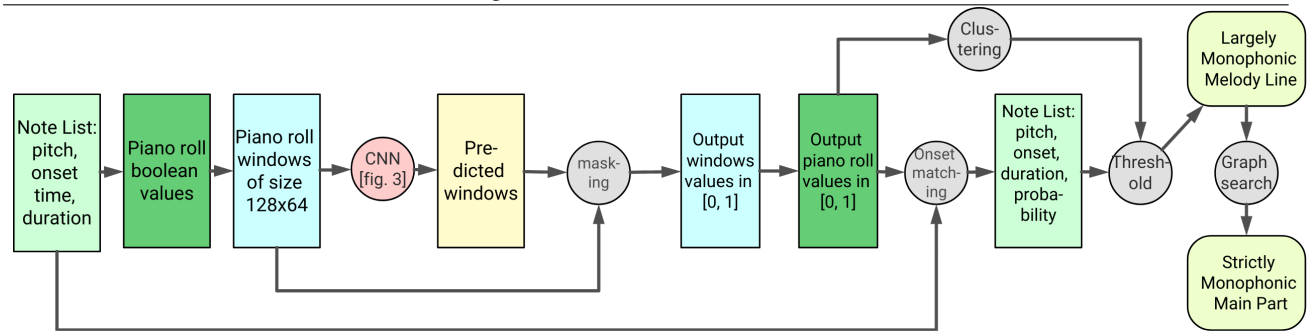


Figure 2. The pipeline of the proposed method (see Section 4).

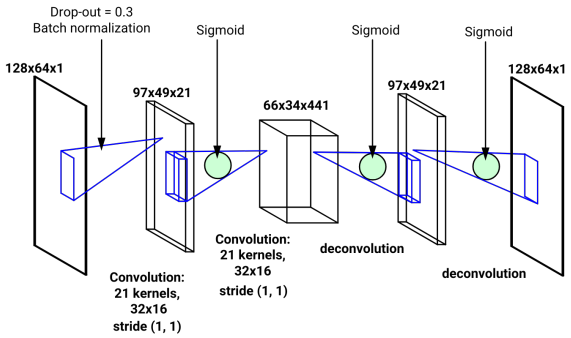


Figure 3. The architecture of the fully convolutional neural network used in the proposed method. The architecture of the network was determined using hyper-parameter optimization (see Section 6.2 for explanation).

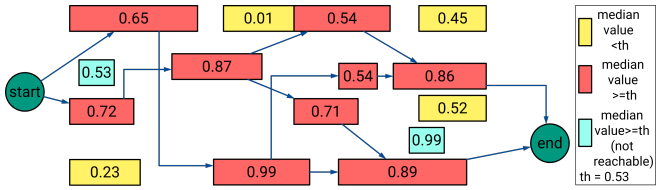


Figure 4. Example of graph built with Algorithm 1. Red notes are notes over threshold, yellow notes are under threshold, while blue notes are over threshold but are not reached by any path. The green circles are the starting and ending nodes. Numbers indicate note probabilities, which are computed as the median of their pixels.

probability as a shorthand to refer to the probability of a note to belong to the melody.

In Figure 1 we show an excerpt of Mozart's Piano Sonata K. 545 and three vertically aligned piano rolls corresponding to the excerpt. The second row of this figure is the input piano roll, while the third row gives the ground truth melody line that we aim to identify in the input. The bottom gives the piano roll that we obtain as output. The output is color-coded with the notes that were identified as melody highlighted in red.

A threshold is needed to determine which note probabilities should indicate melody notes. Distributions of probabilities differ between pieces, so a hard threshold (e.g. 0.5) would be inappropriate. Instead, we find a threshold for each piece using a statistical analysis of the values of the note probabilities. In the implementation of the proposed method we use hierarchical *single-linkage* clustering [23]: two clusters across the values of note probabilities are identified, and a piece-wise threshold is selected as the largest value of the lowest cluster. We then compare each note probability to this threshold and either retain the note as melody or filter it out as accompaniment. This produces largely (but not entirely) monophonic melody output – in some cases, multiple simultaneous notes pass the threshold. A graph-based method, explained next, was thus implemented to select a strictly monophonic melody line from this output.

4.2 Graph Search

Having identified notes that pass the threshold as defined above, we have to select a sequence of these notes that maximizes the probability of the sequence being monophonic. This is achieved using a graph-based approach. Algorithm 1 is used to build a *directed acyclic graph* (or digraph, see Figure 4). Such a graph consists of a set of nodes and a set of directed edges. Each of these edges specifies a connection from a node to another. In the graph defined by Algorithm 1, each note that passes the threshold is represented by a node, and the pitch, onset and duration information of this note are used to determine to which nodes is the note connected (in order to guarantee a strictly monophonic sequence). Note probabilities are used to determine the strength of the connection between nodes (similar to a "distance"; notes with high probabilities are considered "closer"). Additionally, we set a start and end node at the beginning and end of the piece, respectively. We can then use a single-source shortest path algorithm to find the main melody line as the shortest path from the start to the end nodes. In our current implementation, we use the negative note probabilities as connection weights and the Bellman-Ford algorithm [2] to find the shortest path through the graph. [3]

[2] `https://docs.scipy.org/doc/scipy-1.2.1/reference/generated/scipy.sparse.csgraph.bellman_ford.html`

[3] Depending on the choice of the connection weights, other shortest path algorithms (e.g., topological sorting, Dijkstra's, etc.) are possible.

Algorithm 1 Melo-digraph building

$L \leftarrow$ list of notes
$\alpha \leftarrow$ starting node (end time $= 0$)
$\omega \leftarrow$ ending node (onset $= \infty$, probability $= -0.5$)
Push α to the beginning of L
Push ω to the end of L
for $note$ in L **do**
  $L' \leftarrow$ notes with onset $\geq$ end time of $note$
  $L' \leftarrow$ notes with onset $=$ minimum onset in L'
  for $note'$ in L' **do**
    if probability of $note' \geq$ threshold **then**
      $p =$ probability of $note'$
      add an edge $(note, note')$ with weight $-p$
    end if
  end for
end for

4.3 Training

The CNN is trained in a supervised fashion to filter out accompaniment parts. Inputs are provided in the form of piano roll segments and the targets are the corresponding piano rolls with only the melody notes. We also augmented the training dataset by 50% by creating copies of the original examples in the dataset with the melody transposed down for 2 octaves or up for 1 octave. Though the standard loss function for binary classification problems like this one is the binary cross entropy, during development of the model, we achieved more accurate models by minimizing mean squared error for the match between output and target piano rolls. The networks were trained using AdaDelta [30] with initial learning rate set to 1. In order to avoid overfitting, we use dropout with probability $p_{dropout} = 0.3$ and L_1-norm weight regularization. Additionally we use batch-normalization [16]. The training is stopped after 20 epochs without improvement in validation loss [22] .

5. DATASETS

We used three different datasets to evaluate the performances of our method. The first dataset ("Mozart") consists of 38 movements from (13) Mozart Piano Sonatas, for which the main melody line was annotated manually by a professional pianist. The second dataset ("Pop") consists of 83 popular songs (including pop and jazz). We used the vocal part of these songs as the melody line, and treated them as though they were compressed onto a single track (identifying the main track in multi-track music is a separate question, see [8, 14, 19, 20]).

These datasets were used for training and testing. A third dataset ("Web"), used only for testing, comprises MIDI files crawled from the web. This dataset includes 169 Western art music compositions from the late 16th to the early 20th centuries. All of these pieces included a solo instrument (typically voice, flute, violin or clarinet) and accompaniment (typically strings or piano).

The first and third of these datasets are publicly available for research purposes in the companion site – see footnote 1. We do not have distribution rights for the second dataset, which was professionally curated and annotated, but we provide the full list of pieces.

6. EXPERIMENTS

6.1 Evaluation Metrics and Baseline Methods

In all experiments, we evaluated the quality of the predictions using the F-measure. We experimented on the largely monophonic (which we denote *cnn* in the following discussion) and strictly monophonic (denoted as *cnn mono*) variants of the proposed model described in Sections 4.1 and 4.2, respectively. As a baseline comparison, we used the skyline algorithm and VoSA (both described in Section 3). Since VoSA does not directly output the melody line, we first separate the piece into individual voices (as identified by VoSA), then select the voice with the highest F-measure as the melody. These modifications allowed us to consider the best case scenario of VoSA.

6.2 Network Architecture

To determine the architecture of the network, we used hyper-parameter optimization. [4] The number of convolutional layers, kernel size and number, and window lengths were optimized. This hyper-parameter optimization was done on 100 pieces randomly selected from across the three datasets plus 65 MIDI files collected online using the same criteria as the Web dataset. To compare models, we constructed training, validation, and test sets from the 100 pieces. A model configuration was selected that performed most successfully on the test set. The selected network architecture is shown in Figure 3: 2 convolutional layers, each with 21 kernels of size 32×16 (i.e., over two and a half octaves in the pitch dimension and 2 beats in the time dimension).

6.3 Evaluation of the Proposed Method

To evaluate the quality of the predictions of the proposed method, we conducted two experiments. In the first experiment, we were interested in evaluating the predictive accuracy of the models trained on different datasets. In the second experiment we tested how well models generalize to different music styles. For the first experiment, we performed a 10-fold cross-validation on each of the Mozart and Pop datasets. In each of these cross-validations, the dataset was split into 10 folds. The model was trained on 9 of these folds and tested on the remaining one. We did this for all possible combinations so that each piece in each dataset appeared in the test set once. For the second experiment, we tested models trained on Mozart and models trained on Pop on the Web dataset.

[4] Using the "hyperopt" library in Python (`http://hyperopt.github.io/hyperopt/`).

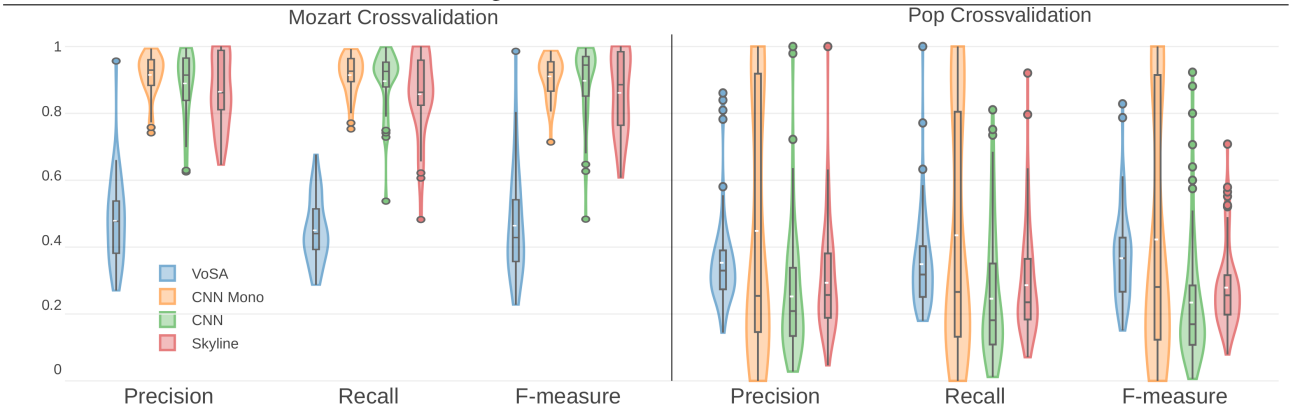


Figure 5. Cross-validation on Mozart and Pop datasets. With the Wilcoxon test applied to F-measure, we found a significant difference between *CNN Mono* and *VoSA* and between *CNN Mono* and *CNN*, but no significant difference was found between *CNN Mono* and *Skyline* in the Pop dataset (only in the Mozart dataset). The mean is marked with a white dash.

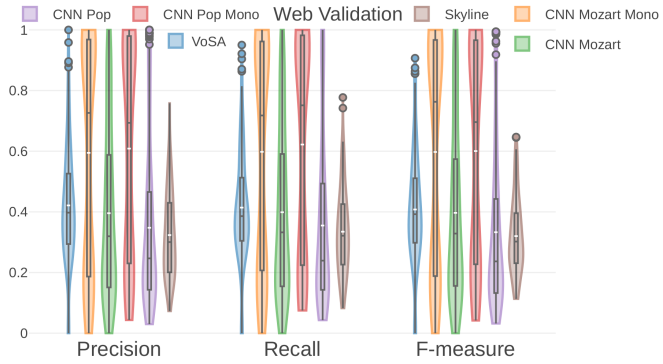


Figure 6. Validation on the Web music dataset. With the Wilcoxon test, we found a significant difference between *Mono* models and *Skyline*/*VoSA*, but there was not always a significant difference when comparing non-*Mono* models and *Skyline*/*VoSA*. The mean is marked with a white dash.

7. RESULTS AND DISCUSSION

7.1 Model Performance

The violin plots summarizing the results of these experiments are shown in Figures 5 and 6, while detailed plots are available in the companion website (see footnote 1).

Our first experiment tested how well models predicted melody lines given training and testing on the same genre of music. Wilcoxon signed-rank tests were run on F-measures to assess potential differences between models. Test results are described in the caption of Figure 5. Overall, our proposed method that identified strictly monophonic melody lines (*cnn mono*) performed better than the other models, but this difference was only significant for the Mozart dataset. The Mozart pieces are highly structured and their melody lines tend to occur in the uppermost voice. The Pop dataset, in contrast, contains pieces with variable structure, with longer breaks in the melody (e.g., there is sometimes an interlude in the accompaniment part). Furthermore, the accompaniment part often overlaps in register with the melody line. It seems that without additional timbral information, our model could not sufficiently distinguish between melody and accompaniment lines when they shared a similar texture.

Our second experiment tested how well trained models generalize to new types of data (i.e., Web dataset). We hypothesized that models trained on the Mozart dataset would outperform models trained on the Pop dataset, as the Mozart and Web datasets are more similar in style (though the Web dataset is more heterogeneous). However, no significant difference between models was found – both models performed well on the Web dataset.

Regarding the less-successful performance of the two baseline methods, the skyline method fails when the melody is not the highest voice; furthermore, this method cannot identify when pauses occur in the solo part. The VoSA method, which was developed for use with polyphonic music, tends to create too many voices and shows a bias towards connecting notes separated by small intervals – this is not surprising, as polyphonic music tends to assign voices to small pitch ranges. As a result, accompaniment notes are often wrongly included in the melody line that VoSA identifies.

7.2 Saliency Maps

To investigate what the CNNs are learning, we propose a method (similar to a sensitivity analysis) that evaluates the contribution of individual locations of the piano roll to predictions at other locations using saliency maps. [5] The method involves testing how the probability that a given note belongs to the melody changes (i.e., increases or decreases) when certain other notes are removed (i.e., by converting the pixels belonging to those notes to 0).

For example, take a rectangular input window I and its prediction P. A new input window I' with prediction P' is created by converting the pixels inside a given rectangle R to 0. The difference between the original and new predictions is denoted as $d(P, P')$ and can be interpreted as the contribution given by the notes inside R to the original

[5] Kernels, saliency maps and additional material are available on the companion website – see footnote 1.

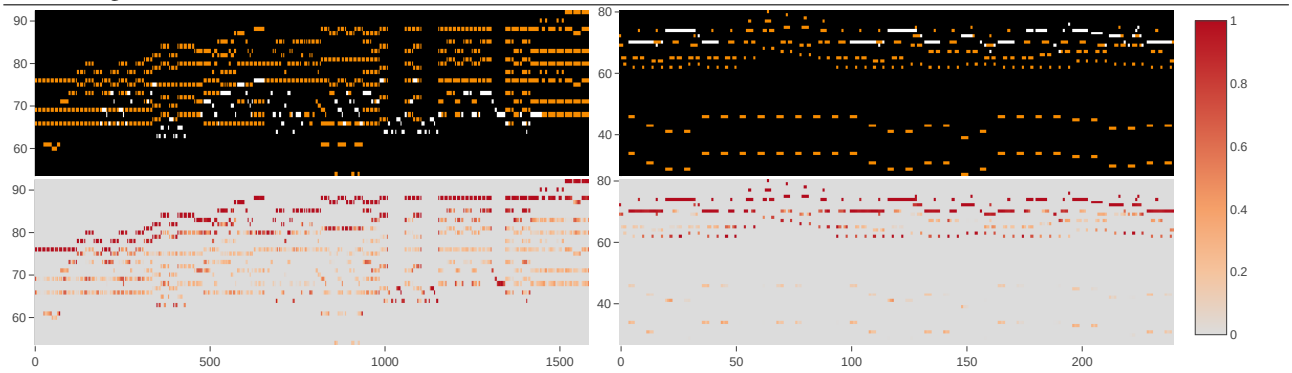


Figure 7. Liszt's *Ihr Glocken von Marling* (left) and an excerpt from Schubert's *Ave Maria* (right). Input piano roll (above), prediction of the CNN (middle). In Liszt, the model fails to identify the main part because the texture is rather different from the most common case and the melody is in the middle voices. In Schubert, instead, the texture changes but the model is not able to identify when the main part starts and stops because the accompaniment plays similar notes.

prediction. By testing different input windows across the piano roll, we can see how different elements of the music contribute to the predictions that are obtained for individual notes.

If we are interested in a particular note n, we can compute $d(P, P')$ specifically for the pixels belonging to n. For our analysis, for certain notes of interest, we define 5 randomly-positioned rectangles R and calculate $d(P, P')$. This difference is summed to the pixels of the notes inside each rectangle R. This procedure is repeated N times (where N is a trade-off between computational complexity and resolution of the saliency map; in our case $N = 30000$), and we select only the iterations in which the pixels of note n are not converted to 0. Each pixel is then normalized by the number of times it was converted to 0. As difference we use

$$d(P, P') = \frac{\sum_{i=n_{start}}^{i=n_{end}} P[i] - P'[i]}{Area(n_{start}, n_{end})} \quad (1)$$

where n_{start} and n_{end} identify the region occupied by the note n. In general, n_{start} and n_{end} indicate two opposite corners of any rectangle.

With this difference function, given a rectangle R, if $d(P, P') > 0$, then $P > P'$ in average across n and, thus, removing the notes inside R decreases the prediction values of n; conversely, if $d(P, P') < 0$, then $P < P'$ and removing the notes inside R increases the prediction.

For example, in the bottom piano roll in Figure 8, the blue high-pitched notes occurring around beats 20 and 35 have non-positive saliency values. Because they are higher pitched, these notes contribute negatively to the melody note highlighted with a green box, making it unlikely for this note to be identified as melody. In the companion website, we show the saliency of other regions highlighting that the prediction of some notes is influenced positively by some regions and negatively by others and that the CNN exploits the regular patterns in the accompaniment to identify the melody notes.

Overall, our model incorporates features of both the skyline algorithm and VoSA. Like the skyline algorithm it focuses on the highest notes of the piece; on the other hand, by allowing for different probabilities like VoSA, it is more successful at drawing coherent melody lines. Unlike VoSA, however, our model does not incorporate explicit perceptual constraints.

8. CONCLUSIONS

We implemented and analyzed a novel method to identify the melody line in a symbolic music score. Some of the functions of our model were found to be similar to functions of the skyline algorithm and VoSA (in particular, focusing on the upper-most pitch, and defining a melody line as finding the sequence of notes that minimizes the connection cost). However, our method does not take into account the long-term sequential nature of music; it can compute windows in any order. While such a property might have some practical benefits, it also makes the network unable to generalize to diverse textures, leading to poor results when musical texture is varied (e.g., Figure 7).

The next step for this line of research would be to develop a model that can take into account a larger temporal context. A promising approach would be to incorporate attention mechanisms into the network.

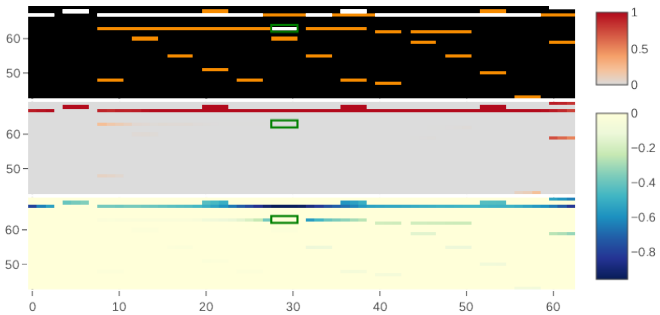


Figure 8. Input piano roll with ground truth in white (top), prediction of the CNN (middle) and proposed saliency computed with respect to the green rectangle (bottom).

9. ACKNOWLEDGEMENTS

This research has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No. 670035 (project "Con Espressione"). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research. We thank Elaine Chew for sharing the code for VoSA. We thank Laura Bishop for proofreading an earlier version of this manuscript.

10. REFERENCES

[1] Rachel M. Bittner, Justin Salamon, Slim Essid, and Juan Pablo Bello. Melody Extraction by Contour Classification. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, Malaga, Spain, 2015.

[2] Juan J. Bosch, Rachel M. Bittner, Justin Salamon, and Emilia Gómez. A Comparison of Melody Extraction Methods Based on Source-Filter Modelling. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, New York, New York, USA, 2016.

[3] Warren Brodsky, Avishai Henik, Bat-Sheva Rubinstein, and Moshe Zorman. Auditory imagery from musical notation in expert musicians. *Perception & Psychophysics*, 65(4):602–6012, 2003.

[4] Emilios Cambouropoulos. Voice and Stream: A Perceptual and Computational Modeling of Voice Separation. *Music Perception*, 26(1):75–94, 2008.

[5] Wei Chai and Barry Vercoe. Melody retrieval on the web. In Martin G. Kienzle and Prashant J. Shenoy, editors, *Multimedia Computing and Networking 2002*. SPIE, dec 2001.

[6] Elaine Chew and Xiaodan Wu. Separating voices in polyphonic music: A contig mapping approach. In Uffe Kock Wiil, editor, *Computer Music Modeling and Retrieval*, pages 1–20, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[7] Reinier de Valk and Tillman Weyde. Deep Neural Networks with Voice Entry Estimation Heuristics for Voice Separation in Symbolic Music Representations. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR 2018)*, pages 281–288, Paris, France, 2018.

[8] Anders Friberg and Sven Ahlbäck. Recognition of the main melody in a polyphonic symbolic score using perceptual knowledge. *Journal of New Music Research*, 38(2):155–169, 2009.

[9] Werner Goebl. Melody lead in piano performance: Expressive device or artifact? *The Journal of the Acoustical Society of America*, 110(1):563–572, 2001.

[10] Werner Goebl. Geformte Zeit in der Musik. In W Kautek, R Neck, and H Schmidinger, editors, *Zeit in den Wissenschaften*, volume 19, pages 179–199. Böhlau Verlag, Wien, Köln, Weimar, 2016.

[11] Patrick Gray and Razvan C. Bunescu. A neural greedy model for voice separation in symbolic music. In Michael I. Mandel, Johanna Devaney, Douglas Turnbull, and George Tzanetakis, editors, *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*, pages 782–788, 2016.

[12] Nicolas Guiomard-Kagan, Mathieu Giraud, Richard Groult, and Florence Levé. Comparing Voice and Stream Segmentation Algorithms. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, Malaga, Spain, 2015.

[13] Nicolas Guiomard-Kagan, Mathieu Giraud, Richard Groult, and Florence Levé. Improving Voice Separation by Better Connecting Contigs. In Michael I. Mandel, Johanna Devaney, Douglas Turnbull, and George Tzanetakis, editors, *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*, pages 164–170, 2016.

[14] Zhi-gang Huang, Chang-le Zhou, and Min-juan Jiang. Melodic Track Extraction for MIDI. *Journal of Xiamen University (Natural Science)*, 1, 2010.

[15] David Huron. Tone and Voice: A Derivation of the Rules of Voice-Leading from Perceptual Principles. *Music Perception*, 19(1):1–64, 2001.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[17] Cihan Isikhan and Giyasettin Ozcan. A Survey of Melody Extraction Techniques for Music Information Retrieval. In *Proceedings of the Fourth Conference on Interdisciplinary Musicology (CIM08)*, Thessaloniki, Greece, 2008.

[18] Zheng Jiang and Roger B. Dannenberg. Melody Identification in Standard MIDI Files. In *Proceedings of the 16th Sound & Music Computing Conference*, pages 65–71, 2019.

[19] Jiangtao Li, Xiaohong Yang, and Qingcai Chen. MIDI melody extraction based on improved neural network. In *Proceedings of the 2009 International Conference on Machine Learning and Cybernetics*, volume 2 of *Machine Learning and Cybernetics, 2009 International Conference on*, pages 1133–1138. IEEE, July 2009.

[20] Raúl Martín, Ramón A. Mollineda, and Vicente García. Melodic track identification in MIDI files considering the imbalanced context. In Helder Araújo, Ana Maria Mendonça, Armando J. Pinho, and M. Inés Torres, editors, *Pattern Recognition and Image Analysis, 4th Iberian Conference, IbPRIA 2009, Póvoa de Varzim, Portugal, June 10-12, 2009, Proceedings*, volume 5524 LNCS of *Lecture Notes in Computer Science*, pages 489–496. Springer, 2009.

[21] Andrew McLeod and Mark Steedman. HMM-Based Voice Separation of MIDI Performance. *Journal of New Music Research*, 45(1):17–26, 2016.

[22] N. Morgan and H. Bourlard. Generalization and parameter estimation in feedforward nets: Some experiments. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 630–637. Morgan-Kaufmann, 1990.

[23] Daniel Müllner. fastcluster: Fast hierarchical, agglomerative clustering routines for R and Python. *Journal of Statistical Software*, 53(9):1–18, 2013.

[24] Walter Piston. *Counterpoint*. W. W. Norton & Company, 1947.

[25] Justin Salamon, Rachel M. Bittner, Jordi Bonada, Juan J. Bosch, Emilia Gómez, and Juan Pablo Bello. An Analysis/Synthesis Framework for Automatic F0 Annotation of Multitrack Datasets. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China, 2017.

[26] Felix Salzer and Carl Schachter. *Counterpoint in Composition*. Columbia University Press, New York, NY, USA, 1989.

[27] F. Simonetta, S. Ntalampiras, and F. Avanzini. Multimodal music information processing and retrieval: Survey and future challenges. In *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*, pages 10–18, Jan 2019.

[28] Alexandra L. Uitdenbogerd and Justin Zobel. Manipulation of music for melody matching. In Wolfgang Effelsberg and Brian C. Smith, editors, *Proceedings of the 6th ACM International Conference on Multimedia '98, Bristol, England, September 12-16, 1998.*, pages 235–240. ACM, 1998.

[29] Frans Wiering, Justin de Nooijer, Anja Volk, and Hermi J. M. Tabachneck-Schijf. Cognition-based Segmentation for Music Information Retrieval Systems. *Journal of New Music Research*, 38(2):139–154, 2009.

[30] Matthew D Zeiler. AdaDelta: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*, 2012.

Author Index