Delft University of Technology
Master of Science Thesis in Embedded Systems

Symbolic Guitar Music Style Transfer with Playable Guitar Tablatures

Xuanyu Zhuang

Embedded Systems

TU Delft Delft University of Technology

Symbolic Guitar Music Style Transfer with Playable Guitar Tablatures

Master of Science Thesis in Embedded Systems

Multimedia Computing Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Xuanyu Zhuang

September 22nd, 2023

Author
Xuanyu Zhuang

Title
Symbolic Guitar Music Style Transfer with Playable Guitar Tablatures

MSc Presentation Date
September 22nd, 2023

Graduation Committee

Cynthia Liem	Delft University of Technology
Jorge Martinez Castaneda	Delft University of Technology
Chirag Raman	Delft University of Technology

Abstract

In the task of music style transfer, the symbolic music representation based on Musical Instrument Digital Interface (MIDI) files has always been a popular research medium. By using such representation, some mature models for image style transfer can also be applied to this scenario, such as Cycle-consistent Generative Adversarial Networks (CycleGAN). However, this MIDI-based data representation is not suitable for guitar music because it does not support unique expressive information of guitar playing, such as bending, sliding, or other playing techniques. DadaGP, a dataset made up of guitar-specific format files (tablatures) and their rendered text-like tokens, enables us to perform symbolic guitar music style transfer leveraging expressive guitar playing information, and to produce playable guitar tablatures. We first adopt K-hot encoding to transform the task from sequence generation to binary classification of multiple variables, and use top-k sampling to reproduce sequences from output K-hot vectors. We then propose a novel model we call CycleGMT, a CycleGAN-based model for symbolic guitar music style transfer. Finally, to mitigate the severe sparsity in the data and its resulting content loss, we adopt a skip connection between the input and output of the model, successfully achieving style-transferred music whose quality being competitive with human-composed remixes, while the musical complexity of the style-transferred music can be controlled by adjusting the value of k in top-k sampling.

"DON'T ONLY PRACTISE YOUR ART, BUT FORCE YOUR WAY INTO ITS SECRETS." – LUDWIG VAN BEETHOVEN

Preface

As a music enthusiast and computer science student, I vividly recall a night during my undergraduate studies when I found myself distressed, unable to compose a satisfying melody for my new song due to my insufficient knowledge in music theory. It was then that a thought came to me: Why couldn't I employ artificial intelligence to assist me, or even take over this task entirely? From that day on, I set the pursuit of my career at the intersection of music and artificial intelligence as my objective. Music style transfer had always been a composition task I desired to undertake yet felt incapacitated to accomplish alone. But today, with the aid of artificial intelligence knowledge, I managed to accomplish it. This thesis has also facilitated my initial step into the field of music technology, paving the way for my formal research in this sector at IP Paris. All of this still seems incredible and exhilarating to me.

I would like to express my great thanks to Cynthia Liem and Jorge Martinez Castaneda, my supervisors who granted me the full freedom to decide on the thesis topic, allowing me to engage in what I love and ultimately embark on my path of researching music technology. My special thanks goes to Jorge, for his continuous support and kindness throughout the past 8 months. Although there were ups and downs in my work, you are the one who made this journey bright and pleasant. I would also like to thank Cynthia, for the discussions we had for hours and all your guidance despite your busy schedule. As a musician and computer scientist, you have provided me with numerous professional support, and also serve as a role model whom I aspire to emulate.

Finally, I want to thank my parents, my friends and my girlfriend, who's been supporting me all along this road of studying abroad. You are always my biggest motivation to move forward and see things that I've never seen. I feel truly lucky to have you all by my side.

Xuanyu Zhuang

Delft, The Netherlands
12th September 2023

Contents

Chapter 1

Introduction

The concept of style transfer can be explained as follows: For a sample with Style A, assume its information is composed of two parts, one representing content and the other representing style. With the objective of preserving the content information as much as possible, the style information is transformed from domain A to domain B, resulting in a new sample with the same content but of Style B.

When it comes to the field of music, the style transfer task becomes more specific, that is to transfer the genre (style in terms of music) of a music piece while keeping the basic pattern of its melody and rhythm to ensure that the audiences can recognize the original music from its style-transferred version. This defines the task of music style transfer. Many studies have attempted and successfully employed deep learning models to tackle this challenge. Classic choices among these are deep generative models, such as VAE (Variational Auto-encoders) [12] and CycleGAN (Cycle-Consistent Generative Adversarial Networks) [75], because they can effectively learn the underlying style feature distribution of samples.

For music style transfer and similar generative tasks based on music, the potential musical representations typically include symbols [45], audio waves [33], and spectrograms [51]. Among all these three forms of representations, symbolic representation, where notes, their duration, velocities, and other attributes are explicitly encoded, offers a higher-level view of music. Symbolic music style transfer thus manipulates those structured properties of music, such as the patterns, rhythms, chord progressions, and other features. Moreover, since it's not working directly with waveforms but with musical notations, and sounds are produced from those notations using synthesizing tools, it allows for intricate manipulations without altering the sound quality.

In this thesis, We propose a novel model we call CycleGMT (CycleGAN-based model for symbolic guitar music style transfer) with the inspiration of G.Brunner et al. [8] and the contribution of DadaGP dataset [60]. Being the first guitar music dataset consists of guitar tablatures and fully leverages expressive guitar playing information by a unique encoding-decoding protocol, DadaGP provides a solution for symbolic representation of guitar music. Based on this, we develop a special use case of K-hot encoding to encode the text-like data that DadaGP provides, resulting in high quality and controllable musical complexity of generated style-transferred music. We also propose some measures to mitigate the

negative effect of the sparsity in the K-hot encoded data, including a skip connection outside of the CycleGMT model. As a result, the negative impact of sparsity on preserving the original content was largely mitigated. However, the issue of model outputs becoming homogeneous due to sparsity remains unresolved. It would cause the style-transferred music to have a certain sense of repetition over a long time dimension, but it does not have a significant impact on the overall quality of the style-transferred music.

The remainder of this chapter explains the differences between guitar tablature and stave notation 1.1, and based on this, introduces the research question of this thesis 1.2. The final section outlines the structure of the thesis, accompanied by a schematic diagram to facilitate the reader's understanding 1.3.

1.1 Guitar Tablatures and Stave Notation

Considering that symbolic music style transfer is based on musical notation systems, the choice of which notation system to use for style transfer becomes paramount. In Classical music, the stave notation [69], using 5 horizontal lines and the 4 spaces between them to represent pitches, has been widely used due to its outstanding musical representation capabilities and universality across a large number of Classical instruments. It communicates the pitch, rhythm, dynamics and other facets of a composition, making it a versatile language of music. For this reason, those partition of tasks of music technology who focuses on symbolic representation of music are mainly looking at stave notations and MIDI (Musical Instrument Digital Interface) piano rolls [45].

Nevertheless, when it comes to guitar music, another form of notation which is more intuitive for guitar playing was employed, named guitar tablature. As shown in Figure 1.1, guitar tablature contains basic musical information including pitch and rhythm as well, but all of them were represented based on the desired finger positions on the six strings of guitar rather than directly exhibiting which note to play. The numbers indicate the the exact fret of a string where the musician should place his or her fingers on to produce a particular note or sound. In addition to this, a salient advantage of guitar tablature lies in their ability to provide specific guitar-playing techniques. Such information contributes to the production of more sonorous and pleasing music, characterized by more nuanced stylistic traits.

1.2 Motivation and Research Question

As an enthusiast of the guitar, I have been continually paying attention to works related to guitar music within the Music Information Retrieval (MIR) field. However, such endeavors are not frequently encountered in the literature. I argue that the primary reason for it is that much of the preceding work in MIR domain is centered around Musical Instrument Digital Interface (MIDI) files. A MIDI file [7] is a common format of storing music with a digital symbolic representation that contains sequences of messages or commands representing musical events, where a musical event may contain note and pitches, duration, velocity, etc. While the symbolic representation provided by MIDI can be equivalently converted to a stave notation, this conversion does not support guitar tabla-

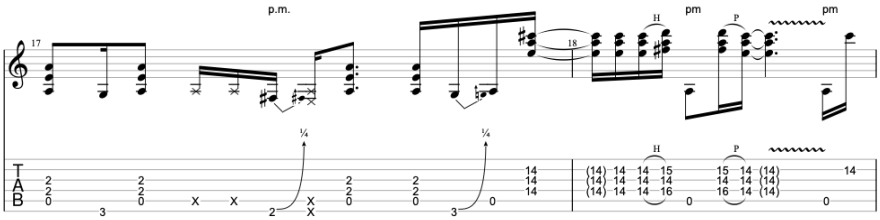

Figure 1.1: **Comparison between stave notation and guitar tablature with the example of *We Will Rock You* by Queen [55]. The top stave notation and bottom guitar tablature are equivalent pairs. The stave notation directly shows which note to play, while the guitar tablature instructs the finger position on a guitar fretboard rather than exhibiting the note itself. The letter *p.m.* (stands for Palm Mute), *let ring* and the bending arrow represents different guitar playing techniques, which are provided by guitar tablature only and are not supported in stave notation.**

tures and excludes the expressive guitar playing information specific to guitar tablatures.

However, the advent of DadaGP [60] offers a solution to this issue. DadaGP, a novel guitar music dataset released by Pedro et al. in 2021, consists of over 26,000 guitar songs, all notated in GuitarPro format [67] as digital guitar tablatures. Furthermore, they introduced an encoding-decoding scheme accompanying the dataset that achieves the conversion of guitar tablatures into musical events represented by text-like tokens [32], perfectly integrating the expressive information exclusive to guitar tablatures. This initiative fills the previous gap in symbolic representation of guitar music caused by the inability of MIDI files to encode guitar tablatures, thereby paving the way for potential explorations on symbolic guitar music information retrieval tasks.

The research question for this thesis then emerges:

> ‘ *How to achieve symbolic guitar music style transfer based on guitar tablatures, in order to leverage the expressive guitar playing information and produce playable guitar tablatures?* ’

This general research question can be further divided into 3 sub-questions that are more specific:

Research Question 1 *How to utilize the text-like tokens provided by DadaGP? And correspondingly, what kind of modeling approach should be applied?*

Research Question 2 *How to choose and design an appropriate model structure to achieve symbolic music style transfer?*

Research Question 3 *How to quantitatively evaluate the performance of music style transfer?*

1.3 Thesis Structure

The Thesis mainly consists of four parts:

- A literature review of music style transfer and recent works related to guitar music is provided in Chapter 2. Section 2.1 collects works based on the different musical representations used, which are musical symbols, audio waves and spectrograms. Section 2.2 explores supervised and unsupervised approaches. The last Section 2.3 presents review on recent works that focus on guitar music modeling.

- An introduction of DadaGP dataset [60] and our solution to **Research Question 1** by K-hot encoding with probability modeling is presented in Chapter 3. DadaGP provides the foundational possibility for achieving symbolic guitar music style transfer. In Section 3.1, we explain the data structure of DadaGP and further explore the data distribution within DadaGP through statistical summarization. In Section 3.2, we explore the methodology of encoding the DadaGP data samples, as they are provided in the format of text-like tokens. We firstly present the initial attempt on integer encoding and sequential modeling in 3.2.1 and then the K-hot encoding and probability modeling that perfectly solves **Research Question 1** in 3.2.2. Additionally, guitar roll, an extended visualisation form that is very similar to a piano roll, is also presented in 3.2.3.

- The proposed model, CycleGMT (CycleGAN-based model for Symbolic Guitar Music Style Transfer), is presented in Chapter 4 to resolve **Research Question 2**. CycleGMT is built upon a Cycle-consistent Generative Adversarial Network (CycleGAN) [75]. Section 4.1 explains the structure and working mechanism of a CycleGAN, and in Section 4.2 we address the instability in our CycleGAN training due to an over-powered discriminator by modifying the structure to a Wasserstein GAN (WGAN) [2]. In Section 4.3, we discuss how the severe sparsity in the training data affected our result and propose several mitigation measures.

- The evaluation experiments conducted to test the generative quality of CycleGMT and analysis results are provided in Chapter 5, solving **Research Question 3**. By combining an objective evaluation introduced in Section 5.2.1 that uses a genre classifier, and a subjective test presented in Section 5.2.2 by paper survey, we found that the quality of style transfer produced by CycleGMT was compatible with those samples composed by humans artists. Moreover, when we blend it with other two human-made style transfers, participants found it challenging to discern which one was generated by CycleGMT, indicating a high musical quality of the model output.

To facilitate reading, we further illustrate the thesis structure in Figure 1.2. The figure exhibits the chapters and sections of the thesis in sequential manner, as well as their interconnections with each sub research question. The sections where novelty lies in are marked with yellow. The dependencies between each section are represented with arrows.

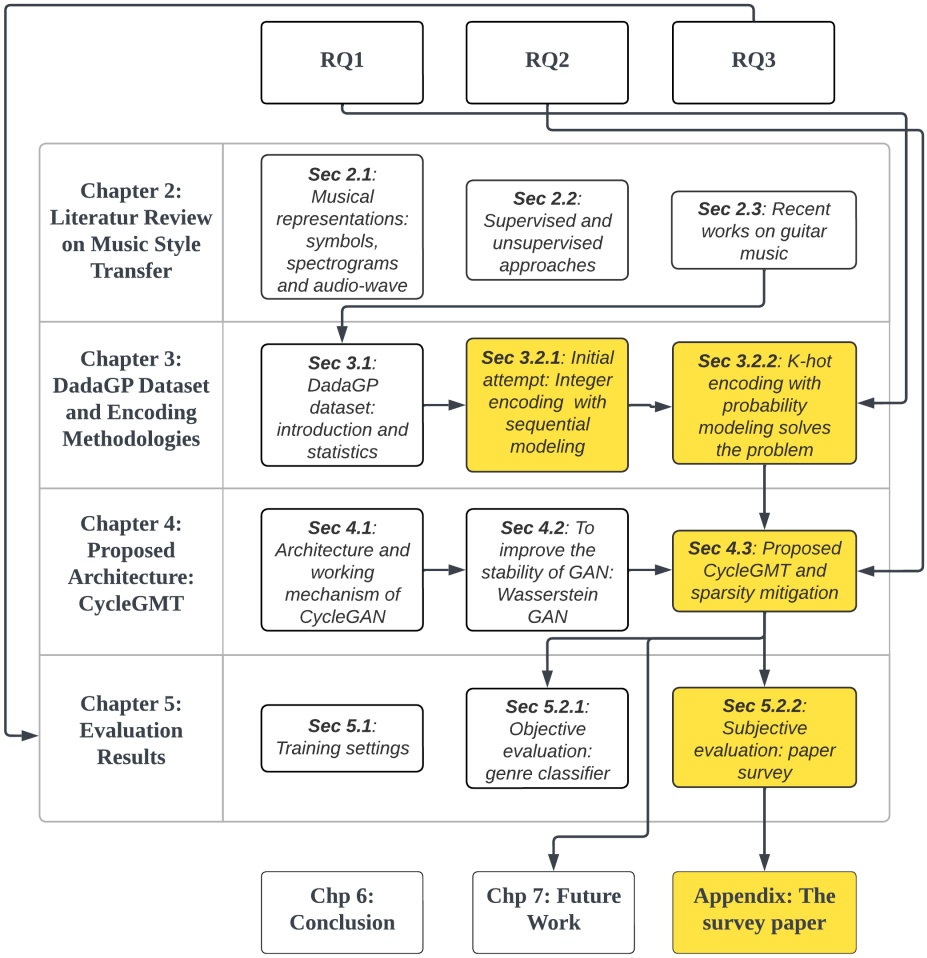

Figure 1.2: **Structure of this thesis. The yellow color marks the sections where novelty lies in, and black arrows represent the dependencies between each section. Note that some of the sections are not included in this figure.**

Chapter 2

From Image Style Transfer to Music Style Transfer

The concept of style transfer was initially originated from computer vision field, referring to the task of rendering the content of one image into the style of another. Gatys et al. [20] first introduced the neural style transfer approach, in which Convolutional Neural Networks (CNNs) [50] are adopted and trained to separate and recombine content and style information. The famous cycle-consistent adversarial network (CycleGAN) structure, proposed by Zhu et al. [75] is another remarkable work designed for unpaired image-to-image translation. It contains two sets of generators and discriminators to transfer samples between domain A and domain B while preserving content features. This is ensured by cycle loss, which is designed to guarantee a successful transfer-back. As a modified version, Wasserstein generative adversarial network (WGAN) was further proposed [2]. Gradient penalty [24] and Wasserstein distance [58] were brought into the original CycleGAN structure to stabilise its training procedure.

The success of style transfer implementations in image-related fields rapidly brought this concept to other domains, such as music [15]. Style transfer can be explained in this case, as preserving the content information of a musical piece (melody, rhythm, etc) while changing its genre.

2.1 Regarding Musical Representations: Symbols, Spectrograms and Audio Waves

Due to the diverse forms of musical representation, a number of style transfer methods have been developed regarding these different forms of expression.

2.1.1 Symbolic Representation

As shortly mentioned in section 1.2, symbolic representation stores musical information based on events which have explicit musical meaning, such as pitches, duration, velocity etc. Typical digital symbolic representation includes MusicXML [21] and MIDI (Musical Instrument Digital Interface) piano roll representation. A piano roll was initially invented as a music storage medium used

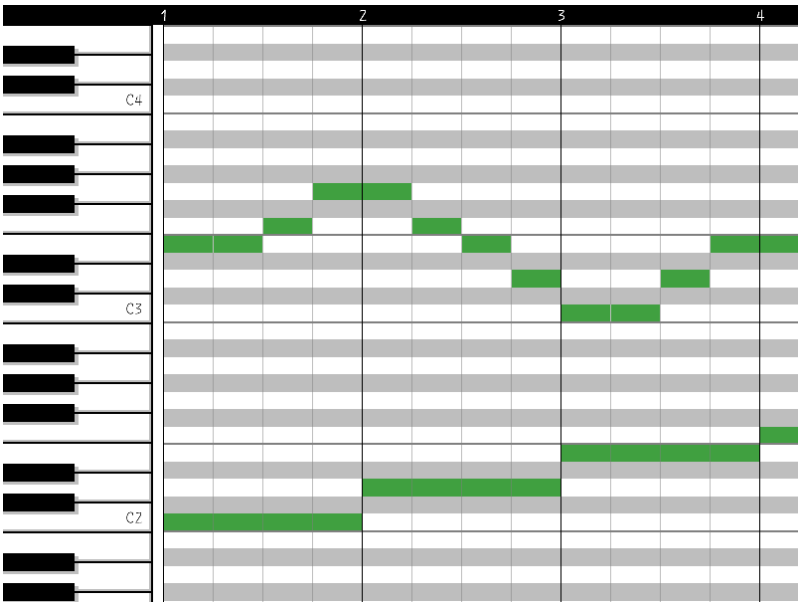

Figure 2.1: **Figure from [68], exhibiting a MIDI piano roll where the Y-axis are pitches represented by piano keys and X-axis represents time stamps.**

to operate a player piano [68]. Specifically, it is a paper with holes in it, where each hole encodes note parameters like pitch and duration. It provides a more intuitive understanding over traditional sheet music, and is further adopted in MIDI files as a standard protocol for controlling and synchronizing digital instruments [48]. As Figure 2.1 illustrates, a MIDI piano roll displays time stamps horizontally and pitches vertically and uses rectangles to stand for consecutive notes.

Huang et al. (2017) made a significant contribution to the domain of symbolic music generation by proposing the MuseGAN algorithm [17] , with their model capable of generating music with multiple tracks, demonstrating the effectiveness of Generative Adversarial Network (GAN) for style transfer in symbolic music. G.Brunner et al. in [8] introduced the idea of symbolic music style transfer with a CycleGAN. The music samples were stored in MIDI files and then rendered into piano rolls, which gives them an image like quality and therefore allows to apply CycleGAN, which was an image-style-transfer technology. Another work by Wu et al. embedding Transformer-based architecture into a Variational Auto-encoder (VAE) [70] was proposed to exploit advantages of both model to achieve an improved performance of music style transfer.

2.1.2 Spectrograms and Audio Representation

MelGAN-VC [51] proposed another solution based on the Mel-spectrograms of music, in which the idea similar to an image style transfer was also adopted. Lu et al. adopted a multi-modal structure also upon Mel-spectrogram based representations and achieved timbre-enhanced music style transfer [41]. Hung et al. [33] introduced an audio-based method, which focuses on disentangling

latent factors such as pitch and timbre directly from audio waves of arbitrary music source and thus learning its composing features. These features are then recombined to produce different composing arrangement, thus achieving music style transfer. The work of McAllister et al. [42] explored music style transfer with another interesting representation of music, which is the Constant-Q transformed [6] spectrograms.

2.2 Supervised and Unsupervised Approaches

2.2.1 Unsupervised Approaches

Another technical point of distinction is whether to use supervised or unsupervised learning. Most of the works in this field, including ones mentioned above and [46] were using unsupervised method because of the difficulty of acquiring paralleled data for training. In this context, generative models that can be trained with unsupervised methods, like CycleGAN [8] and VAE [7] [70], becomes popular. Such models are capable of learning the latent feature distributions within data samples of different style, thereby defining distinct styles and accomplishing style transfer based on this learned understanding.

2.2.2 Supervised Approaches

The work of [14], [13] and [12] instead focuses on one-shot music style transfer using self-supervised learning. A synthetic method [14] [12] was used to mitigate the problem of lacking paired training data by using synthetic tools to synthesize different style versions of the original song, and then apply supervised training. This could indeed be a solution but the power of the system is therefore limited by the performance of chosen synthetic tools.

2.3 Recent Works on Guitar Music

Due to the superior music representation capacity of the stave notation and its universality across a variety of classical instruments, most work in music style transfer centers around stave notation and MIDI symbolic representations. In 2021, Sarmento et al. introduced a dataset for guitar music incorporating guitar tablatures with more than 26K songs [60]. Along with it, inspired by [32], they also provided an encode-decode algorithm to render the digital guitar tablatures into text-like tokens representing musical events, successfully achieved symbolic representation for guitar music with tablatures. This novelty lies the foundation of implementing guitar music information retrieval tasks using symbolic guitar music representation. Another one of their recent works [61] has already explored the feasibility of guitar music generation with DadaGP dataset [60]. They adopted a Transformer-XL structure [16] and implemented sequence generation to generate musical tokens one by one according to the previous token. Additionally, they made successful attempts on controlling the instrumentation and genre of the generated music by adopting control tokens at the beginning of each song in the training corpus for pre-conditioning.

In addition to the guitar music generation that Sarmento et al. have been researching, there are many previously vacant guitar music information retrieval tasks that can now be accomplished through the symbolic guitar music representation provided by DadaGP, such as the guitar-bass transcription [56], and the guitar music style transfer that we are going to investigate in this thesis.

Chapter 3

Guitar-tablature-based Music Dataset and Encoding Methodologies

This chapter firstly provides insights of the DadaGP dataset [60], which is a guitar music dataset based on guitar tablatures in the format of GuitarPro [67] files. The existence of DadaGP is the foundation for achieving symbolic guitar music style transfer, as previous works mainly focused on Musical Instrument Digital Interface (MIDI) files [7], in which guitar music symbols are not supported. Meanwhile, the expressive tokens that DadaGP contains (guitar playing techniques, rhythm arrangements etc.) make it possible to use them to enhance the performance of guitar music style transfer.

The second section of this chapter addresses the **Research Question 2**. Two encoding and modeling approaches, integer encoding with sequence modeling and K-hot encoding with probability modeling, are introduced and qualitatively evaluated. While the latter one turns to perform significantly better than the former by ensuring that the produced style-transferred music has rational music structure and its musical complexity can be controlled by top-k sampling method [38], it also can be extended to a form of visualization named guitar roll, which is very similar to a piano roll mentioned in Section 2.1.1.

3.1 DadaGP Dataset: A Guitar Music Dataset based on Guitar Tablatures

3.1.1 Introduction to DadaGP

DadaGP [60] is a dataset consisting of tokenized GuitarPro songs. GuitarPro [67] is a type of guitar music editing software, and it also serves as a popular electronic format for guitar tablatures, which can also be manipulated and analyzed using a Python package: PyGuitarPro [59]. DadaGP collects 26,181 guitar songs from more than 700 musical genres, with all of them represented with guitar tablatures in GuitarPro format. Same as other well-known musical formats like MIDI, GuitarPro contains accurate pitch and rhythm information

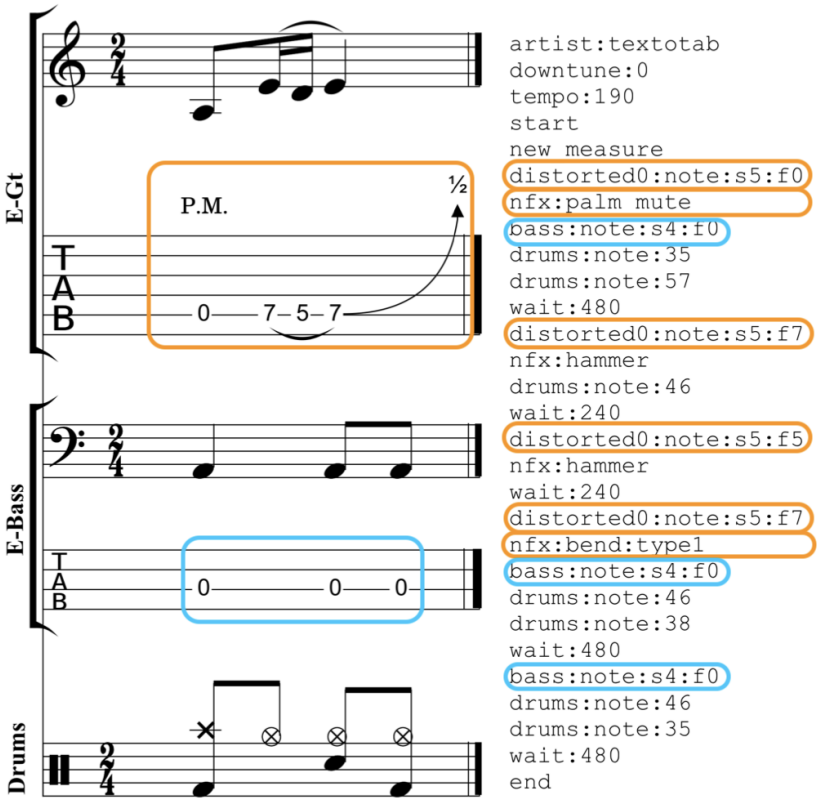

Figure 3.1: **A figure from [60], displaying a measure with a distorted guitar, bass and drums in GuitarPro's graphical user interface (left), and its conversion into token format (right). As an example, orange rectangles mark the electrical distorted guitar (E-Gt) track on the left and its corresponding text-like tokens on the right, and blue rectangles mark those for the electrical bass (E-Bass).**

and works with multi-instrument tracks as well. In each line of guitar tablature, it provides an equivalent stave notation reference to annotate the notes. Inspired by event-based MIDI encoding protocol [32], the authors also developed an encoding/decoding algorithm to achieve conversion between GuitarPro format and text-like tokens, whose protocol can be explained by Figure 3.1 presented in paper [60].

3.1.2 The Data Structure of DadaGP

All the tokens in DadaGP dataset can be roughly classified into 3 categories:

- **Note tokens**. This refer to those tokens containing a 'note', for instance, $distorted0 : note : s5 : f0$. The structure of these kinds of notes follows $Instrument : note : String : Fret$ where the string and fret instructs which position on the guitar. Each of these tokens represents a single note with the selected pitch played on a specific instrument, and multiple note

tokens can be played simultaneously to produce one complex, harmonized sound.

- **Positional tokens**: tokens that appear only at the start or at the end of a song ($artist:, downtune:, temp:...$), or tokens indicating the separation of musical bars or harmonized sounds (*new_measure, wait:*). These tokens are necessary components for the decoding mechanism of DadaGP. Additionally, *wait:* tokens are paid extra attention to as they perform as the positional indicators of actual sounds. Each *wait:* token represents a certain time interval between two harmonized sounds, where a harmonized sound consists of one or more **note tokens**.

- **Expressive tokens**. The tokens containing information related to guitar playing techniques and other expressive notes e.g. *nfx : let_ring, nfx : vibrato*. According to the encoding-decoding protocol of DadaGP, the presence of these tokens will affect all the notes played by arbitrary instrument at that moment. This means if a set of tokens between two *wait:* tokens contains an expressive note, then all the tokens in this set will share that effect.

Traditional Music Information Retrieval (MIR) tasks [9] often starts from the audio itself or graphical notations such as piano rolls. However, this text-like encoding allows music data to be treated as discretized and sequential data, analogous to tasks in natural language processing. Therefore, many models that have been proven successful in discrete data domain can be applied. Simultaneously, this text encoding fully retains the specific information on guitar playing techniques which are not supported by conventional representations, making it a learnable feature which could potentially enhance the performance of the desired guitar music style transfer.

3.1.3 The Selection of Genres based on Statistics of DadaGP

For a music style transfer task, the selection of the two genres to be transferred will have a pivotal impact on the performance of the transformation. To determine such choices, we conducted the following statistical analysis on the dataset:

Amount of songs in each genre. A genre refers to the category of music, for example, as per typical western colloquialism, Rock and Classical. Every song in DadaGP is labeled with one or more genre labels. Firstly, the two selected genres must contain sufficient data to support training, thereby ensuring the performance of the model. According to the genre distribution of the dataset shown in Figure 3.2, we initially decide the subset of candidate genres including Rock, Metal, Jazz, Blues, Classical and Folk, as each of them contains at least hundreds of songs.

Proportion of note tokens in each genre. When a genre has a higher proportion of **note tokens**, or to say a higher content proportion, it generally means that it is more focused on the melody itself and has a lower musical complexity in terms of musical effects arrangement and guitar playing techniques. On the contrary, those genres with lower content proportion are likely to be more dependent on expressive components to create a specific musical atmosphere.

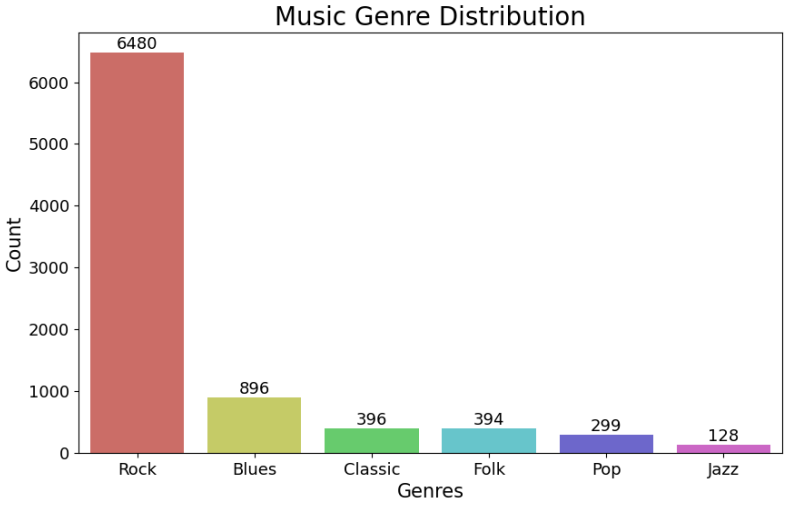

Figure 3.2: **The distribution over the number of songs of a subset of candidate genres in DadaGP dataset (songs with genre label *blues_rock* are considered as Blues).**

In order to make the effect of the music style transformation more pronounced, we should select two genres with as large a difference in content proportion as possible to create a contrasting effect.

Proportion of playing-techniques-related tokens in each genre. This statistic is similar to the former one, while it targets on those **expressive tokens** representing guitar playing techniques which always start with a *nfx:* or *bfx:*. By collecting such tokens and calculate its frequency of appearance across the genre, the assumption of how much a genre relies on expressive guitar playing information like playing techniques could be made. Similarly, for this statistical indicator, the two genres we choose should ideally have as large a difference as possible, thereby allowing the model to more easily capture those differences and learn how to achieve style transfer effectively.

	Content Proportion	Nfx **Proportion**
Rock	0.824	0.090
Jazz	0.829	0.067
Classic	**0.872**	**0.062**
Popular	0.846	0.087
Folk	0.814	**0.115**
Blues	**0.790**	0.096

Table 3.1: **Statistics of DadaGP dataset.** Nfx **Proportion stands for the proportion of *nfx:* or *bfx:* tokens that are related to expressive guitar playing techniques.**

Table 3.1 exhibits the statistics mentioned above. Many Classical music pieces

in the dataset come from Classical instruments such as the piano, so they typically do not contain information specific to guitar performance, and Classical music generally focuses more on melody, hence it has the highest content proportion and the lowest nfx proportion. Conversely, Blues and Folk music are often played on the guitar, thus containing more guitar playing technique information and effects. They have higher arrangement complexity, and thus possessing a lower content proportion and a higher nfx proportion. Considering that there is more Blues music in terms of data volume, and the guitar playing techniques used in Blues music such as bending or sliding are more prominent in auditory perception, we eventually chose Blues as the target genre for music style transfer and Classical music, which has a lower complexity, as the original genre.

3.2 To Encode Text-like Tokens: Integer Encoding or K-hot Encoding?

To utilize the text-like tokens that DadaGP dataset provides, we need to first encode them to a format conducive to processing by deep learning models. Moreover, the choice of the encoding protocol is likely to have a significant impact on the overall performance of the model. In this context, we will introduce, analyse and compare two approaches: integer encoding and K-hot encoding.

3.2.1 The Initial Attempt with Integer Encoding and Sequential Modeling

Encoding the text-like tokens with integers is a common way of tokenization in Natural Language Processing (NLP) tasks [65], which simply uses integers to represent different words. By using integer encoding in our case, musical events are regarded as words and consecutive bars can be regarded as sentences [5]. After inspecting the vocabularies with respect to each genre (shown in Table 3.2), a dictionary with vocabulary size of 1512 is built by collecting all tokens that appeared in song samples belonging to genre 'Classical' and genre 'Blues'. Tokens from all other genres are not included in this dictionary as these genres may contain unique instruments or arrangements not found in Classical or Blues songs. Including these could potentially introduce interference. Furthermore, if these tokens appear in the output, they are likely to affect the genre identification of the output results.

	Jazz	**Folk**	**Pop**	**Blues**	**Rock**	**Classical**
Vocab size	726	1009	1223	1446	1819	1235

Table 3.2: **Vocabulary size of each genre**

By working with additional embedding processes, integer encoding is suitable for both Sequence-to-sequence (Seq2seq) models [53] and Convolutional Neural Networks (CNNs) [50]. The task itself can now interpreted very similar to a text style transfer task [34]. However, certain drawbacks are introduced with this approach regarding the specific features of DadaGP dataset, these are:

1. **Relatively poor quality of generated music due to improper arrangement of *wait:* tokens in the output sequence.** With integer encoding, all tokens are collected and converted into integers according to the established dictionary, including the *wait:* tokens mentioned in Section 3.1. These tokens have significant impact on the length of output music samples and decides which tokens will be assembled together to form a harmonized sound on a beat. For instance, we assume that an input integer-encoded sequence has length 128 and includes 9 *wait:* tokens which separates the music represented by this sequence to 10 harmonized sounds with a certain rhythm pattern. If the model failed learning to properly reproduce those *wait:* tokens, for example, only generating 5 *wait:* tokens in the output sequence, the resulting music will perceptually have the auditory length of only five harmonized sounds. Additionally, each harmonized sound would contain about twice the musical events than the original input music, causing it to sound significantly different and may even come across as strange and grating to the listener.

   In other words, the generated style-transferred music samples will sound totally different from the original song even if they share the same melody notes if *wait:* tokens are located in bad positions or assigned unappropriated values. During experiments with models leaning on integer encoding, it was found that they essentially fail to learn how to properly utilize the *wait:* tokens in the output samples, resulting in music outputs of subpar quality that were not capable of bearing similar characteristics and lengths to the input music.

2. **The Dilemma of Determining Output Target Length**. The first intuition would be to set the target length of the output same as the length of the input sequence to ensure that the input and output shares a similar content, also providing more convenience for loss calculation. However, this is not applicable in our context. In the task of music style transfer, we must account for the complexity differences between the original and target musical genre. Within our musical samples, compared to Classical music samples which are predominantly characterized by melodies, the expression in Blues music samples heavily relies not only on melodies based on Blues scales but also on a myriad of accompaniments and expressive information to evoke a particular ambiance. Consequently, Blues samples exhibit greater complexity, demanding more tokens for representation. This leads to Blues samples having a larger average of number of tokens in harmonized sounds separated by *wait:* tokens comparing to Classical music samples, as evidenced by Table 5.2. Hence, determining the target length for the output sequence presents a dilemma: If it's set equal to the input sample length, the complexity difference between input and output cannot be properly maintained, which may harm the performance of style transfer. However, if the target length for the output sequence is set slightly longer than the input, the natural question arises: how much longer is appropriate? Moreover, when the lengths of input and output differ, how should the loss function be defined and computed? This is a rather vague problem and finding a satisfactory resolution proves challenging, thus we decided not to pursue further exploration on it.

3. **The Gradient cut-off due to non-differentiable sampling operation when using cyclic structure like a CycleGAN.** By taking integer encoded sequences as input, both Seq2seq model and CNNs tend to output Softmax [19] activated probability distributions over all categories of each token in the sequence, usually with shape (*sequence length, category length*).

   But when using cyclic structure like a Cycle-Consistent Generative Adversarial Network (CycleGAN), the output of one sub-model (generator) should be able to be reused as the input of another sub-model (another generator) with the same model structure, which emphasises the consistency between the data type and shape of input and output. But, as mentioned above, for Seq2seq or CNN models, the output would be probability distributions when using integer sequences as input, failing to address such consistency.

   To tackle this, a sampling process [10] could be adopted to reproduce integer sequences from the probability outputs of the sub-models, and reuse them as the input of another sub-model. However, this approach would introduce extra sampling uncertainty into a CycleGAN, and would also cause a more serious problem: Gradient cut-off. The sampling approaches that produce discrete sequence from a probability distribution are all non-differentiable operations, which means the back propagation calculated on the corresponding loss was cut off and were not able to flow back to adjust the model parameters. In this case, the gradients are thus not applicable for losses calculated on sampled sequences and the generators can not be trained properly. A mitigation solution for this issue could be sequence generative adversarial nets with policy gradient (SeqGAN) [73], in which GANs are updated in a reinforcement learning (RL) manner by calculating policy gradients to avoid gradient cut-off. While it almost addresses the problem, after a set of experiments we found that the calculated policy gradients are not effective enough in terms of preserving the input content and we believe that the performance of the model is unavoidably compromised, which still can not make integer encoding an outstanding choice for pre-processing methodology.

3.2.2 K-hot Encoding with Probability Modeling Solves the Problem

In order to better utilize the dataset and take into account the auditory experience and musical structure of the output music, K-hot encoding is employed in this context. The primary function of K-hot encoding is to transform the sequence generation task, which uses integer encoding, into a task of binary classification on multiple continuous variables. This is done while ensuring that the output music has the same duration and basic musical structure as the original input music, ensuring the input and output music are at a comparable level.

Technically, the K-hot encoding protocol exploits the characteristic of the *wait:* tokens to segment a music piece into several sequential harmonized sounds, as mentioned in Section 3.1.2. All the tokens in the entire dataset are split with *wait:* tokens, and the tokens between each two *wait:* tokens are collec-

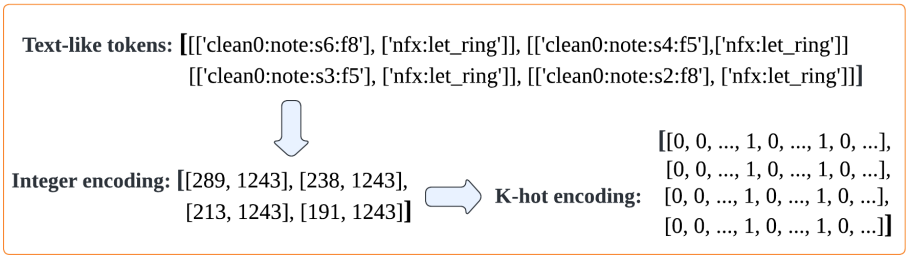

Figure 3.3: **Illustration of the K-hot encoding protocol. First, data are separated into harmonized sounds by *wait:* tokens (which are then removed from the data). In this example, the text-like tokens consists of four harmonized sounds, and each of them has two token components: one clean guitar note token and one nfx expressive token. Next step is to encode the text-like tokens to integers according to the established dictionary and then apply K-hot encoding, i.e. the vector $[289, 1243]$ will result in the $289th$ and $1243rd$ element in the K-hot vector to be 1 and others to be 0.**

ted and represented by a single k-hot vector. In other words, we extract the positional tokens from the data to leave pure content data for training, and re-insert the extracted positional tokens into the model output to reproduce music with reasonable structure. A more detailed explanation is provided in Figure 3.3. Each k-hot vector therefore represents one harmonized sound which is produced by playing the multiple note tokens and performing the specific expressive tokens that are marked with 1 in the K-hot vector together at the same time. By sequentially assembling a fixed amount of such harmonized sounds, a music piece is formed and can be represented by a K-hot matrix with shape (*sequence length, vocabulary size*), which is suitable as an input to models including Seq2seq model and CNNs. We choose to gather 24 harmonized sounds together as a single training music sample. To build the dictionary, we collect all the tokens that appeared in Blues and Classical songs and resulted in a vocabulary size of 1512. The reason for not including tokens used in songs of other genres into the dictionary is to eliminate interference and to compress the depth of the dictionary as much as possible, so as to reduce the sparsity of the K-hot matrix. Hence, we finally have K-hot matrices with shape $(24, 1512)$ for training.

Comparing to integer encoding, K-hot encoding has the following significant advantages:

1. **Outputs music with more rational structure and better quality.** While training or inference stage, the input to models are K-hot matrix in which *wait:* tokens are not included. By inserting a set of *wait:* tokens assigned with the same duration value between each harmonized sound, the original input music and the produced output music are guaranteed to share the same musical duration and fundamental rhythmic relationship between measures. This means that, regardless of whether the model can retain detailed information such as the melody or beat of the original song, it ensures the input and output at least share similar musical dimensions

and have comparability in a musical sense.

2. **Can be seen as a continuous binary classification problem that assign 0 or 1 to multiple variables simultaneously.** The task of the generator can now be reduced from sequence-to-sequence generation to a multi-class classification problem. Each element in the K-hot vector represents the binary class of a token component, where 1 indicates its appearance in the harmonized sound and 0 means absence. The total number of token components is equal to the amount of tokens in the dictionary and they are not mutually exclusive, which means there could be several classes labeled with 1 at the same time. The output of the generator should therefore be a probability vector of the same shape, where each element is activated by a Sigmoid [26] function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

   A Sigmoid function maps its input to an output of a certain range, typically 0 to 1. Each value thus represents the probability of the corresponding token class being 1, or to say the likelihood of its occurrence within the harmonized sound. This also avoids any gradient-related problem from happening, as now all the data included in the training loop are regarded continuous and therefore no non-differentiable sampling happens in the gradient flow.

3. **Controllability over the complexity of each bar of the output music thanks to top-k sampling.** After acquiring outputs from the generator, sampling are done to retrieve integer tokens from probability vectors. The conventional sampling method uses a threshold to determine whether a token class value is 1 or 0. However, this method is not suitable for the context of this study, as there may be significant differences between the average probabilities of the produced harmonized sound vectors. For example, if 0.5 is used as the threshold, some output harmonized sounds may contain as many as dozens of token components that are assigned with probabilities larger than 0.5, while others may not contain any token components above such threshold. The number of token components in a harmonized sound directly determines the perceived complexity of it, with harmonized sounds containing more tokens likely featuring a greater variety of instruments or playing techniques. The style of a piece of music also has a strong relationship with this complexity, so uneven distribution of the number of token components in each harmonized sound can potentially lead to poor musical perception and makes it harder to correctly and stably control the style of the generated music.

   Instead, the top-k [38] approach is utilized in this study which perfectly satisfies all the demands. This methodology transforms a probability output into music by selecting the k token components with the highest occurrence probabilities in a harmonized sound vector. In the resulting music output, each harmonized sound consists of exactly k token components, and between each harmonized sound vector an arbitrary *wait* token can be inserted. This avoids the uneven distribution of number of token components across each harmonized sound, which thus ensures the consistency in

musical complexity across the entire musical piece, making the style of the output music more stable and pronounced. Meanwhile, we can adjust the complexity of the output music by varying the value of k. For instance, when we want to transform a certain type of music to folk or Classical, we might opt for a relatively small k value; for genres like Rock or Blues, usually performed by a band, often featuring a multitude of instrument tracks or guitar-playing techniques, we would prefer a larger k value to assemble more tokens in a harmonized sound to ensure the complexity of the output music. A visualized comparison between outputs under different k values are presented in Chapter 5 in Figure 5.3 and Figure 5.4.

3.2.3 Guitar Roll: Extended Visualisation of K-hot Encoding

By utilizing K-hot encoding, we also achieve a musical representation similar to a MIDI piano roll mentioned in Section 2.1.1. It shares a similar concept of visualising with our K-hot matrices except that the pitches in our case is replaced by the depth of vocabulary. An example of guitar roll is shown in Figure 3.4. By this representation, the input music and its style-transferred output can be compared in a more intuitive way.

3.3 Conclusion

In this chapter, we provided explanation of data structure and statistics of DadaGP [60], which is the first symbolic guitar music dataset with over 26K songs represented by guitar tablatures in GuitarPro files [67]. Classical music is selected as the original genre and Blues as the target genre according to their large difference on musical complexity illustrated in Table 3.1, in order to maximize the effect of style transfer.

Furthermore, after encountering challenges with integer encoding and sequential modeling, we propose a K-hot encoding solution with probability modeling, which subsequently demonstrated significant improvements and finally allowed our model to produce playable tablatures with reasonable structure, successfully addressing **Research Question 1**. We also developed guitar roll, which is an extended visualisation of K-hot encoded data and shares similar concept with a piano roll. It offers a clearer insight into the differences before and after the style transfer. In the next chapter, we will discuss how to choose and design the proper model structure for symbolic guitar music style transfer based on K-hot encoded data.

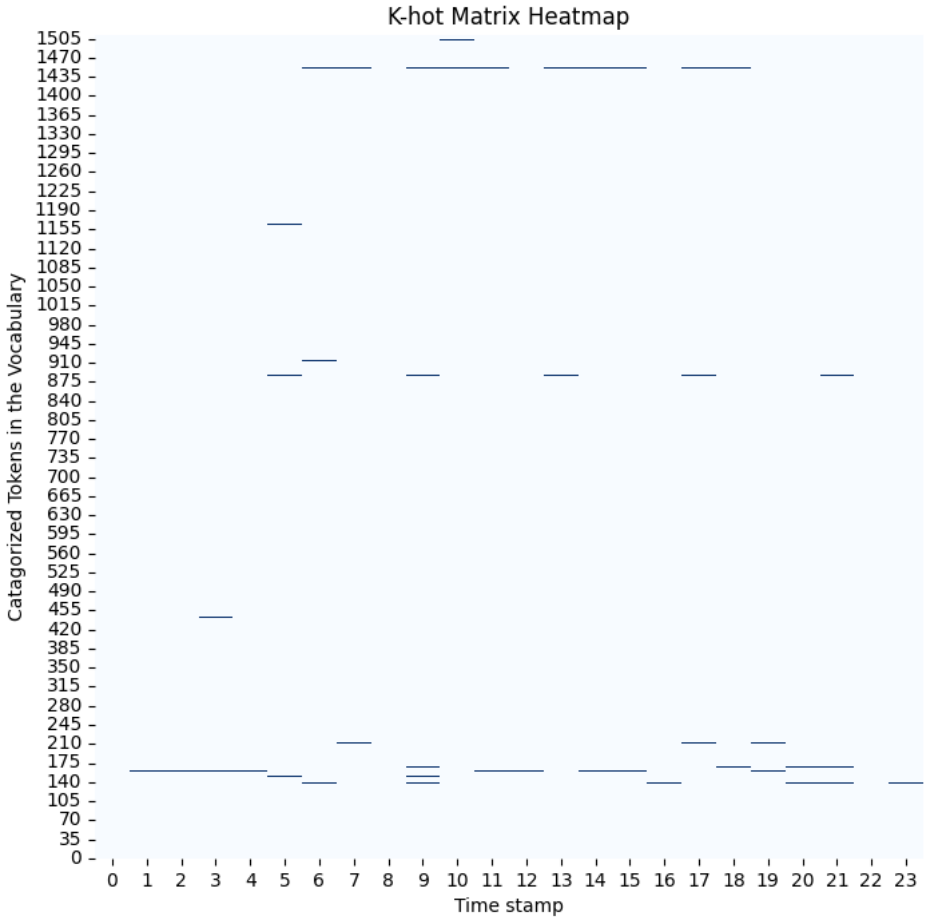

Figure 3.4: **An example of a guitar roll. X-axis represents time stamps and Y-axis represents each categorized token in the vocabulary. Each blue line marks the presence of the corresponding token in the certain period. The value consists of only 0 and 1 so the middle value in the color map can be ignored.**

Chapter 4

CycleGMT: Proposed CycleGAN-based Model for Guitar Music Style Transfer

In this chapter, the architecture and working mechanism of the proposed model, named CycleGAN-based Guitar Music Style Transfer Model (CycleGMT) is discussed, in order to solve the **Research Question 2**.

Cycle-consistent Generative Adversarial Network (CycleGAN) [75] was initially proposed and implemented in image style transfer tasks and was a great success thanks to the two-GAN structure and contribution of cycle loss. Inspired by the work of Brunner et al. [8] that extended this success of CycleGAN to symbolic music style transfer tasks, we borrow it as the backbone model and further update it to CycleGMT by choosing specific generator and discriminator structure. Furthermore, we extend it to a Wasserstein Generative Adversarial Network (WGAN) [2] which can achieve a more stable training.

A significant part of this chapter discusses the special mode collapse case in our experiments due to the sparsity of training data, as well as corresponding attempts to mitigate such sparsity including adopting weighted binary cross entropy and a skip connection outside of the model.

4.1 Cycle-consistent Generative Adversarial Network

As the backbone structure of our proposed model, Cycle-consistent Generative Adversarial Network, introduced by Zhu et al. in 2017 [75], is an extension of Generative Adversarial Network(GAN) [22] and was particularly efficient on domain adaption and style transfer tasks in image processing field [35]. The key contribution of CycleGAN is its ability to learn mappings from one domain to another with unpaired training data, leveraging a cycle consistency loss to achieve unsupervised content preservation during style transfer. This section includes an overview of the architecture and working mechanism of CycleGAN, in order to provide a clear insight of how it and our proposed model CycleGMT being able to achieve successful style transfer.

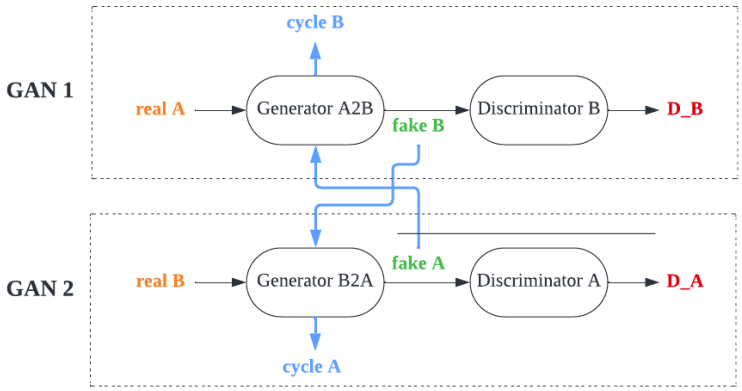

Figure 4.1: **The architecture of a CycleGAN. GAN 1 is responsible for the domain transfer from A to B and GAN 2 for the opposite direction. The real domain A samples are fed into *GeneratorA2B* to produce fake domain B samples. These fake domain B samples are then provided to *Discriminator B* to evaluate its probability of being fake. Same procedures also happen in GAN 2, except the inputs are real samples from domain B. The two GANs are connected in such a way: the generated fake domain B samples and fake domain A samples are additionally passed to the other GAN's generator(*GeneratorB2A* and *GeneratorA2B* respectively) to produce cyclic version of real domain A samples and real domain B samples.**

4.1.1 Architecture

A Generative Adversarial Network (GAN) consists of a generator and a discriminator, which are all learnable neural networks. The generator tries to produce the domain transferred version of its input samples while the discriminator learns to distinguish between the real samples and fake samples generated by the generator. During training, a two-player min-max game is played where the generator improves the quality of the generated fake output to try to fool the discriminator and the discriminator learns to better identify them. As an extension of GAN, CycleGAN consists of two GANs, with one of them responsible for the transformation of style from domain A to B and another for the opposite direction. The two GANs are connected with a cyclic manner to preserve the content of input without paralleled data as supervision. Figure 4.1 provides a more detailed explanation of how a CycleGAN is established.

4.1.2 Working Mechanism

Thanks to its architecture depicted in Figure 4.1, a CycleGAN can have multiple objectives during training. Inherited from a standard GAN, a discriminator strives to correctly classify real and fake samples. Mathematically, it should learn to assign a probability value as close to 1 as possible to a real sample and 0 otherwise. Thus, the loss of a discriminator can be defined as equation 4.1:

$$L_D = \frac{1}{2}\mathbb{E}_{x\sim p_x(x)}[(D(x)-1)^2] + \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[D(G(z))^2] \tag{4.1}$$

in which L_D stands for the discriminator loss. $\mathbb{E}_{x\sim p_x(x)}$ represents the mathematical expectation over sampled real data distribution and $\mathbb{E}_{z\sim p_z(z)}$ denotes which over the sampled noisy real data distribution. G and D stands respectively for the generator and the discriminator. $D(x)$ stands for the discriminator's output for real samples, and $D(G(z))$ denotes the discriminator's output for the fake samples generated from noisy input samples by the generator. As defined in this equation, $D(x)$ should approach 1 and $D(G(z))$ should approach 0 to achieve a minimal L_D.

The discriminator loss of a CycleGAN is thus the summation of its two discriminator components:

$$L_D = L_{D_A} + L_{D_B} \tag{4.2}$$

where L_{D_A} stands for the discriminator for samples in domain A and L_{D_B} represents the discriminator for samples in domain B.

In the meantime, a generator attempts to deceit the discriminator by generating fake samples that looks 'real', which can be illustrated by an adversarial loss, explained by equation 4.3:

$$L_A = \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[(D(G(z)) - 1)^2] \tag{4.3}$$

where L_A stands for the adversarial loss. The higher the probability prediction of the discriminator on the generated fake sample is, the smaller the adversarial loss is.

Additionally, a CycleGAN necessitates the preservation of original sample information. Thus, in alignment with the unique structure of CycleGAN which connects two GANs' output, the cycle-consistent loss is defined accordingly as equation 4.4:

$$L_C = \mathbb{E}_{y\sim p_y(y)}[\|G(F(y)) - y\|_1] + \mathbb{E}_{x\sim p_x(x)}[\|F(G(x)) - x\|_1] \tag{4.4}$$

where L_C stands for the cycle-consistent loss (cycle loss). G and F denote respectively the mapping function of $Generator_{A2B}$ and $Generator_{B2A}$, with each of them responsible for transferring the samples from domain A to domain B or vice versa. The x represents the real input samples from domain A and y represents those from domain B. With '$\|\|$' representing the L_1 norm, the cycle-consistent loss can be interpreted as the L_1 norm of difference between real samples and the cyclically generated samples for both domain A and B. For minimizing this loss, CycleGAN should be adept at preserving the original content information of input real sample after passing it through the generator. This action ensures that the cyclically generated samples, which are the outputs after sequential processing by the two generators, can retain the original content information. Consequently, the difference between these and the input real samples is minimized, leading to a reduced value of this cycle-consistent loss.

From a more concrete perspective, using images as an example,the cycle-consistency loss enforces that translating an image from one domain to another and then back to its original domain should yield a reconstructed image that is similar to the original input. This principle forms up the core concept of CycleGAN, which is to enable unsupervised learning of domain transfer without paired training data.

To further strengthen the preservation of content, identity loss could be additionally defined as equation 4.5:

$$L_I = E_{y\sim p_{data}(y)}[||G(y) - y||_1] + E_{x\sim p_{data}(x)}[||F(x) - x||_1] \tag{4.5}$$

where L_I stands for the identity loss, and same denotation is used as the definition of cycle loss in equation 4.4. This loss forces that the generators should generate a sample that is close to the input sample when the fed sample is from the target domain, e.g. the output of $Generator_{A2B}$ should produce output that is similar to input when fed samples of domain B as input.

Finally, the overall generator loss can be defined as equation4.6:

$$L_G = L_{A_{A2B}} + L_{A_{B2A}} + L_C + L_I \tag{4.6}$$

in which L_G denotes the overall generator loss, while $L_{A_{A2B}}$ and $L_{A_{B2A}}$ representing the adversarial loss of the two generators shown in equation 4.3.

4.2 To Achieve Greater Stability in GAN Training

Stability is an important part of GAN training, and is a well-know challenge for the research community. In adversarial training, the goal is not simply to minimize adversarial loss, but rather to achieve a dynamic balance between the generator and the discriminator [44]. Only under this dynamic balance can the model successfully complete the task of style transformation.

In the initial phase of our experiment, the CycleGAN model we employed faced serious training imbalance issues. Ideally, under the influence of the Sigmoid [26] function (illustrated in equation 3.1), the output range should be distributed between 0 and 1. However, the actual output vector values were all centered around 0.5, a phenomenon known as 'Mode Collapse'. In this section, we will explain the concept of mode collapse, its underlying causes, and introduce our solution using Wasserstein Generative Adversarial Network (WGAN) to achieve a stable training.

4.2.1 Consequence of Instability: Mode Collapse

Take the discriminator as an example, if the discriminator's ability to distinguish real and fake samples is too weak, then the samples generated by the generator that are not within the target domain will not be effectively penalized, and as a result, the generator cannot learn the correct style transformation. On the contrary, if the discriminator is too strong, which often manifests as the discriminator's loss rapidly decreasing to zero within a few epochs during training, then the generator not only cannot obtain gradients that can be effectively used for training from the discriminator, but more importantly, the generator will be excessively penalized during the learning phase because the generated samples are not good enough.

As a result, the generator will not achieve a comprehensive exploration of the target domain and instead tend to generate some 'safe' samples, which are not guaranteed to be realistic but will not be penalized by the discriminator. This will lead to problems of output homogenization and insufficient diversity, which is known as mode collapse [39].

In the first experiments, we encountered several situations where mode collapse happened due to the over-powered discriminator. After multiple attempts with different approaches to address this problem, we found using a WGAN

instead of a standard GAN to form up the CycleGAN is a relatively effective solution to stabilise the training.

4.2.2 Wasserstein Generative Adversarial Network: A Much More Stable Variant of GAN

Soon after released in 2017 by Martin et al. [2], WGAN drew large attention because of its superior ability of solving the mode collapse problem and instability in GAN training. The main modification lies in the usage of the "earth mover's" distance [58] or Wasserstein-1 distance as the target for the establishment of loss function, which tends to produce smoother gradients to address gradient vanishing problem in a standard GAN. According to another work of Martin et al. [1], in a traditional GAN structure, the loss function can be regarded approximately equivalent to the Jensen-Shannon(JS) divergence [43] between the real probability distribution p_r and the generated fake probability distribution p_g as long as the discriminator is trained to its maximum performance. However, this brought in the problem of gradient vanishing [30] as the JS divergence would always equal to $\log_2$ no matter how far p_g is from p_r, provided that p_g are almost not possible to have non-negligible overlap.

On the contrary, the "earth mover's" distance or Wasserstein-1 distance provides desired gradients and reflection of proximity of p_g and p_r even if they have no overlap, therefore works better than the JS divergence as a metric of distance. It is interpreted by equation 4.7:

$$W(p_g, p_r) = \inf_{\gamma \in \Pi(p_g, p_r)} \int_{\mathbb{X} \times \mathbb{X}} d(x, y) d\gamma(x, y) \tag{4.7}$$

in which p_g and p_r represents two probability distributions, and γ ranges over the set of all joint distributions $\Pi(p_g, p_r)$ on $\mathbb{X} \times \mathbb{X}$, which is the space of all possible outcomes. $d(x, y)$ stands for the Euclidean distance [66] between point x and y in space X.

The key idea of WGAN is to replace the discriminator with a critic that is trained to approximate the Wasserstein distance, rather than to classify instances as real or fake. The optimization objective of the critic is defined as equation 4.7:

$$\max \mathbb{E}_{x \sim p_r(x)}[C(x)] - \mathbb{E}_{z \sim p_z(z)}[C(G(z))] \tag{4.8}$$

in which max stands for the maximum value, C stands for the critic, and all other symbols are defined similarly as in equation 4.1. This function is to maximize the difference between its two outputs for respectively the real samples and the fake generated samples. Only when this maximum is achieved, the critic loss can be approximated as the Wasserstein distance. Thus the critic tries to minimize the opposite of this objective in the training loop, which defines the critic loss function as equation 4.9:

$$L_C = \mathbb{E}_{z \sim p_z}[C(G(z))] - \mathbb{E}_{x \sim p_{data}}[C(x)] \tag{4.9}$$

where L_C denotes the critic loss, while the same other symbols are used as the previous equation 4.8. The lower this loss is, the better approximation the critic can achieve to the Wasserstein distance. As the Wasserstein distance does not experience gradient vanishing under the optimal critic, it allows us to train

the critic to optimality during early stage of training. This typically manifests in a training loop as multiple rounds of optimization performed on the critic before optimizing the generator. After the critic has achieved a good fit for the Wasserstein distance, the generator attempts to shorten the Wasserstein distance between the p_r and p_g, i.e., to minimize the objective represented by equation 4.8.

Since the first term of this objective equation only relates to the critic and the training dataset, and is not related to the generator, the generator only needs to minimize the second term. The generator loss in a WGAN can hence be expressed by equation 4.10:

$$L_{G_W} = -\mathbb{E}_{z\sim p_z}[C(G(z))] \tag{4.10}$$

where L_{G_W} represents the generator loss in a WGAN. The other symbols are defined similarly as in equation 4.8.

Additionally, the authors proposed methods to ensure that the critic function lies within the Lipschitz constraint [25]. A function is said to be Lipschitz continuous if there exists a positive real constant K (known as the Lipschitz constant), such that for all points x and y in its domain,

$$|f(x) - f(y)| \leq K|x - y| \tag{4.11}$$

where $|x-y|$ represents the absolute value of difference between x and y. Equation 4.11 guarantees the difference between two points of the target function for not exceeding a certain constant value. In the case of training a WGAN, making the critic loss function obey this constraint means limiting the gradient from changing drastically, thus making the training of WGAN more stable. Initially the authors implements weight clipping that forces the absolute value of all the parameters of the critic falls in a certain range, e.g. [-0.01, 0.01] to indirectly achieve Lipschitz constraint. However, further experiments has proved that the critic will thus push all its parameters to the two extremes, as its objective of maximizing the difference between output for real and fake samples requires larger covariance in the distribution of parameter values.

As a modification, gradient penalty is further proposed [24].The gradient penalty is calculated by taking the gradient of the critic's output with respect to its input (which is a randomly interpolated sample between real and fake data), and then penalizing the critic based on how much this gradient deviates from 1. This helps ensure that the critic function is Lipschitz continuous. The gradient penalty is defined mathematically as the following equation 4.12:

$$GP = \lambda[(\|\nabla_{\hat{x}} C(\hat{x})\|_2 - 1)^2] \tag{4.12}$$

in which GP stands for the gradient penalty. $\|\|_2$ denotes the L_2 norm, while $\hat{x}$ denotes the interpolated samples. C represents the critic, and $\nabla_{\hat{x}}$ stands for the gradient with respect to the interpolated samples. λ is the weight hyperparameter.

The entire loss function of a WGAN is therefore well established by engaging

all the losses mentioned above, expressed in equation 4.13:

$$L_{G_W} = -\mathbb{E}_{x\sim p_x}[C(G(x))] + \mathbb{E}_{y\sim p_y}[\|G(F(y)) - y\|_1] + \mathbb{E}_{x\sim p_x}[\|F(G(x)) - x\|_1] \\ + E_{y\sim p_y}[||G(y) - y||_1] + E_{x\sim p_x}[||F(x) - x||_1]$$

$$L_{C_W} = \mathbb{E}_{z\sim p_z}[C(G(z))] - \mathbb{E}_{x\sim p_{data}}[C(x)] + \lambda E_{\hat{x}\sim p_{\hat{x}}}[(\|\nabla_{\hat{x}} C(\hat{x})\|_2 - 1)^2] \tag{4.13}$$

where L_{G_W} and L_{C_W} stands respectively for the overall generator loss and critic loss in a WGAN. The overall generator loss consists of three parts: the discrimination loss defined by equation 4.10, the cycle loss explained by equation 4.4 and the identity loss represented by equation 4.5. The overall critic loss is the combination of equation 4.9 and the gradient penalty explained in equation 4.11, where $E_{\hat{x}\sim p_{\hat{x}}}$ denotes the mathematical expectation over sampled interpolates distribution. The key contribution of WGAN is its superior stability during training, perfectly addressing the mode collapse problem encountered in our experiments.

4.3 Proposed Architecture: CycleGMT

Based on the goal of the project which is to produce playing-technique-enhanced and playable style transportation of guitar music, adopting the basic architecture of WGAN to ensure a stable training, CycleGAN for Guitar Music Style Transfer (CycleGMT) is proposed and implemented.

4.3.1 Generator Structure

Introduced in Section 3.2.2, training data are presented in a 'guitar roll' format, which are K-hot matrices consists of binary values. The length of each K-hot vector equals to the depth of the dictionary used for encoding. To handle this form of input, a Convolutional Neural Network (CNN) with a Conv-Deconv [18] structure that also integrates Residual Network (ResNet) [27], known as a Residual Conv-Deconv net [18], is a good choice [11], which is also used in our backbone model borrowed from [8]. Furthermore, we modified this base structure by replacing Conv2D layers [54] with Conv1D layers [52] and adding an Embedding layer [49] after the input layer of the model, as illustrated in Figure 4.2.

ResNet addressed the degradation problem in deep neural networks, where the model performance intrinsically degrades as the network gets deeper [57]. Rather than continue to learn high level features given the output of last layer, ResNet tries to learn its residual by using a skip connection which adds the input to the output. This approach guarantees the output of a ResNet block for not degrading by combining the newly learnt high level features with the previous low level features. Figure 4.3 from the original paper [27] presents a more intuitive depiction of its structure.

Upon adopting Residual Conv-Deconv structure [18], our adjustment is mainly due to the characteristics of our data. In their work, Brunner et al. [8] had to process piano roll data transformed from MIDI files with a height of only 84, as they only selected 84 pitches in the $C1$ to $C8$ range as research objects.

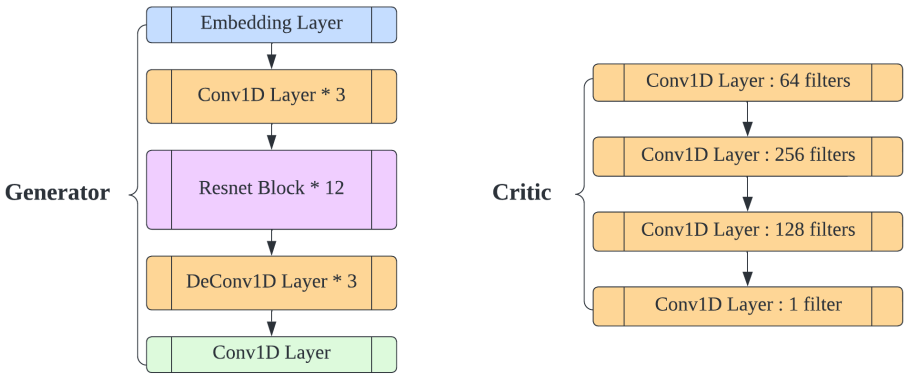

Figure 4.2: **Generator and critic (discriminator) structure. For the generator, after an embedding layer [49] and a standard Residual Conv-Deconv net [18] (but with Conv1D layers [52]), an extra Conv1D with 1512 filters (equals to the vocabulary size) is used with Sigmoid activation 3.1 to produce probability predictions of each event.**

However, in our project, the dictionary we use has 1,512 words, so each K-hot vector is 1,512 units in length. However, typically only 1-3 of these 1,512 binary variables have a value of 1, which results in severe data sparsity. Due to this sparsity, the model would struggle to learn useful features. As a solution, we added an Embedding layer at the beginning of the model to encode these sparse input data into non-sparse data for the ResNet part of the model to learn [64]. As for the encoded input, they can now be considered as word embeddings, hence we use Conv1D layers in place of Conv2D layers [36]. In addition to the differences in the dimensions of their filters, the principal distinction between Conv2D and Conv1D layers is that the filters of the former slide in both dimensions within a 2D space, whereas the filters of the latter slide in only one dimension. As a result, Conv2D [54] is more suited to processing spatial data, such as images, while Conv1D [52] is more apt for handling temporal data such as text and sequential data that is concentrated in one dimension. Taking text processing as an example, a Conv1D filter slides between each word embedding [28], that is, it slides along the sequence direction for each set of features in the word embedding, thus capturing the order information in the training data samples more effectively.

In this thesis, although in Section 3.2.2 we have transformed the sequence generation problem into a multi-variable classification problem, the data represented by K-hot still has sequential properties (the order relationship between each note), and it is essentially sequential data. Therefore, in order to make better use of this sequential information, Conv1D layers are used to replace Conv2D layers. Moreover, Conv1D requires fewer computational resources to run experiments compared to Conv2D.

4.3.2 Critic (Discriminator) Structure

In this model, the construction of the critic, which stands for the discriminator in a WGAN, still follows the structure based on Conv1D layers [52] in the

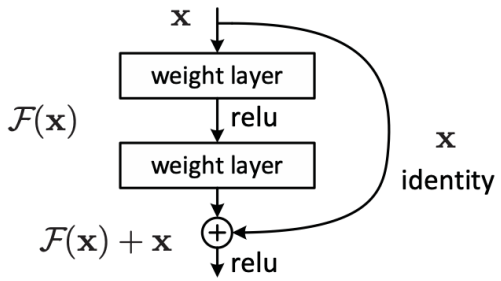

Figure 4.3: **A figure from [27], exhibiting block diagram of a ResNet where $F(x)$ denotes the high level features learnt by the weight layers. ResNet combines this high level feature with the original input and apply a ReLU [72] function to guarantee the final output for not degrading.**

generator. This is to enable the critic to make score-like predictions based on sequential information, while ensuring the generator and critic have performance in the same dimension to avoid one over dominating the other as the balance between these two components is most valuable in GAN training. The critic structure we determined after experimentation consists of four Conv1D layers (shown in Figure 4.2), while the last Conv1D layer using only one filter to produce a single value as the critic score for that input, corresponding to the first or second term in Equation 4.9 depending on the authenticity of the input sample. The reason for not adopting a more complex critic is to avoid over-fitting as the volume of training data is relatively low.

4.3.3 Loss Definition

Upon the general loss function presented in equation 4.13, we have to specifically choose a suitable mathematical form of cycle loss and identity loss regarding the nature of task and the structure of our data. As mentioned in Section 3.2.2, the token sequence generation problem has now been reduced to multi-variable binary classification problems inside each note vector. In this case, cycle loss and identity loss should use Binary Cross-Entropy (BCE) instead of Mean Absolute Error (MAE) [74]. BCE is a fundamental loss function predominantly used for binary classification tasks. Its primary role is to quantify the dissimilarity between the predicted probabilities and the actual binary labels of the data points. Expressed by equation 4.14, BCE is more suitable here as it deals with the probabilities output from the last layer of the model:

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y_i}) + (1 - y_i)\log(1 - \hat{y_i})] \tag{4.14}$$

where $L(y, \hat{y})$ stands for the BCE loss between true binary labels y and predicted probabilities $\hat{y}$. This first term of the equation becomes significant when the actual label is 1, while the second term is considered when the true label is 0. BCE penalizes the models more when they are confident but wrong, which is a desirable property in this problem setting.

4.3.4 Problems Induced by Sparsity and Their Mitigation Strategies

Unfortunately, during experiments we noticed that this model has suffered from two serious troubles:

1. **Content loss.** In other words, the musical information of the input music piece is lost during style transformation and the original melody doesn't appear in the output at all, which is totally unacceptable to a style transfer task.

2. **Mode collapse.** The outputs of the model all sounds similar, which are like small oscillations around a fixed musical pattern and being completely independent of the input.

After investigation, we discovered that the issue originated from the cycle loss and identity loss not functioning properly: the BCE reduced to a number close to zero within just a few epochs. This is obviously counter-intuitive, as the model does not seem to preserve the content of the input at all from the result, even to the point where the output is entirely unrelated to the input. Upon further investigation, we identified sparsity in the K-hot input as the root cause of the problem. Since there are only about a thousandth of 1s with the rest being 0s in the input, the generators learn to cheat the cycle loss and identity loss terms by simply generating vectors with all values very close to 0 to achieve a low BCE, rather than learning and memorizing the input content. Though the added embedding layer has encoded them into non-sparse data inside the network, it has no contribution to the loss calculations outside of the model.

Apart from the fact the Classical music samples are sparse already, for example, generally 1 to 3 one values in a 1235-width vector, the effort of mapping songs from Classical to Blues makes it even sparser by building a joint dictionary with a larger width of 1512, while the number of 1s remain unchanged.

Though being homogeneous, the outputs themselves sounds like meaningful music and are very 'Blues' even if the output probability of each token is generally close to 0, as the K-hot encoding protocol ensures that they have a nice basic musical structure, and top-k sampling ignores the low magnitude of the values of the output probabilities and picks them on the basis of their relative magnitude to each other. However, this is far from sufficient for a domain transfer task where content preservation and diversity of output are significant.

Therefore, we further propose the following attempts to mitigate such sparsity:

4.3.4.1 Loss Modification

Our initial educated guess is to attempt to mitigate the sparsity during the gradient calculation process by modifying the original loss function [29]. Based on this, we adopt two improved loss functions:

- *Weighted Binary Cross Entropy.* On top of the regular Binary Cross Entropy (BCE), we multiply it with a weight mask. This weight mask is determined by the distribution of 0s and 1s in the input data and a Lambda value: for the positions where the input data is 0, the weight is 1;

for the positions where the input data is 1, the weight is $\lambda + 1$, as shown in equation 4.15:

$$L_{C'} = -\frac{1}{N}\sum_{i=1}^{N}(\lambda x + 1)[y_i \log(\hat{y_i}) + (1 - y_i)\log(1 - \hat{y_i})] \tag{4.15}$$

where $L_{C'}$ denotes the modified cycle loss with weight mask. The symbols are defined same as in equation 4.14, except that λ is added as the weight hyper-parameter. Ideally, the content will be perfectly preserved if the chosen λ is large enough to guide the learning process. However, experiments have shown that such large λ will introduce even more instability into the GAN training, resulting in achieving balance between the generator and critic for convergence to be crucial. Therefore, it is necessary to find a suitable value for λ, which can enhance the cycle loss and identity loss as much as possible while ensuring the stability of training.

- *Summation Loss.* While the initial attempt being to establish a loss function ensuring a consistent amount of one values across the input and output, it failed as finding such a differentiable operation is difficult. As a substitution, we propose a summation loss as equation 4.16, which is trying to approximate consistency by forcing a close summation between input and output matrices.

$$L_S = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}(y_{ij} - \hat{y_{ij}}) \tag{4.16}$$

in which L_S stands for the summation loss. N and M are the number of rows and columns of the input K-hot matrix.

4.3.4.2 Skip Connection outside of the Model

After a series of experiments using the above methods, we regrettably found that while there were some improvements in content preservation, the results were still not ideal. Therefore, we proposed a somewhat circumventing method, which allowed us to achieve our current best results. This method involves adding an extra skip connection during the inference phase outside of the model, that is, simply adding the original input onto the produced output. However, it is essential to clarify that this is not a mere superposition of the input song's melody onto the output song. In other words, it is not a simplistic overlay of the melody. Because as mentioned in Section 3.1.2, an expressive token would affect all the tokens in the same note vector, so the added original content will also share the musical effect produced in the model output.

This practice would be meaningless in standard style transfer models, but it is highly applicable to our model and can overcome the issue of cycle loss ineffectiveness, achieving complete content preservation. Specifically, in Section 3.2.2, we mentioned that both the input and output K-hot matrices are composed of probability values, and we use a top-k method to sample the probability values into integer tokens during the inference stage, thereby obtaining corresponding text tokens and the final music. As the output K-hot matrix is activated by a Sigmoid 3.1 function and has a range of (0, 1), we can simply add the input

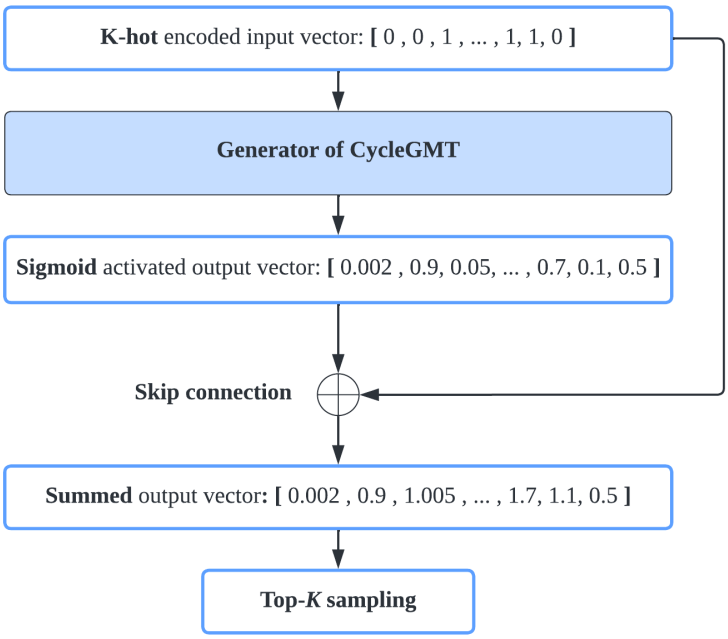

Figure 4.4: **An illustration of the skip connection we used. After added the K-hot encoded input with the Sigmoid-activated 3.1 output, we obtain the final output vector where all the elements that appeared in the original content (has value 1 in the K-hot encoded input vector) are assigned a value larger than 1. These elements are then guaranteed to be selected with highest probabilities by top-k sampling since all other elements are clipped between $(0, 1)$ by Sigmoid function. A complete content preservation is thus achieved.**

directly to the output so that the values of the items that are 1 in the original input will be greater than 1 in the added vector, thus ensuring they will be sampled by the top-k algorithm with the highest probabilities. In this way, content preservation can be achieved completely. An illustration of this skip connection is shown in Figure 4.4.

Indeed, this type of skip connection is more suitable for situations where the complexity of the original music style is lower and the complexity of the target music style is higher, as its concept is actually similar to ResNet. ResNet uses skip connections to combine the input low-level features with its generated high-level features, aiming to learn high-level features while preserving the original low-level features, preventing the degradation problem. The use of skip connections in CycleGMT can also be understood in this way: classical music inputs composed mostly of guitar note tokens forming pure melodies, with a relatively small proportion of expressive tokens (according to Table 3.1), can be considered as low-level features. On the other hand, Blues music, which relies more on expressive tokens and the atmosphere created by drums and other multiple instruments, is more complex compared to classical music. Thus, parts of Blues music apart from guitar note tokens can be considered as high-level

features. Therefore, the skip connection in CycleGMT can be interpreted as preserving the low-level music melody information while adding high-level expressive information such as guitar playing techniques, which fits well with the research goal that is to leverage the expressive guitar playing information to achieve symbolic guitar music style transfer.

4.3.4.3 The Drawback in CycleGMT

It's also important to note that the use of skip connections does not mean a complete abandonment of cycle loss and identity loss, but rather serves as an immediate enhancement for content preservation. However, this enhancement is not perfect as skip connection is not part of the training loop, therefore has no impact on the model's actual output. If the cycle loss is ineffective, then although the skip connection can forcibly preserve the content of the original music, the part of the actual model output representing learned high-level musical features, may suffer from homogeneity due to the lack of restrictions. Unfortunately, during experiments we have found that the loss modifications we made were not effective enough to validate the cycle loss and address the sparsity in the training data. As a result, mode collapse happened: The learned high-level Blues musical features in actual model outputs were all oscillations around a fixed musical pattern, regardless of the inputs. This constitutes a significant drawback in CycleGMT. Due to time constraints, we are not able to fix it in this thesis and have discussed potential directions of future work for addressing this in Chapter 7.

4.4 Conclusion

In this chapter, we have introduced the concept of Generative Adversarial Network (GAN) and its application form for domain transfer tasks, CycleGAN, whose performance lays the foundation for achieving symbolic guitar music style transfer. We then modify the model to a WGAN structure, which utilizes the Wasserstein distance to stabilize the overall network training. Upon this, we proposed CycleGMT to solve the **Research Question 2** by adopting Conv1D layers and Embedding layers in the generator structure to handle the sequentiality in our K-hot encoded text-like data.

The final challenge arose from the sparsity in the training data, which caused the cycle loss and identity loss for not working properly and resulted in mode collapse. To tackle this problem, we implemented loss modification by replacing Binary Cross Entropy (BCE) with Weighted BCE and added a summation loss. Most importantly, we introduced a skip connection outside of the model to force an effective content preservation. The preserved content can be mixed with the expressive tokens in the model output to produce style-transferred music with high quality. However, the homogeneity in the model output remains unsolved, necessitating further exploration in future work.

Chapter 5

Experimental Results

In this chapter, we provide details of training settings and present evaluation of our guitar music style transfer results to address **Research Question 3**. The evaluation methodologies of the performances of music generation or music style transfer tasks has always been an active research field, as the perception of music is generally considered subjective and so far there's no well-established objective metric for it. To solve the **Research Question 3**, we have borrowed the idea from previous works [7] [37] and designed our evaluation to consist of two parts: the objective evaluation based on a well-trained genre classifier, and the subjective evaluation by a paper survey on the quality of produced style-transferred music.

5.1 Training settings

In order to maximize the potential performance based on the structure of the model, the hyper-parameters and setting of the training experiments should be selected and tested carefully. After filtering the DadaGP dataset according to genre labels, we collected 396 Classical songs and 202 Blues songs and produce 9563 Classical samples and 5216 Blues samples according to Section 3.2.2. We also noticed that the number of Blues songs would increase to 896 if we incorporate the label 'blues_rock' while filtering. In the training of a CycleGAN model, the amount of data in domain A and domain B should be consistent, typically based on the smaller of the two. Therefore, to maximize the amount of data, we supplemented our existing Blues samples with samples tagged as 'blues_rock' until their total quantity reached 9563, matching the number of Classical samples.

Adopting the general training settings of WGAN [24], we use RMSprop [23] as the optimizer, with the critic optimized 5 times in each training round. The batch size is set to 12, with initial learning rates of 0.0004 and 0.0006 for the generator and critic, respectively. These learning rates will decay according to equation 5.1 when the epoch exceeds 20:

$$\alpha_{G,C} = \alpha \frac{101 - E}{100 - 20} \tag{5.1}$$

where $\alpha_{G,C}$ and α denotes the decayed learning rate and the initial learning rate respectively, while E is the current number of training epochs.

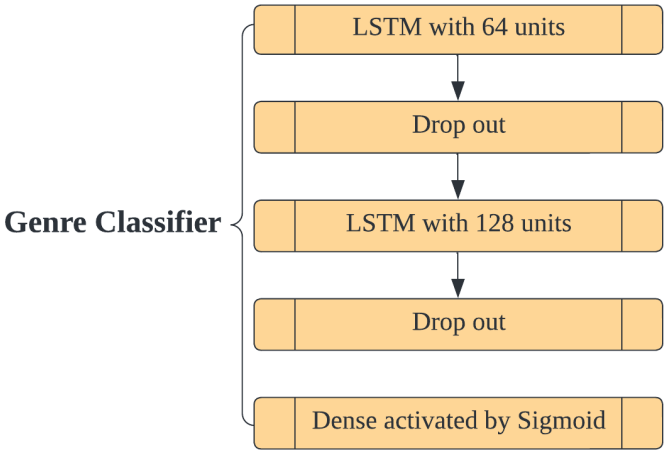

Figure 5.1: **The structure of genre classifier for objective evaluation. The binary classifier consists of two LSTM layers, two drop out layers in between and a Sigmoid 3.1 activated dense layers at the end. The classifier outputs the probability of input sample being labeled with 1, which in our case represents the probability of input sample's genre belonging to Blues music.**

Some of the generated style-transferred music samples have been made available for listening [1].

5.2 Evaluation Methodologies

Regarding the evaluation of unsupervised music style transfer, there currently are no well-suited evaluation metrics to assess the output, as genre identification is subjective and difficult to interpret objectively. Therefore, we use a combination of both objective evaluation and subjective evaluation to validate our results.

5.2.1 Objective Test: Genre Classifier

We train a binary classifier on the Classical and Blues samples used for CycleGMT training and use it to predict the probability of the output samples belonging to Blues genre. We utilize a binary classifier composed of 2 long short-term memory (LSTM) layers [31] and 2 dropout layers [3], the structure of which is shown in Figure 5.1. We gathered predictions under different k values, with results as displayed in Table 5.1.

Furthermore, to align with our project goal which is to leverage expressive information contained in guitar tablatures to contribute to the performance of symbolic guitar music style transfer, we test the proportion of harmonized sound vectors that contain nfx : or bfx : tokens (mentioned in 3.1.2) for each output under different k values for top-k sampling method 3.1.3, which we call an nfx

[1] Generated music samples available at: `https://www.youtube.com/@zhuang4826/videos`

	Blues	**Classical**	k**=2**	k**=3**	k**=4**	k**=5**	k**=6**
Prediction score	0.974	0.026	0.547	0.836	0.915	0.954	0.974

Table 5.1: **Prediction results under different k for top-k sampling**

frequency. The results shown in Table 5.2 exhibits a clear increasing trend of such frequency when k gets larger, which indicates that the choice of k values indeed affects the amount of guitar playing techniques used in the output. An example of input and outputs under different k values are shown in Figure 5.2, Figure 5.3 and Figure 5.4 in the format of guitar roll as explained in 3.2.3.

	Blues	**Classical**	k**=2**	k**=3**	k**=4**	k**=5**	k**=6**
Average number of tokens	3.92	2.14	2	3	4	5	6
nfx frequency	0.262	0.136	0.155	0.288	0.336	0.367	0.460

Table 5.2: nfx **frequency and average number of tokens in each harmonized sound vector under different k for top-k sampling**

Taking both Table 5.1 and Table 5.2 into consideration, the prediction of probability of the output music belonging to Blues genre also increases as k gets larger. However, during experiments we noticed that while a high k value indicates complexity in each bar of output, the output may sounds noisy if the k is significantly larger than the average number of tokens in each harmonized sound vector in the input. Instead, $k = 4$ is tested to be generally good for producing music that sounds pleasant while keeping the high level features of Blues music. Thus, we chose the outputs when $k = 4$ to be the testing samples in our subjective test with a paper survey.

5.2.2 Subjective Test: Paper Survey

To further evaluate the style transfer performance of CycleGMT, we design a paper survey (presented fully in Appendix A) and conduct among groups of people with different musical involvement levels, of which these differences are quantitatively analysed by questions extracted from Golden Smith [47]. We then ask the participants to listen to a clip from *Für Elise* [63] by Ludwig van Beethoven, as well as a clip from *Bad Feeling Blues* [4] written by Blind Blake as a reference of Blues music, prepared for those participants who barely knows anything about Blues. After that we ask the participants to listen to 3 Blues-remix versions of that *Für Elise* clip, with one of them being generated by CycleGMT and the other two being composed by human artists and uploaded on MuseScore [71] [62]. For each remix, before getting informed that one of the remixes is generated by CycleGMT, the participants should rate out of 5 for respectively how 'Blues' it sounds like, as well as their subjective audience perception. When all the ratings have been done, they are asked to make judgments of which remix is most likely to be generated by our model in two scenarios: without and with the option to listen to those remixes again. Without playing the music samples to them again, the participants would make decisions merely base on their first impression to see if the generated style-transferred

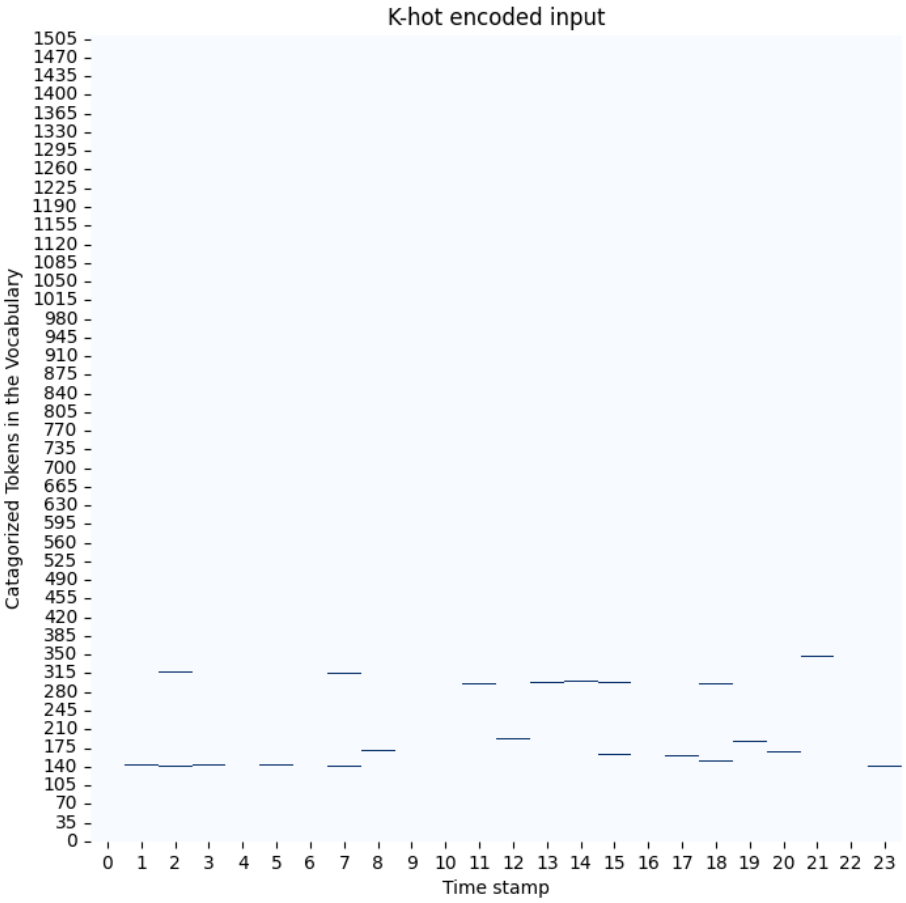

Figure 5.2: **Visualization of a K-hot encoded input in the form of guitar roll, where the X-axis represents time stamps and Y-axis represents categorized tokens in the vocabulary and the colored lines mark the presence of a token in the certain period.**

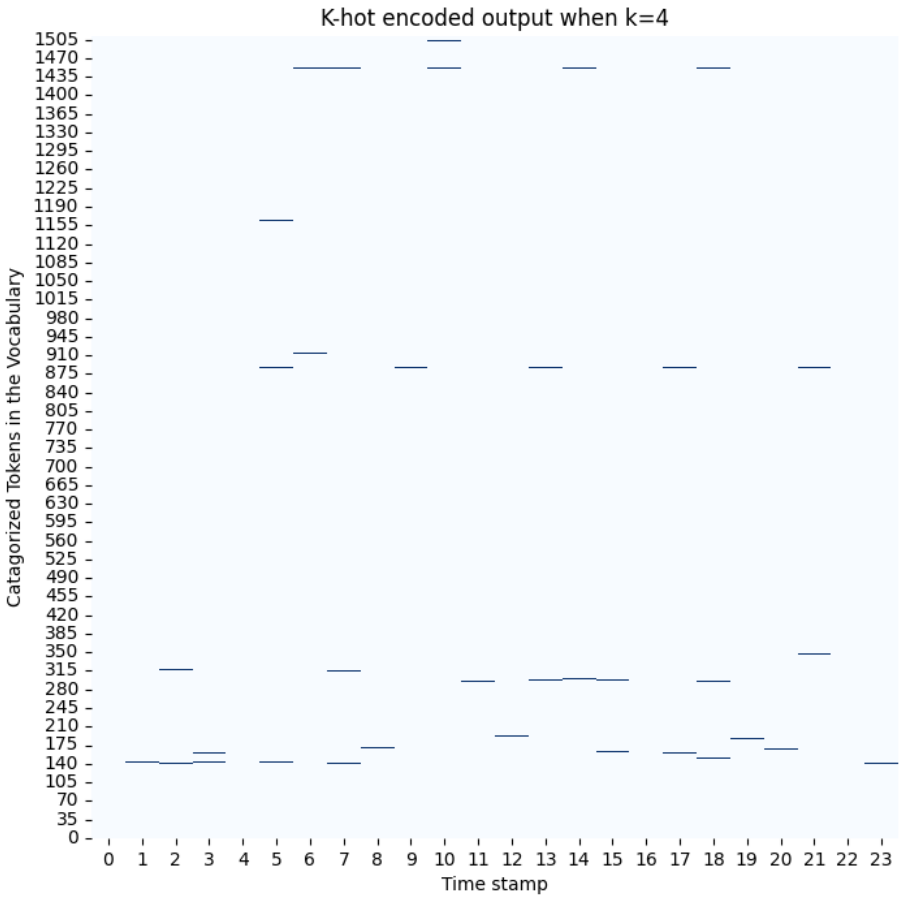

Figure 5.3: **Visualization of the K-hot encoded output when k is set to 4 for top-k sampling. The figures shows that the original content of the input shown in Figure 5.2 are completely preserved in the output and some extra tokens are added to achieve style transfer.**

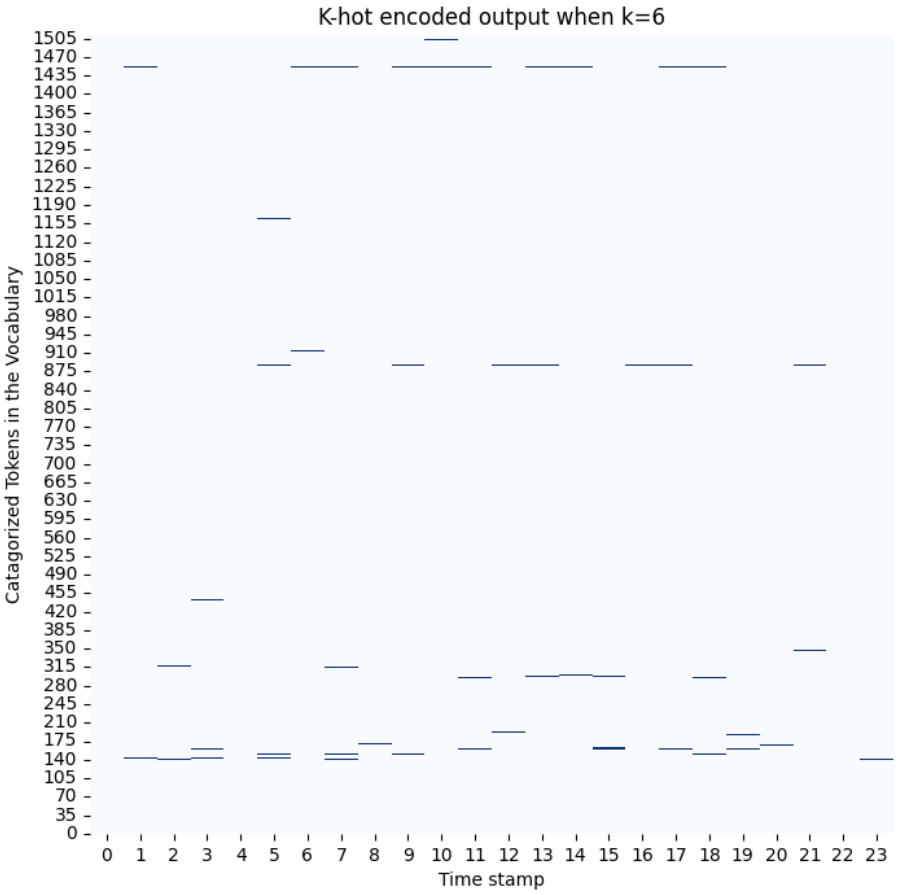

Figure 5.4: **Visualization of the K-hot encoded output when k for top-k sampling is set to 6. More tokens are appearing in this output comparing to that in Figure 5.3 when $k = 4$, indicating a higher complexity in terms of musical arrangement. This demonstrates that the choice of k can indeed control the complexity of the output music.**

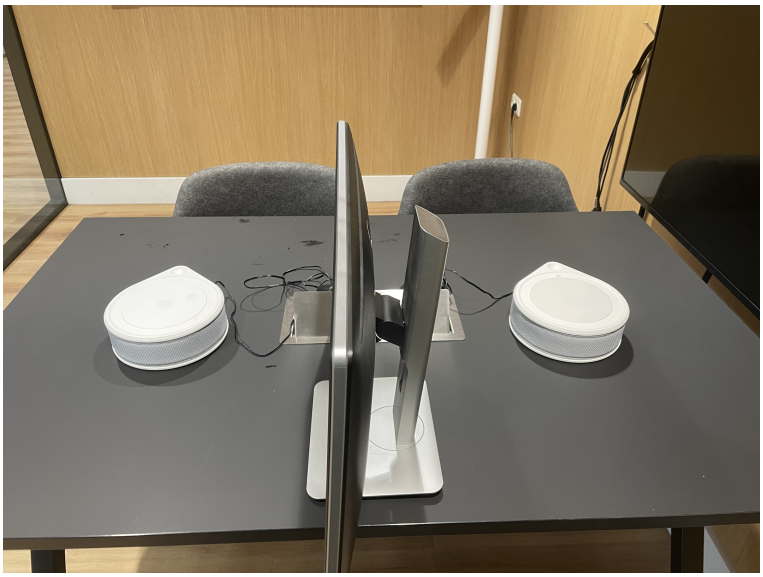

Figure 5.5: **The experimental setup in a meeting room in TU Delft. Two speakers are used to play the testing samples.**

music could be easily recognised. When listening to each of the remixes again, the participants will listen carefully to each sample with the knowledge that one of them is computer-generated music, at which point the quality of the generated music will be rigorously tested for recognisable flaws.

We were finally able to gather 57 participants comprising individuals of various ages and genders, including those with or without musical experience. To ensure the consistency of the results, we conducted the experiments within the same room, maintaining uniformity in the experimental facilities and environmental conditions. The experiment could accommodate 1-2 individuals at a time, and the setup of our experimental environment is depicted in the following Figure 5.5:

5.2.2.1 Overall Results

Figure 5.6 shows the overall average of the two ratings and Figure 5.7 exhibits the proportion of chosen options for the two decisions made, with re-listening to the three remixes and without, respectively. The results have shown that the quality of the generated style-transferred music is competitive to the two other human-made versions, even has higher average than the first remix in both ratings. As for to identify the generated music from human-made remixes, only about 35% of the participants provided the correct answer, indicating the high quality of our generated samples. People found it challenging to distinguish music generated by CycleGMT from human compositions, even when given a chance to carefully listen to them again.

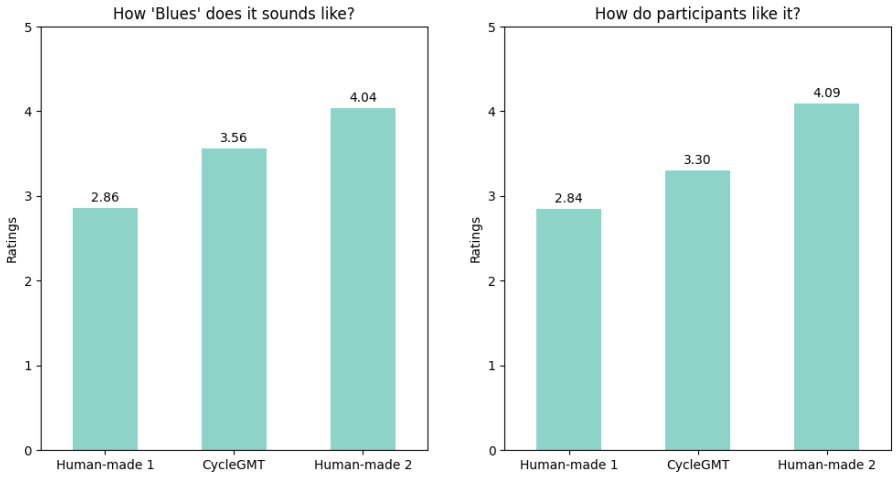

Figure 5.6: **The average over ratings (1 to 5) given by 57 participants, for the two survey questions respectively. The results indicate that CycleGMT performs well on both the first and second questions, scoring lower than the $2nd$ human-made remix but higher than the $1st$ one. This suggests that the Blues style transfer produced by CycleGMT is compatible with human-made compositions, in terms of both genre transferring and audience perception.**

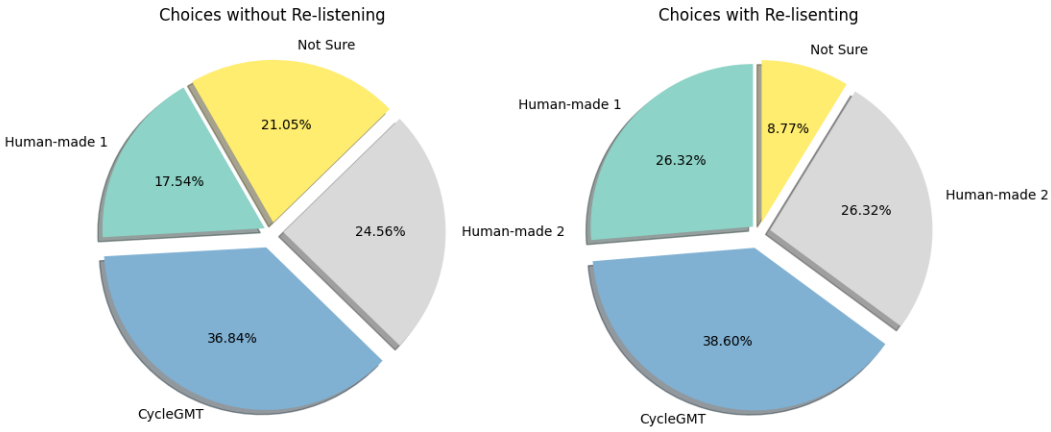

Figure 5.7: **The distributions of the two decisions made by participants on which music sample was generated by CycleGMT among the three Blues remixes we offered in the survey. Without and with re-listening to the three Blues remixes, only 36.8% and 38.6% of participants correctly distinguished the CycleGMT-generated sample from the other two human-made Blues remixes. Also, the proportion of correct choices made by participants increases after the re-listening. This indicates that the style transfer generated by CycleGMT is more likely to confuse people by the first impression, and still has some recognizable flaws if carefully listened to.**

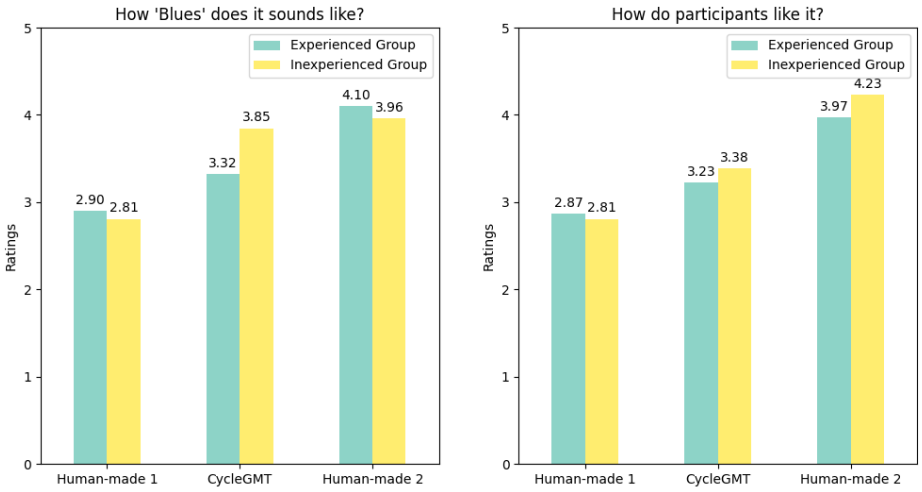

Figure 5.8: **The different ratings given by musically experienced and inexperienced group of participants. The left figure indicates that compared to the inexperienced group, the musically experienced participants gave more positive evaluations on whether the two human-made compositions were Blues. Conversely, they gave significantly more negative evaluations for the sample generated by CycleGMT. For the right figure, both experienced and inexperienced group gave competitive ratings for samples produced by CycleGMT comparing to the other two human-composed remixes, while the rating of the experienced group is still relatively lower.**

5.2.2.2 Different results for Musical Experienced and Inexperienced Groups of Participants

Moreover, we further investigate whether individuals with different musical backgrounds perceive the music generated by CycleGMT differently. We categorized the participants into two groups: The first group comprised individuals who had practiced a musical instrument or were frequent listeners of blues. This group, with a stronger musical background, could potentially make judgments based more on their knowledge of music theory or their familiarity with the conventions of blues compositions. Their judgments thus hold relatively higher value in assessing the authenticity and musicality of the generated music. The second group consisted of participants who had no experience with any musical instrument and were not particularly familiar with blues. Their ratings and choices were likely to be more influenced by subjective listening preferences, and these opinions are also of great significance as their perception represents the general group. The average rating for both questions of both musical experienced and inexperienced groups are given in Figure 5.8, and the distribution of the choices made by both groups after re-listening to the three remixes is shown in Figure 5.9.

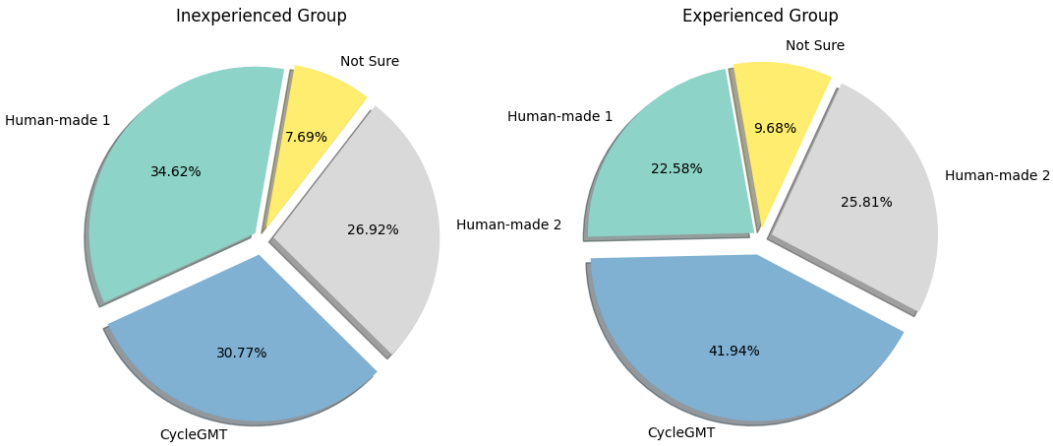

Figure 5.9: **The distribution of choices after re-listening to the three remixes for musically experienced group and inexperienced group. Comparing the two pie charts, around 42% of participants in the experienced group correctly identified the remix generated by CycleGMT, while this number is reduced to about 31% in the inexperienced group. This suggests that potential flaws in the CycleGMT-generated samples are harder to elude the scrutiny of relatively more musically knowledgeable individuals.**

5.3 Conclusion

In this chapter, we conducted both objective evaluation and subjective evaluation to demonstrate the performance of our symbolic guitar music style transfer model, CycleGMT 4.3, as well as the capability of K-hot encoding and top-k sampling 3.1.3, addressing our **Research Question 3**.

First, we are going to discuss the result of objective evaluation with genre classifier. Table 5.1 has shown that the style transfers produced have a high probability to be predicted as Blues by a well-trained genre classifier, especially when the k value for top-k sampling is set larger than 4. This demonstrates that the fake Blues music samples generated by CycleGMT is able to cheat a genre classifier which is trained on the same data domain. Table 5.2 has exhibited an increasing trend of nfx frequency when k is getting larger. These two tables together indicate that different k for top-k sampling can indeed control the musical complexity of an output sample by adjusting the amount of nfx and bfx tokens included in it, and thus determine its probability of being predicted as Blues music by the genre classifier. Figure 5.2, 5.3 and 5.4 together provides visualisation of an input, its output when k for top-k sampling equals to 4 and the output when k equals to 6, showing the difference they made in the form of guitar roll.

Then we are analyzing the outcome of subjective evaluation with paper survey. When it comes to the subjective evaluation, Figure 5.6 shows that the participants are giving competitive ratings for the CyleGMT-generated Blues remix when comparing to another two human-made Blues remixes, in terms of how 'Blues' the remix sounds and the subjective perception of participants to that

remix. Figure 5.7 illustrated that participants found it challenging to correctly identify the Blues remix generated by CycleGMT, even if they had a second chance for listening to the three Blues remixes once again. Taking all figures above into consideration, all of them demonstrated a high quality of Blues style transfer generated by CycleGMT that is competitive to human compositions. Figure 5.8 and Figure 5.9 exhibited the difference of perceptions between musically experienced and inexperienced participants. Experienced participants, compared to their inexperienced counterparts, are more adept at discerning the remix produced by CycleGMT from the other two human-composed remixes, and they tend to give it a lower score on the question of "how 'Blues' it sounds." Nonetheless, the evaluation of the CycleGMT-generated remix by the experienced group remained notably high. Moreover, only 40% of them successfully identified the remix generated by CycleGMT.

In general, the 'fake' Blues music generated from Classical music by CycleGMT successfully fooled the binary genre classifier, and its musical complexity is demonstrated to be controlable by the k value of top-k sampling. As for audience perception, the participants of the paper survey gave high ratings that is competitive to human-made remixes for CycleGMT-generated sample. Also, both musically experienced and inexperienced participants found it challenging to correctly identify the CycleGMT-generated sample from human compositions.

Chapter 6

Conclusions

In this thesis, symbolic guitar music style transfer that transfers Classical guitar music to Blues guitar music was implemented leveraging DadaGP [60], which is a symbolic guitar music dataset with over 26K songs presented in guitar tablatures. These guitar tablatures are also rendered into text-like tokens. Consequently, our initial educated attempt was to encode these tokens by integer encoding and employ text generation models, such as a Transformer [64], as the generator of a CycleGAN. However, such attempts failed due to the unique structure and discrete nature of DadaGP data. Thus, we proposed the use of K-hot encoding to transform each original token sequence into a K-hot vector with fixed length, transitioning the sequence generation task to a multi-variable binary classification task within each K-hot vector, predicting the probability of each token appearing in the output music piece. The output K-hot vectors are then sampled using the top-k sampling method [38] to select the k tokens with the highest probabilities, reassembling them into the generated style-transferred music. This approach effectively addressed the issues encountered with integer encoding and sequence generation, converting the discrete data flow into a continuous one while fitting perfectly with the unique format of DadaGP dataset. Additionally, the musical complexity of the style-transferred output can be controlled by setting different values for k used in top-k sampling.

Based on our encoding protocol, and inspired by the work of Brunner et al. [8], we adopted CycleGAN as our backbone model. For the generator structure, we chose a Residual Conv-Deconv [18] net composed of Conv1D layers [52] and an Embedding layer [49] at the front end. Furthermore, we updated our backbone structure from a standard CycleGAN to a Wasserstein GAN to stabilise training and address the mode collapse problem we encountered in our early experiments. The last challenge was to tackle the sever sparsity in the K-hot encoded data, which caused the cycle loss to be defective. We attempted to validate the cycle loss by modifying its definition, and introduced a skip connection which connects the input with the model output to enhance content preservation. As a result, the contents were successfully preserved and thus music style transfers with high quality were produced. However, the cycle loss remained ineffective, causing the model output (without connecting with the input) being oscillations around a certain pattern, regardless the input.

It is reassuring to note that even though the issue of model output homogenization was not resolved, the style transfer results obtained merely by enforcing content preservation through skip connections have already achieved commendable outcomes in our evaluation experiments. As the perception of quality of music style transfer is generally subjective can hardly be defined by mathematical metrics, we follow the idea of paper [37] and designed the evaluation to consist of one objective evaluation based on a genre classifier and one subjective evaluation by conducting a paper survey among 57 participants with different musical background. Results of the objective evaluation have shown that the outputs have a high probability to be predicted as Blues by the genre classifier 5.1, and demonstrated the controlability over musical complexity by choosing different k for top-k sampling. In the subjective evaluation, participants of the survey gave high ratings for the quality of the generated style-transferred music, and found it challenging to correctly identify the generated one from other two human-composed remixes.

Back to the research question:

> ' *How to achieve symbolic guitar music style transfer based on guitar tablatures, in order to leverage the expressive guitar playing information and produce playable guitar tablatures?* '

Generally speaking, we have successfully provided a solution to it in this thesis by solving the three sub-questions respectively in Chapter 3, 4 and 5. We proposed K-hot encoding with probability modeling, together with CycleGMT to achieve symbolic guitar music style transfer utilizing expressive guitar playing information contained in DadaGP dataset, and the superiority of the combination of K-hot encoding, probability modeling and top-K sampling ensures the output tablatures to be generally playable and have reasonable structures. However, this solution is not perfect, as the issue of cycle loss inefficacy due to sparsity in the K-hot encoded data and the mode collapse it caused is a matter of future research.

Chapter 7

Discussion and Future Work

In this Chapter, we mainly discuss the limitations of the current work and propose possible directions for future improvement. The most significant challenge faced in this thesis is the persistent issue of cycle loss inefficacy stemming from the sparsity of K-hot encoded training data, a subject which was explained in Chapters 4 and 5. Although we introduced an external skip connection between the original input and the model output to mitigate the effects of poor content preservation resulting from ineffective cycle loss, the standalone model output still demonstrates symptoms of mode collapse. Specifically, the cycle loss becomes ineffective, leading the output to be largely independent of the input, resulting in homogenized model outputs.

Another limitation of this thesis pertains to the selection of the model and its training precision. These limitations were primarily consequences of time constraints and computational resource limitations. Given the considerable wait times associated with supercomputer server queues, a single experiment typically takes between 4-7 days. This prolonged timeline restricted us from exploring other models with potential, such as a Variational Auto-Encoder (VAE), and confined us to a limited set of parameter combinations.

Based on this, we propose the following possible directions of future work:

7.1 Regarding sparse data

- Further loss modification. In Section 4.3.4.1, we have modified the cycle loss to a combination of Weighted Binary Cross Entropy (WBCE) and summation loss. However, the effectiveness of this approach remains limited. One primary impediment lies in the tuning of the 'weight' parameter within the WBCE. If this weight is set too low, it fails to enhance the cycle loss. Conversely, an excessively high weight can destabilize the training process. As of the completion of this thesis, an optimal weight magnitude remains elusive. Future research endeavors might explore the potential settings of this weight parameter. Moreover, it could be beneficial to seek loss functions that are more adept than WBCE, ones intrinsically designed

to mitigate the ramifications of sparsity.

- Dictionary compression. Another attempt could be to compress the vocabulary size of K-hot encoding, resulting in K-hot vectors being much more narrower. Once the K-hot vectors are narrow enough, the objective is to abandon the current external skip connection to allow a full change on melody content. Our current attempt on dictionary compression is to prune the instrumentation information from note tokens. For example, modify *bass : note : s5 : s0* to *s5: s0*, which drops the instrumentation feature but reduces the token space from 1512 to 401. However, further experiments have proved that a 401-width dictionary is still to sparse for learning reasonable style transfers. In comparison, earlier work employing MIDI piano rolls [7] [8] utilized pure pitch as their vocabulary, spanning a mere width of 84. Consequently, identifying novel methods for vocabulary compression might offer a foundational solution to the challenges posed by data sparsity. It is worth noting, however, that such an approach might likely result in a loss of musical information inherent in the dataset.

7.2 Regarding model structure

- To seek better parameter combinations. Given more time and computational resources, it is plausible to identify superior parameter combinations that could potentially unlock improved performance from the model.
- Explore alternative models with potential in style transfer. In the realm of style transfer, VAEs stand out as a promising area for exploration [40]. Due to time constraints, we were unable to experiment much with it as a backbone for our model. However, future endeavors can consider combining VAEs with transformers, akin to approaches observed in [70], to construct generative models, and potentially undertake sequential modeling experiments.

7.3 Regarding experiments and evaluation

- Conduct more experiments on different genres. Currently we only explored the genre transfer between Classical music and Blues music, other potential candidates could be Rock, Folk or Jazz.
- Modify the binary genre classifier used for the objective evaluation to a multi-genre classifier with more sophisticated structure. Regarding the multi-classifier, we have already made attempts. But the multi-classifier tends to classify based on the melodic information composed of note tokens, instead of focusing on the differences between the expressive tokens primarily used in different genres. This is due to that although there are differences in the proportions of expressive tokens between different genres, they are all predominantly occupied by note tokens 3.1. This leads to overfitting and prevents us from achieving the desired classification results. Future work could focus on trying out different structure of a multi-genre classifier and use machine learning technologies to overcome the overfitting, in order to make more precise genre classifications.

Bibliography

[1] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[3] Pierre Baldi and Peter J Sadowski. Understanding dropout. *Advances in neural information processing systems*, 26, 2013.

[4] Blind Blake. Bad feeling blues, 1927.

[5] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. Deep learning techniques for music generation–a survey. *arXiv preprint arXiv:1709.01620*, 2017.

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

[7] Gino Brunner, Andres Konrad, Yuyi Wang, and Roger Wattenhofer. Midi-vae: Modeling dynamics and instrumentation of music with applications to style transfer, 2018.

[8] Gino Brunner, Yuyi Wang, Roger Wattenhofer, and Sumu Zhao. Symbolic music genre transfer with cyclegan, 2018.

[9] Nicholas J Bryan, Gautham J Mysore, and Ge Wang. Source separation of polyphonic music with interactive user-feedback on a piano roll display. In *ISMIR*, pages 119–124, 2013.

[10] Chung-Ching Chang, David Reitter, Renat Aksitov, and Yun-Hsuan Sung. Kl-divergence guided temperature sampling, 2023.

[11] 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.

[12] Ondřej Cífka, Alexey Ozerov, Umut Şimşekli, and Gael Richard. Self-supervised vq-vae for one-shot music style transfer. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 96–100. IEEE, 2021.

[13] Ondřej Cífka, Umut Şimşekli, and Gaël Richard. Supervised symbolic music style translation using synthetic data. *arXiv preprint arXiv:1907.02265*, 2019.

[14] Ondřej Cífka, Umut Şimşekli, and Gaël Richard. Groove2groove: One-shot music style transfer with supervision from synthetic data. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2638–2650, 2020.

[15] Shuqi Dai, Zheng Zhang, and Gus G Xia. Music style transfer: A position paper. *arXiv preprint arXiv:1803.06841*, 2018.

[16] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[17] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[18] Damien Fourure, Rémi Emonet, Elisa Fromont, Damien Muselet, Alain Tremeau, and Christian Wolf. Residual conv-deconv grid network for semantic segmentation. *arXiv preprint arXiv:1707.07958*, 2017.

[19] Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*, 2017.

[20] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[21] Michael Good et al. Musicxml: An internet-friendly format for sheet music. In *Xml conference and expo*, pages 03–04. Citeseer, 2001.

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[23] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[24] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[25] William W Hager. Lipschitz continuity for constrained processes. *SIAM Journal on Control and Optimization*, 17(3):321–338, 1979.

[26] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International workshop on artificial neural networks*, pages 195–201. Springer, 1995.

[27] 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.

[28] Tong He, Weilin Huang, Yu Qiao, and Jian Yao. Text-attentional convolutional neural network for scene text detection. *IEEE transactions on image processing*, 25(6):2529–2541, 2016.

[29] Yaoshiang Ho and Samuel Wookey. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE access*, 8:4806–4813, 2019.

[30] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[31] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[32] Yu-Siang Huang and Yi-Hsuan Yang. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM international conference on multimedia*, pages 1180–1188, 2020.

[33] Yun-Ning Hung, I-Tung Chiang, Yi-An Chen, and Yi-Hsuan Yang. Musical composition style transfer via disentangled timbre representations, 2019.

[34] Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. Deep learning for text style transfer: A survey. *Computational Linguistics*, 48(1):155–205, 2022.

[35] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review. *IEEE transactions on visualization and computer graphics*, 26(11):3365–3385, 2019.

[36] Yoon Kim. Convolutional neural networks for sentence classification, 2014.

[37] Junghyun Koo, Marco A. Martínez-Ramírez, Wei-Hsiang Liao, Stefan Uhlich, Kyogu Lee, and Yuki Mitsufuji. Music mixing style transfer: A contrastive learning approach to disentangle audio effects. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

[38] Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International Conference on Machine Learning*, pages 3499–3508. PMLR, 2019.

[39] Youssef Kossale, Mohammed Airaj, and Aziz Darouichi. Mode collapse in generative adversarial networks: An overview. In *2022 8th International Conference on Optimization and Applications (ICOA)*, pages 1–6. IEEE, 2022.

[40] Zhi-Song Liu, Vicky Kalogeiton, and Marie-Paule Cani. Multiple style transfer via variational autoencoder. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2413–2417. IEEE, 2021.

[41] Chien-Yu Lu, Min-Xin Xue, Chia-Che Chang, Che-Rung Lee, and Li Su. Play as you like: Timbre-enhanced multi-modal music style transfer. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 1061–1068, 2019.

[42] Tyler McAllister and Björn Gambäck. Music style transfer using constant-q transform spectrograms. In *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*, pages 195–211. Springer, 2022.

[43] ML Menéndez, JA Pardo, L Pardo, and MC Pardo. The jensen-shannon divergence. *Journal of the Franklin Institute*, 334(2):307–318, 1997.

[44] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018.

[45] Gautam Mittal, Jesse Engel, Curtis Hawthorne, and Ian Simon. Symbolic music generation with diffusion models, 2021.

[46] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. A universal music translation network, 2018.

[47] Daniel Müllensiefen, Bruno Gingras, Jason Musil, and Lauren Stewart. The musicality of non-musicians: An index for assessing musical sophistication in the general population. *PloS one*, 9(2):e89642, 2014.

[48] Meinard Müller. *Music Representations*, pages 1–37. Springer International Publishing, Cham, 2015.

[49] Masato Neishi, Jin Sakuma, Satoshi Tohda, Shonosuke Ishiwatari, Naoki Yoshinaga, and Masashi Toyoda. A bag of useful tricks for practical neural machine translation: Embedding layer initialization and large batch size. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*, pages 99–109, 2017.

[50] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[51] Marco Pasini. Melgan-vc: Voice conversion and audio style transfer on arbitrarily long samples using spectrograms, 2019.

[52] Sirawan Phiphitphatphaisit and Olarik Surinta. Deep feature extraction technique based on conv1d and lstm network for food image recognition. *Engineering and Applied Science Research*, 48(5):581–592, 2021.

[53] Rohit Prabhavalkar, Kanishka Rao, Tara N Sainath, Bo Li, Leif Johnson, and Navdeep Jaitly. A comparison of sequence-to-sequence models for speech recognition. In *Interspeech*, pages 939–943, 2017.

[54] Naris Prombut, Sajjaporn Waijanya, and Nuttachot Promrit. Feature extraction technique based on conv1d and conv2d network for thai speech emotion recognition. In *2021 5th International Conference on Natural Language Processing and Information Retrieval (NLPIR)*, pages 54–60, 2021.

[55] Queen. We will rock you. Album: News of the World, 1977.

[56] Sonia Rodríguez, Emilia Gómez Gutiérrez, and Helena Cuesta. Automatic transcription of flamenco guitar falsetas. In *Holzapfel A, Pikrakis A, editors. Proceedings of the 8th International Workshop on Folk Music Analysis; 2018 Jun 26-29; Thessaloniki, Greece. Greece: Aristotle University of Thessaloniki; 2018.* Folk Music Analysis, 2018.

[57] Prasun Roy, Subhankar Ghosh, Saumik Bhattacharya, and Umapada Pal. Effects of degradations on deep neural network architectures. *arXiv preprint arXiv:1807.10108*, 2018.

[58] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40:99–121, 2000.

[59] S. Abalumov. Pyguitarpro. `https://github.com/Perlence/PyGuitarPro`. [Online].

[60] Pedro Sarmento, Adarsh Kumar, CJ Carr, Zack Zukowski, Mathieu Barthet, and Yi-Hsuan Yang. Dadagp: A dataset of tokenized guitarpro songs for sequence models. *arXiv preprint arXiv:2107.14653*, 2021.

[61] Pedro Sarmento, Adarsh Kumar, Yu-Hua Chen, CJ Carr, Zack Zukowski, and Mathieu Barthet. Gtr-ctrl: Instrument and genre conditioning for guitar-focused music generation with transformers. In Colin Johnson, Nereida Rodríguez-Fernández, and Sérgio M. Rebelo, editors, *Artificial Intelligence in Music, Sound, Art and Design*, pages 260–275, Cham, 2023. Springer Nature Switzerland.

[62] Daniel Thrasher. Für elise blues remix version 2. howpublished = https://musescore.com/user/37001223/scores/6555730, 2021.

[63] Ludwig van Beethove. Für elise, 1867.

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[65] Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. Do nlp models know numbers? probing numeracy in embeddings, 2019.

[66] Liwei Wang, Yan Zhang, and Jufu Feng. On the euclidean distance of images. *IEEE transactions on pattern analysis and machine intelligence*, 27(8):1334–1339, 2005.

[67] Wikipedia contributors. Guitar pro — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Guitar_Pro&oldid=1169719999`, 2023. [Online].

[68] Wikipedia contributors. Piano roll — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Piano_roll&oldid=1160544889, 2023. [Online].

[69] Wikipedia contributors. Staff (music) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Staff_(music)&oldid=1163608166, 2023. [Online].

[70] Shih-Lun Wu and Yi-Hsuan Yang. Musemorphose: Full-song and fine-grained piano music style transfer with one transformer vae. *arXiv preprint arXiv:2105.04090*, 2021.

[71] Xonacatepec. Für elise blues remix version 1. howpublished = https://musescore.com/user/41226/scores/114081, 2013.

[72] Jin Xu, Zishan Li, Bowen Du, Miaomiao Zhang, and Jing Liu. Reluplex made more practical: Leaky relu. In *2020 IEEE Symposium on Computers and communications (ISCC)*, pages 1–7. IEEE, 2020.

[73] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[74] Zhilu Zhang and Mert R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels, 2018.

[75] 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.

Appendix A

The Paper Survey

In this appendix, we provide the design of questions in the paper survey conducted for subjective evaluation.

Please circle the most appropriate category:

1. What are the musical genres you mainly listen to? **Pop** / **Rock, Metal** / **Blues, Soul** / **Folk** / **Classical Music**
2. I listen to music for **0-30 min** / **30-60 min** / **60-90 min** / **2 hrs or more** per day.
3. I engaged in regular, daily practice of a musical instrument (including vocal) for **0** / **1-3** / **4-9** / **10 or more** years. The instrument I play best is (including vocal): _____
4. Have you ever played guitar or is familiar with guitar music? **Yes** / **No**

Now Please listen to a clip from Für Elise and a piece of Blues music for reference...

Please listen to the 1st Blues remix version and answer the following questions:

1. How much does it sounds like Blues? **1** / **2** / **3** / **4** / **5**
2. Would you consider it a good Blues remix? Please rate it in terms of auditory perception and quality: **1** / **2** / **3** / **4** / **5**

Please listen to the 2nd Blues remix version and answer the following questions:

1. How much does it sounds like Blues? **1** / **2** / **3** / **4** / **5**

2. Would you consider it a good Blues remix? Please rate it in terms of auditory perception and quality: **1** / **2** / **3** / **4** / **5**

Please listen to the 3rd Blues remix version and answer the following questions:

1. How much does it sounds like Blues? **1** / **2** / **3** / **4** / **5**
2. Would you consider it a good Blues remix? Please rate it in terms of auditory perception and quality: **1** / **2** / **3** / **4** / **5**

Now that you have listened to 3 versions of the Blues remix:

1. Without listening to them again, which one do you believe was generated by AI?
   1st remix / 2nd remix / 3rd remix / Can not determine with certainty.
2. Listen to them again, now which one do you believe was generated by AI?
   1st remix / 2nd remix / 3rd remix / Can not determine with certainty.

This is the end of the survey. Thanks for your participation.

Study contact details for future information: x.zhuang-1@student.tudelft.nl