BNAIC BENELEARN 2020

Proceedings

November 19–20, 2020

Universiteit Leiden
The Netherlands

Editors: Lu Cao, Walter Kosters and Jefrey Lijffijt

BNAIC/BeneLearn 2020

Proceedings

Leiden, the Netherlands
November 19–20, 2020

Editors: Lu Cao, Walter Kosters and Jefrey Lijffijt

http://www.bnaic.eu

General Chairs

Mitra Baratchi — Leiden University
Jan N. van Rijn — Leiden University

Publication Chair

Frank Takes — Leiden University

Local Organization

Gerrit-Jan de Bruin
Michael Emmerich
Mischa Hautvast
Jaap van den Herik
Mike Huisman
Matthias König
Anna Louise Latour
Enrico Liscio
Michiel van der Meer
Matthias Müller-Brockhausen
Jayshri Murli
Marloes van der Nat
Aske Plaat
Mike Preuss
Peter van der Putten
Suzan Verberne
Jonathan Vis
Hui Wang

Program Committee

Martin Atzmueller — Tilburg University
Bernard de Baets — Ghent University
Mitra Baratchi — Leiden University
Souhaib Ben Taieb — Université de Mons
Floris Bex — Utrecht University
Hendrik Blockeel — Katholieke Universiteit Leuven
Koen van der Blom — Leiden University
Bart Bogaerts — Vrije Universiteit Brussel
Tibor Bosse — Vrije Universiteit Amsterdam
Bert Bredeweg — University of Amsterdam
Egon L. van den Broek — Utrecht University
Lu Cao — Leiden University
Tom Claassen — Radboud University
Walter Daelemans — University of Antwerp
Mehdi Dastani — Utrecht University
Kurt Driessens — Maastricht University
Tim van Erven — Leiden University
Ad Feelders — Utrecht University
George H. L. Fletcher — Eindhoven University of Technology
Benoît Frénay — Université de Namur
Lieke Gelderloos — Tilburg University
Pierre Geurts — University of Liège
Nicolas Gillis — Université de Mons

Nick Harley — Vrije Universiteit Brussel
Frank van Harmelen — Vrije Universiteit Amsterdam
Andrew Hendrickson — Tilburg University
Tom Heskes — Radboud University
Arjen Hommersom — Open University of the Netherlands
Mark Hoogendoorn — Vrije Universiteit Amsterdam
Walter Kosters — Leiden University
Johan Kwisthout — Radboud University
Bertrand Lebichot — Université Libre de Bruxelles
John Lee — Université Catholique de Louvain
Jan Lemeire — Vrije Universiteit Brussel
Tom Lenaerts — Université Libre de Bruxelles
Jefrey Lijffijt — Ghent University
Gilles Louppe — University of Liège
Peter Lucas — Leiden University
Bernd Ludwig — University Regensburg
Elena Marchiori — Radboud University
Wannes Meert — Katholieke Universiteit Leuven
Vlado Menkovski — Eindhoven University of Technology
John-Jules Meyer — Utrecht University
Arno Moonens — Vrije Universiteit Brussel
Nanne van Noord — University of Amsterdam
Frans Oliehoek — Delft University of Technology
Aske Plaat — Leiden University
Eric Postma — Tilburg University
Henry Prakken — University of Utrecht and University of Groningen
Mike Preuss — Leiden University
Peter van der Putten — Leiden University and Pegasystems
Jan N. van Rijn — Leiden University
Yvan Saeys — Ghent University
Chiara F. Sironi — Maastricht University
Evgueni Smirnov — Maastricht University
Gerasimos Spanakis — Maastricht University
Jennifer Spenader — University of Groningen, AI
Johan Suykens — Katholieke Universiteit Leuven
Frank Takes — Leiden University and University of Amsterdam
Dirk Thierens — Utrecht University
Leon van der Torre — University of Luxembourg
Remco Veltkamp — Utrecht University
Joost Vennekens — Katholieke Universiteit Leuven
Arnoud Visser — University of Amsterdam
Marieke van Vugt — University of Groningen
Willem Waegeman — Ghent University
Hui Wang — Leiden University
Gerhard Weiss — University Maastricht
Marco Wiering — University of Groningen
Jef Wijsen — Université de Mons
Mark H. M. Winands — Maastricht University
Marcel Worring — University of Amsterdam
Menno van Zaanen — South African Centre for Digital Language Resources
Yingqian Zhang — Eindhoven University of Technology

Preface

In 2020, the 32nd edition of BNAIC—the annual Benelux Conference on Artificial Intelligence—is organized together with the 29th edition of BeneLearn—the annual Belgian-Dutch Conference on Machine Learning—by the Leiden Institute of Advanced Computer Science (LIACS) of Leiden University, under the auspices of the Benelux Association for Artificial Intelligence (BNVKI).

The conference was scheduled to take place in Corpus, Leiden, but due to the corona virus pandemic and limitations on the organization of events, the conference was organized fully online, for the first time in its history. It took place on Thursday, November 19 and Friday, November 20, 2020. The conference included keynotes by invited speakers, so-called FACt talks, research presentations, a social programme, and a "society and business" afternoon.

The three keynote speakers at the conference were:

- Joost Batenburg, Leiden University
  Challenges in real-time 3D imaging, and how machine learning comes to the rescue
- Gabriele Gramelsberger, RWTH Aachen University
  Machine learning-based research strategies — A game changer for science?
- Tom Schaul, Google DeepMind, London
  The allure and the challenges of deep reinforcement learning

Three FACt talks (FACulty focusing on the FACts of Artificial Intelligence) were scheduled:

- Luc De Raedt, Katholieke Universiteit Leuven
  Neuro-Symbolic = Neural + Logical + Probabilistic
- Nico Roos, Maastricht University
  We aren't doing AI research
- Yingqian Zhang, Eindhoven University of Technology
  AI for industrial decision-making

Authors were invited to submit papers on all aspects of Artificial Intelligence. This year we have received 83 submissions in total. Of the 41 submitted Type A regular papers, both short and long, 24 (59%) were accepted for presentation. All 19 submitted Type B compressed contributions were accepted for presentation. From the Type C demonstrations, 2 out of 3 were accepted. Of the submitted 20 Type D thesis abstracts, 17 were accepted for presentation. Together there are 38 accepted contributions from Type B, C and D. The selection was made based on a single-blind peer review process. Each submission was assigned to three members of the program committee, and their expert reviews were the basis for our decisions. We would like to thank all program committee members (listed on the previous pages) for their time and effort to help us with this task.

All accepted submissions appear in these electronic proceedings, and are made available on the conference web site during the conference. The 12 best accepted regular papers are invited to the postproceedings, to be published in the Springer CCIS series after the conference.

We are grateful to our sponsors for their generous support of the conference:

- SIKS: Netherlands research school for Information and Knowledge Systems
- SNN Adaptive Intelligence: Dutch Foundation for Neural Networks
- BNVKI: Benelux Association for Artificial Intelligence
- SKBS: Stichting Knowledge Based Systems
- ZyLAB
- LIACS: Leiden Institute of Advanced Computer Science

Finally, we would like to thank all who contributed to the success of BNAIC/BeneLearn 2020.

Lu Cao, Walter Kosters and Jefrey Lijffijt
Program Chairs

Contents

Regular papers

Compressed contributions

Regular papers

Evaluating the Robustness of Question-Answering Models to Paraphrased Questions

Paulo Alting von Geusau[0000−0002−3189−4380] and
Peter Bloem[0000−0002−0189−5817]

Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, Netherlands
p.geusau@gmail.com
vu@peterbloem.nl

Abstract. Understanding questions expressed in natural language is a fundamental challenge studied under different applications such as question answering (QA). We explore whether recent state-of-the-art models are capable of recognising two paraphrased questions using unsupervised learning. Firstly, we test QA models' performance on an existing paraphrased dataset (Dev-Para). Secondly, we create a new annotated paraphrased evaluation set (Para-SQuAD) containing multiple paraphrased question pairs from the SQuAD dataset. We describe qualitative investigations on these models and how they present paraphrased questions in continuous space. The results demonstrate that the paraphrased dataset confuses the QA models and leads to a decrease in their performance. Visualizing the sentence embeddings of Para-SQuAD by the QA models suggests that all models, except BERT, struggle to recognise paraphrased questions effectively.

Keywords: natural language · transformers · question answering · embeddings.

1 Introduction

Question answering (QA) is a challenging research topic. Small variations in semantically similar questions may confuse the QA models and result in giving different answers. For example, the questions "Who founded IBM?" and "Who created the company IBM?" should be recognised as having the same meaning by a QA model. QA models need to understand the meaning behind the words and their relationships. Those words can be ambiguous, implicit, and highly contextual.

The motivation for writing this paper springs from the observation that QA models can provide a wrong answer to a question that is phrased slightly different compared to a previous question. Despite the questions being semantically similar. This sensitivity to question paraphrases needs to be improved to provide more robust QA models. Modern QA models need to recognise paraphrases effectively and provide the same answers to paraphrased questions.

Despite the release of high-quality QA datasets, test sets are typically a random subset of the whole dataset, following the same distribution as the development and training sets. We need datasets to test the QA models' ability to recognise paraphrased questions and analyse their performance. Therefore, we use two datasets, based on SQuAD

(Rajpurkar et al., 2016), to conduct two separate experiments on BERT (Devlin et al., 2018), GPT-2 (Radford et al., 2019) and XLNet (Zhilin Yang et al., 2019).

The first dataset we use is an existing paraphrased test set (Dev-Para). Dev-Para is publicly available, and we use it to evaluate the models' over-sensitivity to paraphrased questions.[1] Dev-Para is created from SQuAD development questions and consists of newly generated paraphrases. Dev-Para evaluates the models' performance on unseen test data to gain a better indication of their generalisation ability. We hypothesise that adding new paraphrases to the test set will result in the models suffering a drop in performance. This paper will search for properties that the models learn in an unsupervised way, as a side effect of the original data, setup, and training objective.

In addition, we introduce a new paraphrased evaluation set (Para-SQuAD) to test the QA models' ability in recognising the semantics of a question in an unsupervised manner. Para-SQuAD is a subset of the SQuAD development set, whereas Dev-Para is much larger and consists of newly added paraphrases. Para-SQuAD consists of question pairs that are semantically similar but have a different syntactic structure. The question pairs are manually annotated and picked from the SQuAD development set. We analyse all sentence embeddings of Para-SQuAD in an embedding space with the help of t-SNE visualisation. For each model, we calculate the average cosine similarity of all question pairs to gain an understanding of the semantic similarity between paraphrased questions.

The contributions of this paper are threefold:

1. We test the QA models' performance on an existing paraphrased test set (Dev-Para) to evaluate their robustness to question paraphrases.
2. We create a new paraphrased evaluation set (Para-SQuAD) that consists of question pairs from the original SQuAD development set, the question pairs are semantically similar but have a different syntactic structure.
3. We create and visualize useful sentence embeddings of Para-SQuAD by the QA models, and calculate the average cosine similarity between the sentence embeddings for each QA model.

2 Methodology

In this section, we describe the models and sentence embeddings used, and we introduce our method to create Para-SQuAD.

2.1 BERT, GPT-2 and XLNet

We use QA models that are based on the transformer architecture from Vaswani et al. (2017). The models have been pre-trained on enormous corpora of unlabelled text, including Books Corpus and Wikipedia, and only require task-specific fine-tuning. The first model we use is Google's BERT. BERT is bidirectional because its self-attention

[1] https://github.com/nusnlp/paraphrasing-squad

layer performs self-attention in both directions; each token in the sentence has self-attention with all other tokens in the sentence. The model learns information from both the left and right sides during the training phase. BERT's input is a sequence of provided tokens, and the output is a sequence of generated vectors. These output vectors are referred to as 'context embeddings' since they contain information about the context of the tokens. BERT uses a stack of transformer encoder blocks and has two self-supervised training objectives: masked language modelling and next-sentence prediction.

The second model used in this paper is OpenAI's GPT-2. GPT-2 is also a transformer model and has a similar architecture to BERT; however, it only handles context on the left and uses masked self-attention. GPT-2 is built using transformer decoder blocks and was trained to predict the next word. The model is auto-regressive, just like Google's XLNet.

XLNet, the third model used in this paper has an alternative technique that brings back the merits of auto-regression while still incorporating the context on both sides. XLNet uses the Transformer-XL as its base architecture. The Transformer-XL extends the transformer architecture by adding recurrence at a segment level. XLNet already achieves impressive results for numerous supervised tasks; however, it is unknown if the model generates useful embeddings for unsupervised tasks. We explore this question further in this paper.

We use the small GPT-2, BERT-Base, and XLNet-Base, all consisting of 12 layers. The larger versions of BERT and XLNet have 24 layers; the larger version of GPT-2 has 36 layers.

2.2 Embeddings

Classic word embeddings are static and word-level; this means that each word receives exactly one pre-computed embedding. Embedding is a method that produces continuous vectors for given discrete variables. Word embeddings have demonstrated to improve various NLP tasks, such as question answering (J. Howard and S. Ruder., 2018). These traditional word embedding methods have several limitations in modelling the contextual awareness effectively. Firstly, they cannot handle polysemy. Secondly, they are unable to grasp a real understanding of a word based on its surrounding context.

Advances in unsupervised pre-training techniques, together with large amounts of data, have improved contextual awareness of models such as BERT, GPT-2, and XL-Net. Contextually aware embeddings are embeddings that not only contain information about the represented word, but also information about the surrounding words. The state-of-the-art transformer models create embeddings that depend on the surrounding context instead of an embedding for a single word.

Sentence embeddings are different from word embeddings in that they provide embeddings for the entire sentence. We aim to extract the numerical representation of a question to encapsulate its meaning. Semantically meaningful means that semantically similar sentences are clustered with each other in vector space.

The network structures of the transformer models compute no independent sentence embeddings. Therefore, we modify and adapt the transformer networks to obtain sentence embeddings that are semantically meaningful and used for visualization. We use

The Broncos took an early lead in Super Bowl 50 and never trailed. Newton was limited by Denver's defense, which sacked him seven times and forced him into three turnovers, including a fumble which they recovered for a touchdown. Denver linebacker Von Miller was named Super Bowl MVP, recording five solo tackles, 2½ sacks, and two forced fumbles.
Who was the Super Bowl 50 MVP?
Ground Truth Answers: Von Miller, Miller

Fig. 1. Example of SQuAD 1.1 development set with context, question, and answers.

QA models that are deep unsupervised language representations. All QA models are pre-trained with unlabelled data.

Feeding individual sentences to the models will result in fixed-size sentence embeddings. A conventional approach to retrieve a fixed size sentence embedding is to average the output layer, also called mean pooling. Another common approach for models like BERT and XLNet is to use the first token (the [CLS] token). In this paper, we use the mean pooling technique to retrieve the fixed-size sentence embeddings.

2.3 SQuAD

To create Para-SQuAD, we use the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016), which consists of over 100.000 natural question and answer sets retrieved from over 500 Wikipedia articles by crowd-workers. The SQuAD dataset is widely used as a popular benchmark for QA models. The QA models take a question and context as input to predict the correct answer. The two metrics used for evaluation are the exact match (EM) and the F1 score. The SQuAD dataset is a closed dataset; this means that the answer to a question exists in the context. Figure 1 illustrates an example from the SQuAD development set.

SQuAD treats the task of question answering as a reading comprehension task where the question refers to a Wikipedia paragraph. The answer to a question has to be a span of the presented context; therefore, the starting token and ending token of the substring is calculated.

2.4 Para-SQuAD

To evaluate the robustness of the models on recognising paraphrased questions, we create a new dataset called Para-SQuAD, using the SQuAD 1.1 development set. The SQuAD development set uses at least two additional answers for each question to make the evaluation more reliable. The human performance score on the SQuAD development set is 80.3% for the exact match, and 90.5% for F1.[2]

The first author manually analysed all the questions inside the SQuAD development set to acquire all paraphrased question pairs used in Para-SQuAD. Humans have

[2] https://rajpurkar.github.io/SQuAD-explorer/

a consistent intuition for "good" paraphrases in general (Liu et al., 2010). To be specific, we consider questions as paraphrases if they yield the same answer and have the same intention. The main criteria for well-written paraphrases are fluency and lexical dissimilarity. Moreover, word substitution is sufficient to count as a paraphrase.

Questions in the SQuAD development set relate to specific Wikipedia paragraphs and are grouped together. We manually select paraphrased question pairs that already exist in the SQuAD development set without creating new questions. This method ensures that Para-SQuAD is a typical subset of the SQuAD development set without inducing dataset bias. Moreover, the data distribution and dataset bias in Para-SQuAD and the SQuAD development set remains identical. Para-SQuAD consists of 700 questions, 350 paraphrased question pairs, and 12 different topic categories.

After paraphrase collection, we performed post-processing to check for any mistakes. The paraphrased questions are checked on English fluency using context-free grammar concepts.[3] We used spaCy[4] to conduct a sanity check after manually collecting all paraphrased questions. SpaCy provides paraphrase similarity scores of the question pairs. SpaCy is an industrial-strength natural language processing tool and receives sentence similarity scores by using word embedding vectors.

Using Para-SQuAD for visualisation has a significant advantage compared to using Dev-Para. Namely, the data distribution of Dev-Para changes after the addition of new sentences. On the contrary, the data distribution of Para-SQuAD remains the same because we do not add new sentences; we only annotate the existing paraphrases in the SQuAD development set.

2.5 Para-SQuAD Sentence Embeddings

We present a proof-of-concept visualization of the models' capability to represent semantically similar sentences closely in vector space. Previous research by Coenen et al. (2019) reveals that much of the semantic information, of BERT and related transformer models, is visible and encoded in a low-dimensional space. Therefore, we map all the paraphrased questions from Para-SQuAD to a sentence embedding space for every pre-trained model. Distance in the vector space can be interpreted roughly as sentence similarity according to the model in question.

We calculate the fixed-length vectors for each question using the Flair framework,[5] with mean pooling, to receive the final token representation. Mean pooling uses the average of all word embeddings to obtain an embedding for the whole sentence.

All transformer models produce 768-dimensional vectors for every question, and t-SNE (Laurens van der Maaten and Geoffrey Hinton, 2008) is applied to transform the high-dimensional space to a low-dimensional space in a local and non-linear way. The dimensionality is first reduced to 50 using Principal Component Analysis (PCA) (Karl Pearson, 1901) to ensure scalability, before feeding into t-SNE.

We use a perplexity of 50 for all models, after tuning the 'perplexity' parameter, to capture the clusters. Perplexity deals with the balance between global and local aspects

[3] https://www.nltk.org/
[4] https://spacy.io/
[5] https://github.com/flairNLP/flair

of the data. We tested diverse perplexity values to ensure robustness. We also explore the traditional word-based model GloVe (Pennington et al., 2014) and compare its sentence embeddings to the state-of-the-art transformer models. We investigate if GloVe captures the nuances of the meaning of sentences more effectively as compared to the transformer models.

3 Results

In this section, we evaluate the two experiments. The first experiment measures the performance of the QA models on Dev-Para. The second experiment visualises the sentence embeddings of Para-SQuAD for each QA model.

3.1 Experiments on QA Models

We conduct experiments on three pre-trained models: BERT, GPT-2, and XLNet. The training code of the models is based on the Hugging Face implementation, which is publicly available.[6] In addition to using the pre-trained models directly, we fine-tuned the models on the SQuAD 1.1 training set. We first measure the performance of the pre-trained models on Dev-Para. Secondly, we use the three pre-trained models and GloVe to visualize the sentence embeddings of Para-SQuAD in an embeddings space. Both experiments are performed in an unsupervised manner.

3.2 Dev-Para Performance

We illustrate the performance of all three pre-trained QA models on Dev-Para. Dev-Para consists of the original set and the paraphrased set. The original set contains more than 1.000 questions from the SQuAD development set; the paraphrased set contains between 2 and 3 generated paraphrased questions for each question from the original set (Wee Chung Gan and Hwee Tou Ng, 2019).

The QA models' performance on Dev-Para is presented in Table 1. Although the original set of Dev-Para is semantically similar to the paraphrased set, we see a drop in performance of all three models. Especially GPT-2 and XLNet are suffering a significant drop in performance.

Model	**EM Score**		**F1 Score**	
	Original	Paraphrased	Original	Paraphrased
BERT	82.2	78.7	89.2	86.2
GPT-2	71.6	62.9	80.4	72.7
XLNet	89.4	82.6	93.7	85.3

Table 1. Performance of the QA models on Dev-Para.

[6] https://github.com/huggingface/transformers

The drop in performance is unexpected since the meaning of the questions did not change between the original set and the paraphrased set of Dev-Para. One possible explanation is that the model is exploiting surface details in the original set that are not reproduced by the protocol used to create Dev-Para. If true, this demonstrates a lack of robustness in the models. Moreover, the added questions could be more complicated, therefore allowing for more variability in the syntactic structure, and those questions for which there are paraphrases are variants of more frequent questions.

3.3 Visualization Para-SQuAD

For the following continuous space exploration of Para-SQuAD, we focus on the BERT, GPT-2, XLNet, and GloVe sentence embeddings. Each point in the space represents a question; the 12 colours in Figure 2-5 represent the different categories. The lines in Figure 6-9 illustrate the distance between the paraphrased question pairs. Figure 6-9 all consist of the same amount of lines; however, some lines are difficult to see if both paraphrased question pairs appear close to each other in the embedding space. Paraphrased question pairs that represent the same location in the embedding space appear as a single dot without lines. As a result, it seems that Figure 6 contains fewer lines compared to figure 8, which is a false assumption.

Using visualization as a key evaluation method has important risks to consider. Relative sizes of clusters cannot be seen in a t-SNE plot as dense clusters are expanded, and spare clusters are shrunk. Furthermore, distances between the separated clusters in the t-SNE plot may mean nothing. Clumps of points in the t-SNE plot might be noise coming from small perplexity values.

The visualization of Para-SQuAD consists of all 350 paraphrased question pairs. We argue that the semantics of the questions occupy different locations in continuous space. This hypothesis is tested qualitatively by manually analysing the t-SNE plots of the models. As a sanity check, all sample points in the plots have been manually analysed with the corresponding sentences to check for mistakes (e.g., wrong colour or pairs).

We explore sample points within clusters to gain relevant insights. If two sample points are far from each other in the plot, it does not necessarily imply that they are far from each other in the embedding space. However, the number of long distances between paraphrased question pairs, coming from different clusters, can reveal information on the robustness of the models to recognise paraphrased question pairs and their semantics.

Figure 2 illustrates that BERT creates clear and distinct clusters for every category; we only observe a few errors. Most paraphrased questions are within the same cluster and close to each other (Figure 6). Therefore, it seems that BERT can capture similar semantic sentences effectively.

GPT-2 has trouble clustering the different categories (Figure 3). After manually analysing the sentences in the different clusters, it seems that GPT-2 offers special attention to the first tokens in the sentence. The paraphrased question pairs are close to each other in vector space if they start with the same token. The starting token is often the ‘question word’ in Para-SQuAD. It seems that GPT-2 organises questions by their structure instead of their semantics.

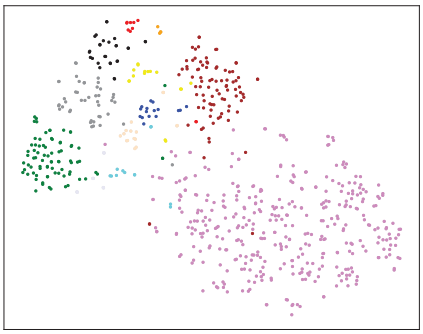


Fig. 2. BERT sentence embeddings.

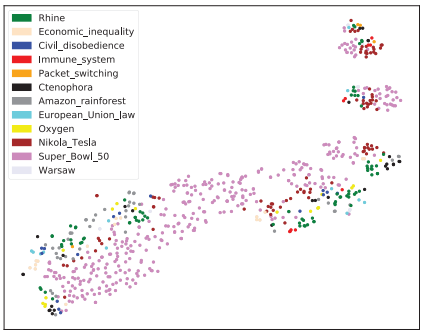


Fig. 3. GPT-2 sentence embeddings.

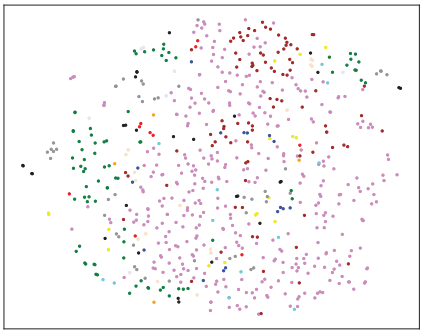


Fig. 4. XLNet sentence embeddings.

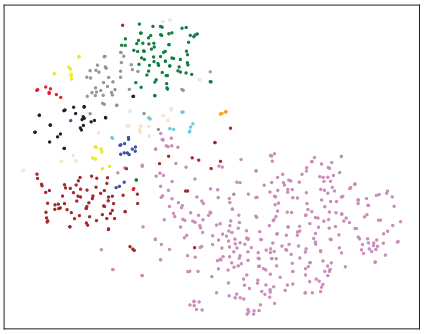


Fig. 5. GloVe sentence embeddings.

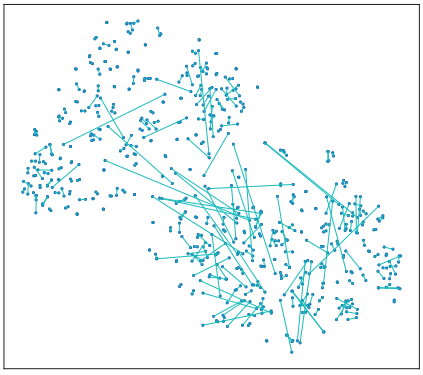


Fig. 6. BERT sentence embeddings.

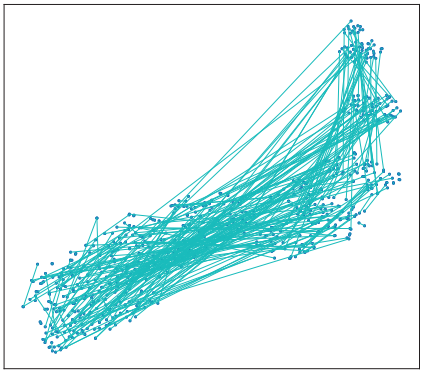


Fig. 7. GPT-2 sentence embeddings.

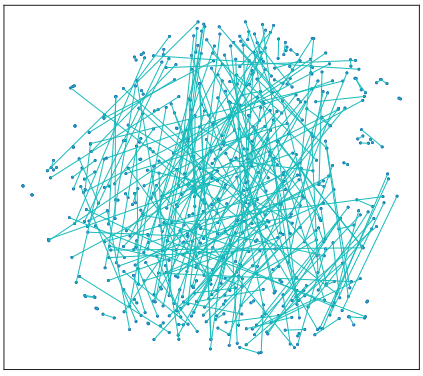


Fig. 8. XLNet sentence embeddings.

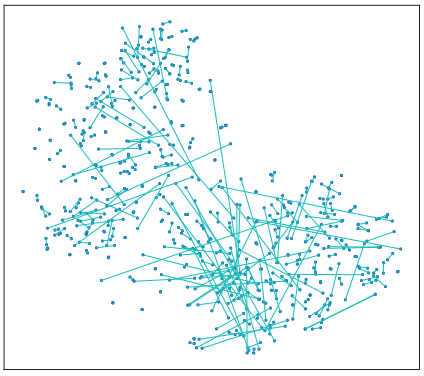


Fig. 9. GloVe sentence embeddings.

XLNet forms one large cluster, with smaller clusters within (Figure 4). However, these clusters are not that clear when compared to BERT. The different categories are all spread out, and no apparent clusters are formed.

Figure 5 suggests that GloVe clusters the different categories more effectively than GPT-2 and XLNet, despite using static embeddings. This finding is interesting, since contextualised embedding are thought to be superior compared to traditional static embeddings. At the same time, the paraphrased questions that appear close to each other in Figure 9 have similar words in the sentence and can be considered as easy paraphrases. GloVe is unable to recognise more complex paraphrases, which can be explained by the model's architecture and not providing contextualised embeddings.

Model	Average Cosine Similarity
BERT	0.875
BERT (fine-tuned)	0.939
GPT-2	0.987
XLNet	0.981

Table 2. Average cosine similarity of the QA models.

In this paper, we use the cosine similarity to measure the closeness between paraphrased question pairs. For each model, we calculate the average cosine similarity for all the paraphrased question pairs in Para-SQuAD to see if the fine-tuned models perform better than the pre-trained models (Table 2). Calculating the average cosine similarity was only relevant for comparing the pre-trained BERT and the fine-tuned BERT. The cosine similarity of the fine-tuned BERT increased with 7.3%. The plots of the fine-tuned models reveal no interesting findings; therefore, we only illustrate the sentence embeddings of the basic pre-trained models.

The average cosine similarity of GPT-2, as illustrated in Table 2, is almost perfect. However, after further investigating the cosine similarity between all paraphrased question pairs, we notice that even two semantically dissimilar sentences have a high cosine similarity. Therefore, this high average reveals extreme anisotropy in the last layers of GPT-2; sentences occupying a tight space in the vector space. We also notice the same effect in XLNet. We can, therefore, suggest that GPT-2 and XLNet are the most context-specific models. This observation is in line with the work of Kawin Ethayarajh (2019).

4 Related Work

Recent research on deep language models and transformer architectures (Vaswani et al., 2017) has demonstrated that context embeddings in transformer models contain sufficient information to perform various NLP tasks with simple classifiers, such as question answering (Tenney et al., 2019; Peters et al., 2018). They suggest that these models produce valuable representations of both syntactic and semantic information.

Attention matrices can encode significant connections between words in a sentence, as illustrated with qualitative and visualization-based work by Jesse Vig (2019). Multiple tests to measure how effective word embeddings capture syntactic and semantic information is defined in the work of Mikolov et al. (2013). Furthermore, the recent work of Hewitt et al. (2019) analysed context embeddings for specific transformer models.

Sentence embeddings can be helpful in multiple ways, analogous to word embeddings. Common proposed methods are: InferSent (Conneau et al., 2017), Skip-Thought (Kiros et al., 2015) and Universal Sentence Encoder (USE) (Cer et al., 2018). Hill et al. (2016) prove that training sentence embeddings on a specific task, such as question answering, impact their quality significantly.

Conneau et al. (2018) presented probing tasks to evaluate sentence embeddings intrinsically. Evaluation of sentence embeddings happens most often in 'transfer learning' tasks, e.g., question type prediction tasks. The study measures to what degree linguistic features, like word order or sentence length, are accessible in a sentence embedding. This study was continued with SentEval (Alexis Conneau and Douwe Kiela, 2018), which serves as a toolkit to evaluate the quality of sentence embeddings. This quality is measured both intrinsically and extrinsically. SentEval proves that no sentence embedding technique is flawless across all tasks (Perone et al., 2018).

Recently, numerous QA datasets have been published (e.g., Rajpurkar et al., 2016; Rajpurkar et al., 2018). However, defining a suitable QA task and developing methodologies for annotation and evolution is still challenging (Kwiatkowski et al., 2019). Key issues include the metrics used for evaluation and the methods and sources used to obtain the questions.

Our analysis focuses on three specific transformer models; however, there are numerous transformer models available. Other notable transformer models are XLM (Lample et al., 2019) and ELECTRA (Clark et al., 2020). Recent papers have focused on generalisability by evaluating different models on several datasets (Priyanka Sen and Amir Saffari, 2020), but not for paraphrasing specifically.

5 Conclusion

This paper presents an initial exploration of how QA models handle paraphrased questions. We used two different datasets and performed tests on each dataset. Firstly, we used an existing paraphrased test set (Dev-Para) to test the QA models' robustness to paraphrased questions. The results demonstrate that all three QA models drop in performance when exposed to more unseen paraphrased questions. The drop in performance could be explained by exposing the models to new paraphrased questions that deviate from the original SQuAD questions. The experiments underline the importance of improving QA models' robustness to question paraphrasing to generalise effectively. Moreover, increased robustness is necessary to increase the reliability and consistency of the QA models when tested on unseen questions in real-life world applications.

Secondly, we constructed a paraphrased evaluation set (Para-SQuAD) based on SQuAD to illustrate interesting insights into QA models handling paraphrased questions. The findings reveal that BERT creates the most promising and informative sentence embeddings and seems to capture semantic information effectively. The other

models, however, seem to fail in recognising paraphrased question pairs effectively and lack robustness.

5.1 Discussion

The models' drop in performance on Dev-Para is unexpected. We hypothesise that the original SQuAD training set does not consist of enough diverse question paraphrases. This lack of variation leads to the QA models not learning to answer different questions, that have the same intention and meaning, correctly. The QA models fail to recognise some questions that convey the same meaning using different wording. Exposing the QA models to more different question phrases would be a logical step to improve the QA models' robustness to question paraphrasing.

Generating paraphrases and recognizing paraphrases are still critical challenges across multiple NLP tasks, including question answering and semantic parsing. A relatively robust and diverse source for generating paraphrases is through neural machine translation. We can make larger datasets consisting of paraphrased questions with the help of machine translation: the question is translated into a foreign language and then back-translated into English. This back-translation approach achieved remarkable results in diversity compared to paraphrases created by human experts (Federmann et al., 2019).

5.2 Limitations

One limitation of the performed experiments is the small size of Para-SQuAD. Increasing Para-SQuAD with data augmentation could be achieved with the use of neural machine translation to generate more paraphrases. Increasing the size of Para-SQuAD would lead to more reliable results, but we would lose the advantage of keeping the data distribution intact.

Another downside is the simplicity of Para-SQuAD. The paraphrases used are relatively simple and basic. Therefore, models achieving excellent results on the set does not guarantee their robustness to question paraphrases.

In general, there is no inter-annotator agreement measure to ensure consistent annotations because we only have one annotator. However, we consider this justified due to the simple task of selecting paraphrased question pairs in the SQuAD development set.

Using visualization as the primary evaluation method has its risks. A common pitfall includes pareidolia; to see structures and patterns that we would like to see. As an example, we can see that BERT forms clear clusters that are known to us; however, other models could form divergent cluster structures to represent patterns. We could, therefore, easily overlook those cluster structures that are unfamiliar to us. Furthermore, clusters can disappear in the t-SNE transformation.

Lastly, with the performed method, it is hard to distinguish whether BERT recognizes the actual semantics of the questions or merely the Wikipedia extracts. Further research is needed to investigate this distinction.

Acknowledgment

We thank the three anonymous reviewers for their constructive comments, and Michael Cochez for his feedback and helpful notes on the manuscript.

References

1. Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder. *arXiv preprint arXiv:1803.11175.*
2. Kevin Clark, Minh-Thang Luong, Quoc V. Le, Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. *arXiv preprint arXiv:2003.10555.*
3. Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda Viégas, Martin Wattenberg. 2019. Visualizing and Measuring the Geometry of BERT. *arXiv preprint arXiv:1906.02715.*
4. Alexis Conneau and Douwe Kiela. 2018. SentEval: An Evaluation Toolkit for Universal Sentence Representations. *arXiv preprint arXiv:1803.05449.*
5. Alexis Conneau, Douwe Kiela, Holger Schwenk, Loıc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing,* pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.
6. Alexis Conneau, German Kruszewski, Guillaume Lample, Loic Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *CoRR*, abs/1805.01070.
7. Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805.*
8. Kawin Ethayarajh. 2019. How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. *arXiv preprint arXiv:1909.00512.*
9. Christian Federmann, Oussama Elachqar, Chris Quirk. 2019. Multilingual Whispers: Generating Paraphrases with Translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019).* Association for Computational Linguistics.
10. Wee Chung Gan and Hwee Tou Ng. 2019. Improving the Robustness of Question Answering Systems to Question Paraphrasing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.
11. John Hewitt and Christopher D Manning. 2019. A Structural Probe for Finding Syntax in Word Representations. *Association for Computational Linguistics.*
12. Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning Distributed Representations of Sentences from Unlabelled Data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* pages 1367– 1377, San Diego, California. Association for Computational Linguistics.
13. J. Howard and S. Ruder. 2018. Fine-tuned Language Models for Text Classification. *CoRR*, abs/1801.06146.
14. Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought Vectors. In *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc.

15. Tom Kwiatkowski, Jennimaria Palomaki, Olivia Rhinehart, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. In *Transactions of the Association of Computational Linguistics.*
16. Guillaume Lample and Alexis Conneau. 2019. Cross-lingual Language Model Pretraining. *arXiv preprint arXiv:1901.07291.*
17. Chang Liu, Daniel Dahlmeier, and Hwee Tou Ng 2010. PEM: A Paraphrase Evaluation Metric Exploiting Parallel Texts. *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 923-932.
18. Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research,* 9:2579–2605.
19. Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781.*
20. Karl Pearson F.R.S. 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science,* Volume 2.
21. J. Pennington, R. Socher, and C. D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
22. Christian S. Perone, Roberto Silveira, and Thomas S. Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *CoRR*, abs/1806.06259.
23. Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Ken- ton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. *arXiv preprint arXiv:1802.05365.*
24. Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 784–789.
25. Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,* pages 2383–2392.
26. Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI Blog.*
27. Priyanka Sen, Amir Saffari. 2020. What do Models Learn from Question Answering Datasets? *arXiv preprint arXiv:2004.03490.*
28. Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, Ellie Pavlick. 2019. What do you learn from context? Probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316.*
29. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. 2017. Attention Is All You Need. *arXiv preprint arXiv:1706.03762.*
30. Jesse Vig. 2019. Visualizing Attention in Transformer-Based Language Representation Models. *arXiv preprint arXiv:1904.02679.*
31. Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237.*

FlipOut: Uncovering Redundant Weights via Sign Flipping*

Andrei C. Apostol[1,2,3], Maarten C. Stol[2], and Patrick Forré[1]

[1] Informatics Institute, University of Amsterdam, The Netherlands
[2] BrainCreators B.V., Amsterdam, The Netherlands
[3] apostol.andrei@braincreators.com

Abstract. We propose a novel pruning method which uses the oscillations around 0 (i.e. sign flips) that a weight has undergone during training in order to determine its saliency. Our method can perform pruning before the network has converged, requires little tuning effort due to having good default values for its hyperparameters, and can directly target the level of sparsity desired by the user. Our experiments, performed on a variety of object classification architectures, show that it is competitive with existing methods and achieves state-of-the-art performance for levels of sparsity of 99.6% and above for 2 out of 3 of the architectures tested. For reproducibility, we release our code publicly at https://github.com/AndreiXYZ/flipout.

Keywords: deep learning · network pruning · computer vision.

1 Introduction

The success of deep learning is motivated by competitive results on a wide range of tasks ([3,9,24]). However, well-performing neural networks often come with the drawback of a large number of parameters, which increases the computational and memory requirements for training and inference. This poses a challenge for deployment on embedded devices, which are often resource-constrained, as well as for use in time sensitive applications, such as autonomous driving or crowd monitoring. Moreover, costs and carbon dioxide emissions associated with training these large networks have reached alarming rates ([21]). To this end, pruning has been proven as an effective way of making neural networks run more efficiently ([5,6,13,15,18]).

Early works ([6,13]) have focused on using the second-order derivative to detect which weights to remove with minimal impact on performance. However, these methods either require strong assumptions about the properties of the Hessian, which are typically violated in practice, or are intractable to run on modern neural networks due to the computations involved.

One could instead prune the weights whose optimum lies at or close to 0 anyway. Building on this idea, the authors of [5] propose training a network until

* Supported by BrainCreators B.V.

convergence, pruning the weights whose magnitudes are below a set threshold, and allowing the network to re-train, a process which can be repeated iteratively. This method is improved on in [4], whereby the authors additionally reset the remaining weights to their values at initialization after a pruning step. Yet, these methods require re-training the network until convergence multiple times, which can be a time consuming process.

Recent alternatives either rely on methods typically used for regularization ([17,18,26]) or introduce a learnable threshold, below which all weights are pruned ([16]). All these methods, however, require extensive hyperparameter tuning in order to obtain a favorable accuracy-sparsity trade-off. Moreover, the final sparsity of the resulting network cannot be predicted given a particular choice of these hyperparameters. These two issues often translate into the fact that the practitioner has to run these methods multiple times when applying them to novel tasks.

To summarize, we have seen that the pruning methods presented so far suffer from one or more of the following problems: (1) computational intractability, (2) having to train the network to convergence multiple times, (3) requiring extensive hyperparameter tuning for optimal performance and (4) inability to target a specific final sparsity.

We note that by using a heuristic in order to determine during training whether a weight has a locally optimal value of low magnitude, pruning can be performed before the network reaches convergence, unlike the method proposed by the authors of [5]. We propose one such heuristic, coined *the aim test*, which determines whether a value represents a local optimum for a weight by monitoring the number of times that weight oscillates around it during training, while also taking into account the distance between the two. We then show that this can be applied to network pruning by applying this test at the value of 0 for all weights simultaneously, and framing it as a saliency criterion. By design, our method is tractable, allows the user to select a specific level of sparsity and can be applied during training.

Our experiments, conducted on a variety of object classification architectures, indicate that it is competitive with respect to relevant pruning methods from literature, and can outperform them for sparsity levels of 99.6% and above. Moreover, we empirically show that our method has default hyperparameter settings which consistently generate near optimal results, easing the burden of tuning.

2 Method

2.1 Motivation

Mini-batch stochastic gradient descent ([2]) is the most commonly used optimization method in machine learning. Given a mini-batch of B randomly sampled training examples consisting of pairs of features and labels $\{(x_b, y_b)\}_{b=1}^{B}$, a neural network parameterised by a weight vector $\boldsymbol{\theta}$, a loss objective $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})$ and a

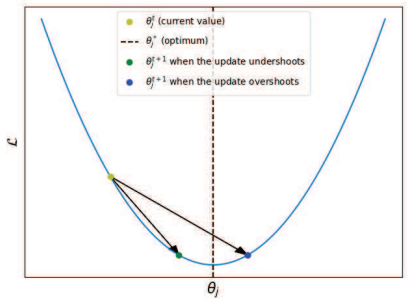


Fig. 1: Over- and under-shooting illustrated. The vertical line splits the x-axis into two regions relative to the (locally-)optimal value θ_j^*. Overshooting corresponds to when a weight gets updated such that its new value lies in the opposite region (blue dot), while undershooting occurs when the updated value is closer to the optimal value, but stays in the same region (green dot).

learning rate η, the update rule of stochastic gradient descent is as follows:

$$\boldsymbol{g}^t = \frac{1}{B}\sum_{b=1}^{B} \nabla_{\boldsymbol{\theta}^t} \mathcal{L}(\boldsymbol{\theta}^t, x_b, y_b)$$
$$\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t - \eta \boldsymbol{g}^t$$

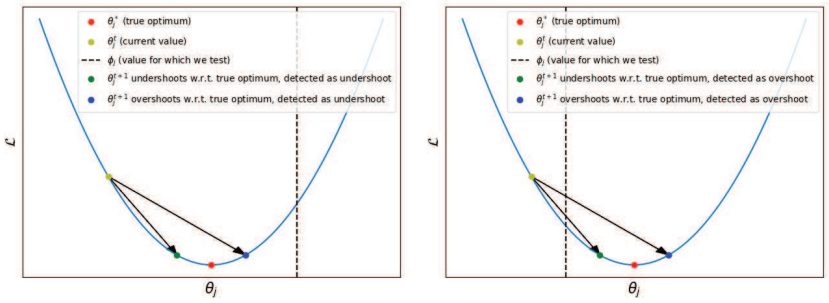
Given a weight θ_j^t, one could consider its possible values as being split into two regions, with a locally optimal value θ_j^* as the separation point. Depending on the value of the gradient and the learning rate, the updated weight θ_j^{t+1} will lie in one of the two regions. That is, it will either get closer to its optimal value while remaining in the same region as before or it will be updated past it and land in the opposite region. We term these two phenomena under- and over-shooting, and provide an illustration in Fig. 1. Mathematically, they correspond to $\eta|g_j^t| < |\theta_j^t - \theta_j^*|$ and $\eta|g_j^t| > |\theta_j^t - \theta_j^*|$, respectively.

With the behavior of under- and over-shooting, one could construct a heuristic-based test in order to evaluate whether a weight has a local optimum at a specific point without needing the network to have reached convergence:

1. For a weight θ_j, a value of ϕ_j is chosen for which the test is conducted
2. Train the model regularly and record the occurrence of under- and over-shooting around ϕ_j after each step of SGD
3. If the number of such occurrences exceeds a threshold κ, conclude that θ_j has a local optimum at ϕ_j (i.e. $\theta_j^* = \phi_j$)

We coin this method *the aim test*.

Previous works have demonstrated that neural networks can tolerate high levels of sparsity with negligible deterioration in performance ([4,5,16,18]). It is then reasonable to assume that for a large number of weights, there exist local



(a) Deceitful observations of under-shooting.(b) Deceitful observations of over-shooting.

Fig. 2: In the plots above, the dotted vertical line represents the value at which the aim test is conducted (i.e. a value we would like to determine as a local optimum or not), while the red dot represents the value of a true local optimum. When testing for a value which is not a locally optimal value $\phi_j \neq \theta_j^*$, over- or under-shooting around ϕ_j can be merely a side-effect of that weight getting updated towards its true optimum θ_j^*. These observations would then contribute towards the aim test returning a false positive outcome, i.e. $\phi_j = \theta_j^*$. Whether we observe an over-shoot or an under-shoot in this case depends on the relationship between ϕ_j and θ_j^*. In (a), we have $\phi_j > \theta_j^*$, where if the hypothesised and true optimum are sufficiently far apart, we observe an under-shoot. Conversely, in (b), we have $\phi_j < \theta_j^*$ and observe over-shooting.

optima at exactly 0, i.e. $\theta_j^* = 0$. One could then use the aim test to detect these weights and prune them. Importantly, when using the aim test for $\phi_j = 0$, the two regions around the tested value are the set of negative and positive real numbers, respectively. Checking for over-shooting then becomes equivalent to testing whether the sign of θ_j has changed after a step of SGD, while under-shooting can be detected when a weight has been updated to a smaller absolute value and retained its sign, i.e. $(|\theta_j^{t+1}| < |\theta_j^t|) \wedge (\text{sgn}(\theta_j^t) = \text{sgn}(\theta_j^{t+1}))$.

However, under-shooting can be problematic; for instance, a weight could be updated to a lower magnitude, while at the same time being far from 0. This can happen when a weight is approaching a non-zero local optimum, an occurrence which should not contribute towards a positive outcome of the aim test. By positive outcome, we refer to determining that $\phi_j = 0$ is indeed a local optimum of θ_j. A similar problem can occur for over-shooting, where a weight receives a large update that causes it to change its sign but not lie in the vicinity of 0. These scenarios, which we will refer to as *deceitful shots* going forward, are illustrated in the general case, where ϕ_j can take any value, in Fig. 2a and Fig. 2b. Following, we make two observations which help circumvent this problem.

Firstly, one could reduce the impact of deceitful shots by also taking into account the distance of the weight to the hypothesised local optimum, i.e. $|\theta_j - \phi_j|$, when conducting the aim test. In other words, the number of occurrences of under-

and over-shooting should be weighed inversely proportional to this quantity, even if they would otherwise exceed κ.

Our second observation is that by ignoring updates which are not in the vicinity of ϕ_j, the number of deceitful shots are reduced. In doing so, one could also simplify the aim test; with a sufficiently large perturbation to θ_j, an update that might otherwise cause under-shooting can be made to cause over-shooting. Adding a perturbation of $\pm\epsilon$ is, in effect, inducing a boundary around the tested value, $[\phi_j - \epsilon, \phi_j + \epsilon]$; all weights that get updated such that they fall into that boundary will be said to over-shoot around ϕ_j. With this framework, checking for over-shooting is sufficient; updates that under-shoot and are within ϵ of the tested value are made to over-shoot (Fig. 3a) and updates which under-shoot but are not in the vicinity of ϕ_j, i.e. a deceitful shot, are now not recorded at all (Fig. 3b). This can also be seen as restricting the aim test to only operate within a vicinity around ϕ_j.

2.2 FlipOut: applying the aim test for pruning

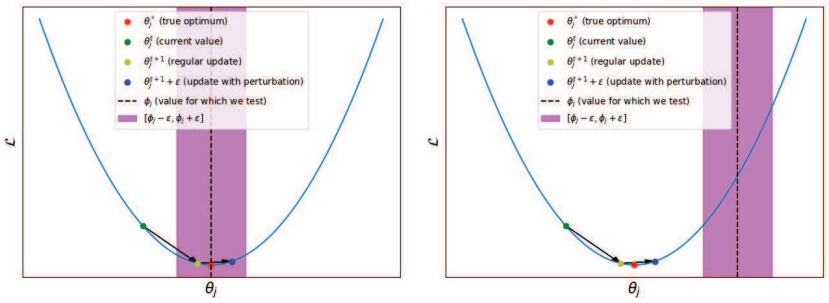
Determining which weights to prune Pruning weights that have local optima at or around 0 can obtain a high level of sparsity with minimal degradation in accuracy. The authors of [5] use the magnitude of the weights once the network is converged as a criterion; that is, the weights with the lowest absolute value (i.e. closest to 0) get pruned. The aim test can be used to detect whether a point represents a local optimum for a weight and can be applied before the network reaches convergence, during training. For pruning, one could then apply the aim test simultaneously for all weights with $\boldsymbol{\phi} = \mathbf{0}$. We propose framing this as a saliency score; at time step t, the saliency τ_j^t of a weight θ_j^t is:

$$\tau_j^t = \frac{|\theta_j^t|^p}{\text{flips}_j^t} \tag{1a}$$

$$\text{flips}_j^t = \sum_{i=0}^{t-1} [\text{sgn}(\theta_j^i) \neq \text{sgn}(\theta_j^{i+1})] \tag{1b}$$

With perturbation added into the weight vector, it is enough to check for over-shooting, which is equivalent to counting the number of sign flips a weight has undergone during the training process when $\phi_j = 0$ (Eq. 1b); a scheme for adding such perturbation is described in Section 2.2. In Equation 1a, the denominator $|\theta_j^t|^p$ represents the proximity of the weight to the hypothesised local optimum, $|\theta_j^t - \phi_j|^p$ (which is equivalent to the weight's magnitude since we have $\phi_j = 0$ for all weights). The hyperparameter p controls how much this quantity is weighted relative to the number of sign flips.

When determining the amount of parameters to be pruned, we adopt the strategy from [4], i.e. pruning a percentage of the remaining weights each time, which allows us to target an exact level of sparsity. Given m, the number of times pruning is performed, r the percentage of remaining weights which are removed at each pruning step, k the total number of training steps, $d_{\boldsymbol{\theta}}$ the dimensionality



(a) Under-shooting can become over-shooting by adding perturbation.

(b) Ignoring deceitful shots.

Fig. 3: (a) All weights that under-shoot but are within ϵ of ϕ_j will be made to over-shoot. (b) When testing at a value which is not a local optimum for θ_j, i.e. $\phi_j \neq \theta_j^*$ and adding a perturbation ϵ to θ_j, not taking under-shooting into account means that if the weight gets updated such that it does not lie in the boundary around ϕ_j induced by the perturbation, an event that would otherwise contribute to a false positive outcome for the aim test will not be recorded, so the likelihood of rejecting ϕ_j as an optimum increases.

of the weights and $||\cdot||_0$ the L_0-norm, the resulting sparsity s of the weight tensor after training the network is simply:

$$s = 1 - \frac{||\boldsymbol{\theta}^k||_0}{d_{\boldsymbol{\theta}}} = (1-r)^m \tag{2}$$

This final sparsity can then be determined by setting m and r appropriately.

Perturbation through gradient noise Adding gradient noise has been shown to be effective for optimization ([19,25]) in that it can help lower the training loss and reduce overfitting by encouraging an exploration in the parameter space, thus effectively acting as a regularizer. While the benefits of this method are helpful, our motivation for its usage stems from allowing the aim test to be performed in a simpler manner; weights that get updated closer to 0 will occasionally pass over the axis due to the injected noise, thus making checking for over-shooting sufficient. We scale the variance of the noise distribution by the L_2 norm of the parameters $\boldsymbol{\theta}$, normalize it by the number of weights and introduce a hyperparameter λ which scales the amount of noise added into the gradients. For a layer l and d_l its dimensionality, the gradient for the weights in

that layer used by SGD for updates will be:

$$\hat{\boldsymbol{g}}^{t,l} \leftarrow \boldsymbol{g}^{t,l} + \lambda \boldsymbol{\epsilon}^{t,l} \tag{3a}$$

$$\boldsymbol{\epsilon}^{t,l} \sim \mathcal{N}(0, \sigma_{t,l}^2) \tag{3b}$$

$$\sigma_{t,l}^2 = \frac{\|\boldsymbol{\theta}^{t,l}\|_2^2}{d_l} \tag{3c}$$

As training is performed, it is desirable to reduce the amount of added noise so that the network can successfully converge. Previous works use annealing schedules by decaying the variance of the Gaussian distribution proportional to the current time step. Under our proposed formulation, however, explicitly using an annealing schema is not necessary. By pruning weights, the term in the numerator in Eq. 3c decreases, while the denominator remains constant. This ensures that annealing will be induced automatically through the pruning process, and there is no need for manually constructing a schedule.

Pruning periodically throughout training according to the saliency score in Eq. 1a in conjunction with adding gradient noise into the weights forms the *FlipOut* pruning method.

3 Related work

3.1 Deep-R

In Deep-R ([1]), the authors split the weights of the neural network into two matrices, the connection parameter θ_k and a constant sign s_k with $s_k \in \{-1, +1\}$; the final weights of the network are then defined as $\boldsymbol{\theta} \odot \boldsymbol{s}$. The connections whose θ_k is negative are inactive; whenever a connection changes its sign, it is turned dormant and another randomly sampled connection is re-activated, ensuring the same sparsity level is maintained throughout training. Gaussian noise is also injected into the gradients during training.

Two similarities with our method can be observed here, namely the fact that the authors also use sign flipping as a signal for pruning a weight, and the addition of Gaussian noise. However, our methods differ in that we do not impose a set level of sparsity throughout training; instead, we use the number of sign flips of a weight in order to determine its saliency, while in Deep-R a single sign flip is required for a weight to be removed. Our method of injecting noise into the gradients also differs in that it does not explicitly encode an annealing scheme, allowing for the pruning process itself to reduce the noise throughout training. Finally, in Deep-R, the network is initialized with a specific level of sparsity which is maintained throughout training, while our method prunes gradually.

3.2 Magnitude and uncertainty pruning

The M&U pruning criterion is proposed in [11]. Given a weight θ_j, its uncertainty estimate $\tilde{\sigma}_{\theta_j}$ and a parameter λ controlling the trade-off between magnitude and

uncertainty, the M&U criterion will evaluate the saliency of the weight as:

$$\tau_j = \frac{|\theta_j|}{\lambda + \tilde{\sigma}_{\theta_j}}$$

Uncertainty is estimated as the standard deviation across the previous n values of that weight, via a process called pseudo-bootstraping. This criterion is a generalization of the Wald test, and is equivalent to it when $\lambda = 0$.

Our method is similar in that our saliency score also normalizes the weight's magnitude by a function of its past values. However, this method assumes asymptotic normality. While this is the case when using negative log-likelihood or an equivalent as the loss function, this property does not necessarily hold when using modified variants of the SGD estimator, such as Adam ([10]) or RMSprop ([22]). In contrast, FlipOut is not derived from the Wald test and does not make any assumptions about the weight distribution at convergence.

4 Experiments

4.1 General Setup

Baselines As baselines, we consider a slightly modified version of magnitude pruning ([5]) (Global magnitude), due to the similarity between its saliency criterion and that of our own method, SNIP ([14]) due to it being an easily applicable method which does not suffer from any of the issues that are commonly found in pruning methods (Section 1) and Hoyer-Square, as introduced in [26], for the state-of-the-art results that it has demonstrated. We also include random pruning (Random) as a control. For FlipOut, Global magnitude and Random, pruning is performed periodically throughout training. We compare these methods at five different compression ratios, chosen at regular log-intervals (Table 1); for Hoyer-square, the performance at those points is estimated by a sparsity-accuracy trade-off curve. Magnitude pruning, in its original formulation, performs pruning only once the network has reached convergence. However, employing this strategy can create a confounding variable: training time. Since we would like to compare all methods at equal training budgets, we have opted to simply perform pruning after a fixed number of epochs for these methods. Note that the training budget that we allocate allows all of the networks that we consider to reach convergence when trained without performing any pruning. We make an exception to this equal budget rule for Hoyer-Square, since it prunes after training and would otherwise not benefit from any SGD updates after sparsification. As such, we have performed an additional 150 epochs of fine-tuning without the regularizer, as per the original method, although we have observed negligible benefits to this. All baselines were modified to rank the weights globally when a pruning decision is made, as per the strategy from [4], in order to avoid creating bottleneck layers. The models that we test on are ResNet18 ([7]) and VGG19 ([20]) trained on the CIFAR-10 dataset ([12]), and DenseNet121 ([9]) trained on Imagenette ([8]).

Table 1: Compression ratios, resulting sparsity levels and prune frequencies used in the experiments, assuming 350 epochs of training and that 50% of the remaining weights are removed at each step.

Compression ratio ($\frac{d_{\boldsymbol{\theta}}}{\|\|\boldsymbol{\theta}\|\|_0}$)	Resulting sparsity ($1 - \frac{\|\|\boldsymbol{\theta}\|\|_0}{d_{\boldsymbol{\theta}}}$)	Epochs before pruning
2^2	75%	117
2^4	93.75%	70
2^6	98.43%	50
2^8	99.61%	39
2^{10}	99.9%	32

Hyperparameters The training parameters for all experiments are taken from [23]; specifically, we use a learning rate of 0.1, batch size of 128, 350 epochs of training and a weight decay penalty of $5e-4$. The learning rate is decayed by a factor of 10 at epochs 150 and 250. The networks are trained with the SGD optimizer with a momentum value of 0.9 ([2]). For the methods that perform iterative pruning (Global magnitude, Random, FlipOut), we remove 50% of the remaining weights at each pruning step, with the pruning frequencies chosen such that the compression ratios from Table 1 are achieved; we use the same pruning rates and frequencies across all three methods. SNIP accepts a single hyperparameter, namely the desired final sparsity, which we have chosen such that it matches the aforementioned compression ratios. For Hoyer-Square, which does not allow for a specific level of sparsity to be chosen and, instead, relies on parameter tuning, we generate a sparsity-accuracy trade-off curve by using 15 different values for the regularization term, ranging from $1e-7$ to $6e-3$ with 3 values at each decimal point (e.g. $1e-7$, $3e-7$, $6e-7$, $1e-6$ etc.) and a fixed pruning threshold of $1e-4$. Finally, for FlipOut, we use the values of $p=2$ (Eq. 1) and $\lambda=1$ (Eq. 3) for all experiments, a choice we motivate in Section 4.2.

4.2 Choosing the hyperparameters for FlipOut

We have experimented with different values of the two hyperparameters and found that $p=2$ (Eq. 1a) and $\lambda=1$ (Eq. 3a) offer consistent, strong results for all networks tested. In the following paragraphs, we detail the procedure used in determining these values.

Choosing $\boldsymbol{\lambda}$ For λ, we have run all networks at 15 different values, ranging from 0.75 to 1.5 in increments of 0.05. The value of $p=2$ was used. The networks are evaluated on a validation set, created by removing a random subset of samples from the training set. The size of the validation set was 10000 for CIFAR10 and 2000 for Imagenette. For our subsequent experiments, (Sections 4.3 and 4.4), the networks have been trained on the full training set. As a metric, we have used the accuracy of the networks at the end of training for the sparsity levels of 93.75% and 99.9%. We provide in Table 2 the accuracies generated by the optimal

Table 2: Accuracies when using the best value of λ discovered by grid search and the value of $\lambda = 1$ at two levels of sparsity. The parantheses indicate the gain offered by the optimal parameter.

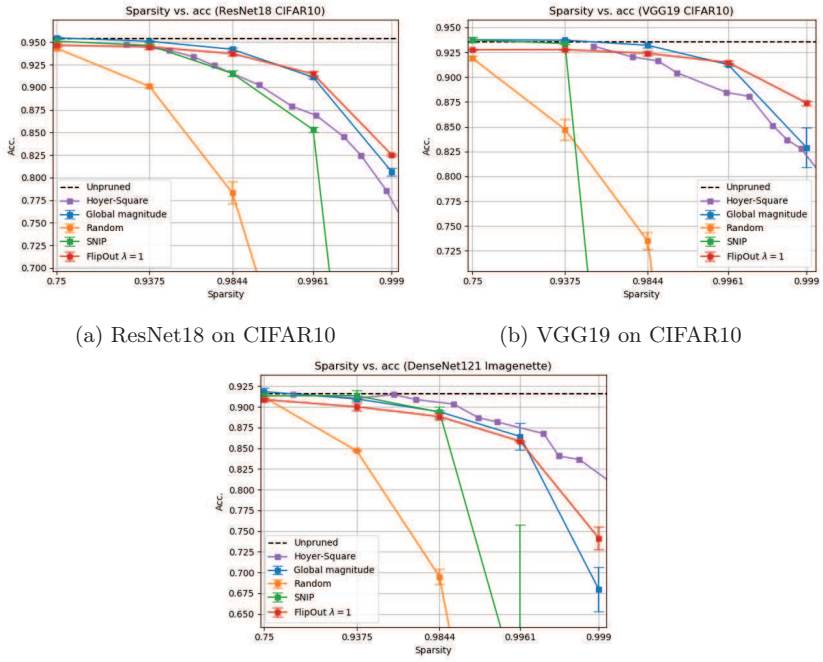
	Acc. at sparsity 93.75%		Acc. at sparsity 99.9%	
Model	λ^*	$\lambda = 1$	λ^*	$\lambda = 1$
ResNet18	94.58(+0.02)	94.56	83.75(+1.68)	82.07
VGG19	93.07(+0.11)	92.96	87.72(+0.48)	87.24
DenseNet121	89.75(+0.0)	89.75	73.5(+1.45)	72.05

Table 3: Table of results for different values of p at two levels of sparsity.

	Acc. at sparsity 93.75%					Acc. at sparsity 99.9%				
Model	$p = 0$	$p = \frac{1}{2}$	$p = 1$	$p = 2$	$p = 4$	$p = 0$	$p = \frac{1}{2}$	$p = 1$	$p = 2$	$p = 4$
ResNet18	93.71	88.39	94.18	**94.26**	94.11	72.69	77.08	79.83	82.07	**83.15**
VGG19	91.68	82.44	92.56	**92.96**	92.57	81.48	80.69	86.01	**87.24**	86.64
DenseNet121	10.35	77.40	88.9	**89.75**	88.86	10.35	10.35	70.85	**72.05**	60.55

value of λ, as discovered through this process, and the ones generated at $\lambda = 1$. Notice that the differences are almost negligible at 93.75% sparsity. For the larger sparsity level the disparity increases, although the default value still remains within 2 percentage points of the optimum value for all networks considered. The largest gap can be seen for ResNet18 and DenseNet121, at approximately 1.7 and 1.5 percentage points, respectively. Since there are only two out of six cases in which optimizing λ has helped beyond a negligible amount, we have used the value of 1 for this hyperparameter throughout our experiments.

Choosing *p* We perform similar experiments for p on five values, $p \in \{0, \frac{1}{2}, 1, 2, 4\}$. Note that the value of $p = 0$ corresponds to the case when the magnitudes of the weights are not taken into account; that is, the pruning decisions will be made solely based on the number of sign flips. As can be seen in Table 3, the value of $p = 2$ consistently outperforms all other tested values, with the exception of ResNet18 at 99.9% sparsity, for which the value of $p = 4$ achieves better results by approximately 1 percentage point. Another interesting observation is that the values of 1, 2 and 4 tend to perform better than 0 and $\frac{1}{2}$; we conjecture that this is due to the fact that deceitful shots (Section 2.1) occur when not taking into account the distance between the weight and its hypothesised local optimum, which have a negative impact on the pruning decision. This can be especially observed at the higher sparsity level and in the case of DenseNet121, where pruning with $p = 0$ causes the network to not perform better than random guessing. Given that the value of $p = 2$ is favored in 5 out of 6 cases, we have decided to use it as a default value in our subsequent experiments.



(a) ResNet18 on CIFAR10 (b) VGG19 on CIFAR10

(c) DenseNet121 on Imagenette

Fig. 4: Results of pruning ResNet18 and VGG19 on the CIFAR10 dataset. Each point is averaged over 3 runs; error bars indicate standard deviation.

4.3 Comparison to baselines

The results for the three models tested are found in Figure 4. FlipOut obtains state-of-the-art performance on ResNet18 and VGG19 for sparsity levels of 99.61% and beyond. For the highest tested sparsity level, it outperforms the second-best method by 1.9 and 4.5 percentage points, respectively (Fig. 4a, 4b). Notably, when using FlipOut on VGG19 for this sparsity, the drop in accuracy compared to the unpruned model is only 6.2 percentage points. At the same time, it remains competitive with other baselines for lower degrees of sparsity, staying within a 1 percentage point difference compared to the best method and with a minimal drop relative to the unpruned model. For DenseNet121, however, Hoyer-Square dominates all other methods tested in most cases (Fig. 4c), with FlipOut as second best for the highest sparsity level.

Interestingly, the simple criterion of magnitude pruning, when modified to rank the weights globally instead of a layer-by-layer basis, is competitive with other, more recent, baselines, and even obtains state-of-the-art results for moderate levels of sparsity. However, at high levels of sparsity, which correspond to more

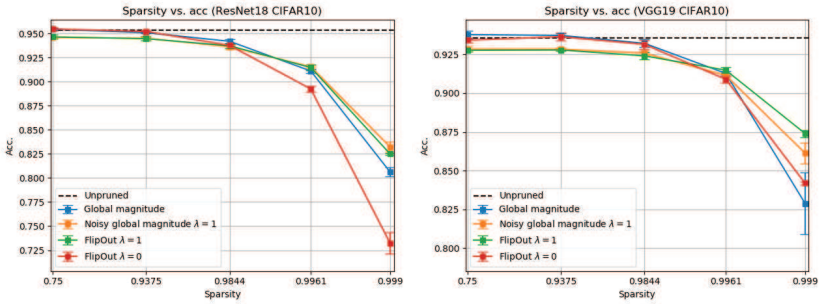
frequent and implicitly earlier pruning steps (Table 1) there is a performance degradation. This suggests that the magnitude of a weight by itself is not a good measure of saliency when the network is far from reaching convergence. It is also worth noting that SNIP collapses at high levels of sparsity, causing the network to perform no better than random guessing. Upon inspecting these cases (not shown for visibility) we noticed that at least one layer has been entirely pruned, effectively blocking any signal from passing. Interestingly, this does not happen for any of the other baselines (except for Random). We conjecture that this collapse as well as the cases where SNIP performs worse than random pruning (Fig. 4b) are a result of pruning at initialization; pruning too too early can cause the saliency criterion to be inaccurate, but also impedes training in and of itself.

During our experiments, we empirically observed that Hoyer-Square requires extensive hyperparameter tuning for optimal performance. Our method, however, has strong default values and can also target the final sparsity directly, while also not requiring additional epochs of fine-tuning. Finally, SNIP, the only other baseline which does not suffer from any of the issues commonly found among pruning methods (Section 1) compromises on performance for high levels of sparsity, whereas FlipOut does not.

4.4 Is it just the noise?

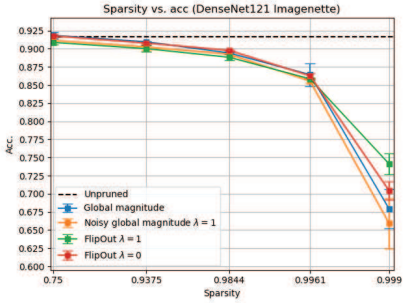
The performance of FlipOut could simply be a result of the noise addition, which is known to aid optimization ([19,25]). To investigate this, we perform experiments with global magnitude as the pruning criterion in which we add noise into the gradients using the recipe from Equation 3c and compare it to our own method. Notably, the saliency criterion of these two methods differ only in that FlipOut normalizes the magnitude by the number of sign flips (denominator in Eq. 1a). The hyperparameters were kept at their default values of $p = 2$ for FlipOut and $\lambda = 1$ for both methods. We also include runs of FlipOut where no noise was added (i.e. $\lambda = 0$). These serve as a control, decoupling the two novel components of our method: noise addition and scaling magnitudes by the number of sign flips. The same pruning rates and frequency of pruning steps have been used as before (Table 1). The results are illustrated in Fig. 5.

For sparsity levels up to 98.44%, adding gradient noise causes a slight deterioration on performance, as can be seen by the fact that both global magnitude and FlipOut with $\lambda = 0$ outperform their noisy counterparts. It can also be seen that FlipOut with $\lambda = 1$ performs comparably to noisy global magnitude, indicating that measuring saliency by sign flips does not benefit accuracy in these regimes compared to using only the magnitude, and the performance gap between the noisy and non-noisy methods is likely a result of noise addition. For sparsity levels of 99.61% and above, however, the opposite is true. It seems that gradient noise disproportionately benefits networks with a small number of remaining parameters; we conjecture that this is due to the fact that the exploration in parameter space induced by noise is more effective when that space is heavily constrained. Focusing on the highest level of sparsity, FlipOut outperforms noisy global magnitude on VGG19 (Fig. 5b) and DenseNet121 (Fig. 5c) by 1.2 and



(a) ResNet18 on CIFAR10

(b) VGG19 on CIFAR10



(c) DenseNet121 on Imagenette

Fig. 5: Results of the ablation study on the noise. Each point is averaged over 3 runs. Global magnitude without adding noise is also shown for comparison.

8.2 percentage points, respectively, while being outperformed by 0.8 percentage points on ResNet18 (Fig. 5a). The standard deviation of FlipOut at this point is lower than for noisy global magnitude for all networks tested, making it more robust to initial conditions and the noise sampling process. At this level, the addition of gradient noise to FlipOut also shows performance boosts compared to its non-noisy counterpart, namely 9.3 percentage points for ResNet18, 3.2 for VGG19 and 3.7 for DenseNet121. The benefits caused by adding noise to global magnitude as compared to adding it to FlipOut are similar for VGG19; however, it is relatively small for ResNet18 at 2.6 percentage points and even causes a 2 percentage point drop in performance for DenseNet121.

Since FlipOut with $\lambda = 1$ outperforms noisy global magnitude in 2 out of 3 cases for the highest level of sparsity while maintaining similar performance in all other cases as well as being less sensitive to the choice of seed, we conclude that its results cannot be explained only by the addition of noise and is also caused by the sign flips being taken into account when computing saliency.

Additionally, we conjecture that occurrences of under-shooting are indeed converted into over-shooting when adding gradient noise, allowing FlipOut to more accurately compute saliencies. This is evidenced by the fact that gradient noise addition benefits FlipOut more so than it does global magnitude, and implies that our method of dealing with deceitful shots is sound.

5 Discussion

In this work, we introduce the aim test, a general method for determining whether a point represents a local optimum for a weight during training, and propose using it for pruning by applying the test for all weights simultaneously and framing it as a saliency criterion. This method, coined FlipOut, demonstrates several desirable qualities: it is computationally tractable, allows for an exact level of sparsity to be selected, requires a single training run and has default hyperparameter settings which generate near optimal results, easing the burden of hyperparameter search.

We compare the performance of FlipOut to relevant baselines from literature on a variety of object classification architectures. We show that it achieves state-of-the-art performance at the highest levels of sparsity tested for 2 out of 3 networks, and maintains competitive performance in less sparse regimes. Finally, we conduct an ablation study on the two components of our algorithm, gradient noise addition and the saliency criterion, and find that both play an important role in yielding this performance performance.

References

1. Bellec, G., Kappel, D., Maass, W., Legenstein, R.: Deep rewiring: Training very sparse deep networks. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=BJ_wN01C-
2. Bottou, L.: Online algorithms and stochastic approximations. In: Saad, D. (ed.) Online Learning and Neural Networks. Cambridge University Press, Cambridge, UK (1998), http://leon.bottou.org/papers/bottou-98x, revised, oct 2012
3. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=B1xsqj09Fm
4. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=rJl-b3RcF7
5. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Advances in neural information processing systems. pp. 1135–1143 (2015)
6. Hassibi, B., Stork, D.G.: Second order derivatives for network pruning: Optimal brain surgeon. In: Advances in neural information processing systems. pp. 164–171 (1993)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)

8. Howard, J.: Imagenette (2019), accessed April 6th, 2020 at https://github.com/fastai/imagenette/tree/6395a747bef7a9760b95cd582ece09d90f8a4769
9. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
10. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. International Conference on Learning Representations (12 2015)
11. Ko, V., Oehmcke, S., Gieseke, F.: Magnitude and uncertainty pruning criterion for neural networks. In: 2019 IEEE International Conference on Big Data (Big Data). pp. 2317–2326 (2019)
12. Krizhevsky, A.: Learning multiple layers of features from tiny images (2009)
13. LeCun, Y., Denker, J.S., Solla, S.A.: Optimal brain damage. In: Touretzky, D.S. (ed.) Advances in Neural Information Processing Systems 2, pp. 598–605. Morgan-Kaufmann (1990), http://papers.nips.cc/paper/250-optimal-brain-damage.pdf
14. Lee, N., Ajanthan, T., Torr, P.: Snip: Single-shot network pruning based on connection sensitivity. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=B1VZqjAcYX
15. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. In: International Conference on Learning Representations (2017)
16. Liu, J., Xu, Z., Shi, R., Cheung, R.C.C., So, H.K.: Dynamic sparse training: Find efficient sparse network from scratch with trainable masked layers. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=SJlbGJrtDB
17. Louizos, C., Welling, M., Kingma, D.P.: Learning sparse neural networks through l_0 regularization. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=H1Y8hhg0b
18. Molchanov, D., Ashukha, A., Vetrov, D.: Variational dropout sparsifies deep neural networks. In: Proceedings of the 34th International Conference on Machine Learning (2017)
19. Neelakantan, A., Vilnis, L., Le, Q.V., Sutskever, I., Kaiser, L., Kurach, K., Martens, J.: Adding Gradient Noise Improves Learning for Very Deep Networks. arXiv e-prints arXiv:1511.06807 (Nov 2015)
20. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: International Conference on Learning Representations (2015)
21. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in nlp. In: ACL (2019)
22. Tieleman, T., Hinton, G.: "Lecture 6.5—rmsprop: Divide the gradient by a running average of its recent magnitude". COURSERA: Neural Networks for Machine Learning (2012)
23. Train CIFAR10 with PyTorch (GitHub Repository), Unknown Author: Pytorch cifar-10 github repository (2017), accessed April 6th, 2020 at https://github.com/kuangliu/pytorch-cifar/tree/ab908327d44bf9b1d22cd333a4466e85083d3f21
24. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
25. Welling, M., Teh, Y.W.: Bayesian learning via stochastic gradient langevin dynamics. In: Proceedings of the 28th international conference on machine learning (ICML-11). pp. 681–688 (2011)
26. Yang, H., Wen, W., Li, H.: Deephoyer: Learning sparser neural network with differentiable scale-invariant sparsity measures. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=rylBK34FDS

Emotion Intensity and Gender Detection via Speech and Facial Expressions

Elahe Bagheri[1], Oliver Roesler[2], Hoang-Long Cao[1], and Bram Vanderborght[1]

Robotics and Multibody Mechanics Research Group, Vrije Universiteit Brussel and Flanders Make, Brussels, Belgium.
Artificial Intelligence Lab, Vrije Universiteit Brussel, Brussels, Belgium.
elahe.bagheri@vub.be, oliver@roesler.co.uk, hoang.long.cao@vub.be,
bram.vanderborght@vub.be

Abstract. Human emotion detection has received increasing attention over the last decades for a variety of applications and systems. However, detecting the intensity of the expressed emotion has not been investigated as much as detecting the type of the expressed emotion. To fill this gap, we investigate the utility of different facial and speech features for emotion intensity detection. To this end, we designed different Deep Neural Network based models and applied them to the RAVDESS dataset. Obtained results show that speech signal features are better indicators of emotion intensity than facial features. However, in the absence of speech signals, finding emotion intensity by facial expressions is more accurate for males in comparison to females.
The difference between the accuracy of emotion intensity detection for two genders motivated us to use speech signals for the gender detection task. The obtained results confirm that the proposed model achieves higher accuracy in emotion intensity detection and is more robust in gender detection than the state-of-the-art.

Keywords: Emotion Intensity Detection · Gender Detection · Deep Learning.

1 INTRODUCTION

Detecting human emotions is crucial in developing cognitive and adaptive behaviors for artificial intelligent systems, robots, and (virtual) agents. Emotion detection is the ability to recognize another's affective state, which typically involves the integration and analysis of human expressions through different modalities, like facial expression, speech, body movements, and gestures [5]. Mehrabian [21] showed 55% of human emotions are conveyed through facial expression and 38% through speech, therefore, facial and speech emotion recognition received significant attention during the last decades. Although finding the type of expressed emotion is essential to adapt to a user's affective state, it is not enough, and a difference in intensity has been proven to be important to distinguish different emotional states [13]. For instance, a polite smile versus embarrassed smile [1] and posed versus spontaneous smile are separable by differences in their expression intensities [7]. Since there is not much research on emotion intensity detection, in the following sections, the state-of-the-art in speech emotion recognition and facial emotion recognition are discussed.

1.1 Speech Emotion Recognition (SER)

The effect of emotions can be seen in both acoustic characteristics and lexical content of speech. Some examples of acoustic features are Mel-Frequency Cepstral Coefficients (MFCC), energy, jitter, and shimmer, which are known as Low-Level Descriptors (LLDs) [14] [1]. Some examples of lexical features are the presence/absence of word stems, and bag-of-words sentiment categories [28]. However, when the linguistic content is not emotionally rich, recognizing emotion from the transcript is very difficult [28], thus, in this study, our focus is on the acoustic characteristics for recognizing the emotion intensity.
In traditional SER methods, the acoustic features are first extracted and then different machine learning algorithms like Support Vector Machine (SVM) [23], K-nearest neighbor [15] and Hidden Markov model [26] are applied to the obtained features to classify them into considered emotion classes. To obtain these features from utterance-level, each signal is broken into shorter frames of 20 to 50 milliseconds, and their features, i.e., frame-level features, are extracted. However, emotional contents are not in static values of these features but are in their temporal variations. Thus, different statistical functions, e.g., minimum, maximum, mean, variance, linear regression coefficients, etc., are applied to these

[1] Some other investigated acoustic characteristics, i.e., LLDs, are zero-crossing rate, duration, and higher-order formants, Mel-filterbank features, spectral features, formant locations/bandwidths, perceptual linear prediction, fundamental frequency, and pitch.

features to illustrate their temporal variations and contours. The obtained results, afterward, are unified in a vector to achieve utterance-level features [22].
Due to the success of deep learning in different fields like image, video, and natural language processing, the interest in applying Deep Neural Networks (DNN) for speech emotion recognition also increased. Authors in [11] and [17] used deep feed-forward and recurrent neural networks to learn the frame-level acoustic features, and used extreme learning machines for the utterance-level aggregation. Mirsamadi et al. [22] used Rectified Linear Unit (ReLU) dense layers to learn frame-level features, and Bidirectional Long Short-Term Memory (BiLSTM) recurrent layers to learn the temporal aggregation. Neumann et al. [25] used a Convolutional Neural Network (CNN) with one convolutional layer and one pooling layer, to learn the representation of the audio signal, and an attention layer to compute the weighted sum of all the information extracted from different parts of the input. Lim et al. [18] transformed the speech signal into 2D representation using Short Time Fourier Transform and sent them to concatenated CNN and LSTM architectures without using any traditional hand-crafted features. Trigeorgis et al. [32] applied two BiLSTM layers to balance the frame-level characterization and utterance-level aggregation and transform frame-level convolutional features directly into continuous arousal and valence output so that the model learned direct mapping from time-domain speech signals into the continuous model of emotion. The temporal model proposed in [12] used BiLSTM to represent forward/backward contextual information of temporal dynamics of the speech signal and conducted a CNN and a capsule net to learn temporal clusters and classify the extracted patterns. Mustaqeem et al. [24] obtained the spectrogram of signals and then used CNN and LSTM to classify the speech.

1.2 Facial Emotion Recognition (FER)

Ekman [8] showed six basic emotions[2] are expressed universally the same through facial muscles. He introduced the Action Unit (AU) to indicate fundamental movements of a single or group of muscles through the facial expression of a special emotion [3]. He also defined the Facial Action Coding System (FACS) to encode the movements of these AUs [10]. One way to recognize a facially expressed emotion is detecting the status of all individual AUs and then analyzing the combination of the activated AUs to obtain the expressed emotion. On the other hand, promising results of DNN based approaches in comparison with classical machine learning algorithms lead to the proposal of numerous DNN based FER methods in the research community. For instance, Bagheri et al. [2] used facial muscle activities as raw input for a Stacked Auto Encoder (SAE). The applied SAE returns the best combination of muscles in describing a particular emotion, which is then sent to a Softmax layer to fulfill the multi-classification task. Liu et al [19] proposed a sign-based DNN architecture to investigate the effect of AUs in emotion recognition. The proposed model consists of three sequential modules, where the first module generates a complete representation of all expression-specific appearance variations by a convolution layer stacked by a max-pooling layer. The second module finds the best simulation of the combination of the AUs and the last module learns hierarchical features by Restricted Boltzmann Machines (RBM). Finally, a linear SVM classifier is used to recognize the six basic emotions. However, AU-aware layers, in the second module, are not able to detect all FACS in images. Pitaloka et al [27] used CNN to extract features from an input image, which is then passed to a max-pooling layer to reduce the image size. A fully connected layer, in the end, classifies the input image into one of the six basic emotions. However, the performance of the proposed algorithm decreases when the dimension of the input image increases regarding the complexity of the high dimensional images.

Research on emotion intensity detection has been focused on the estimation of the intensity of Action Units (AU), e.g., [6, 13] and FERA 2015[4], however, there is no conclusion about the intensity of the expressed emotion, thus, the goal of this study is developing a model by which the intensity of the expressed emotion in a given image, speech signal or video can be identified.
The remainder of this paper is structured as follows: the applied models, dataset, and extracted features are explained in Section 2. Section 3 demonstrates the conducted experiments and obtained results. Finally, Section 4 concludes this paper.

[2] Happiness, sadness, fear, anger, surprise, and disgust.
[3] https://imotions.com/blog/facial-action-coding-system/
[4] Facial Expression Recognition and Analysis challenge

Table 1
The architectures of the proposed DNN based models.

LSTM		BiLSTM		CNN		
Simple	**Attention**	**Simple**	**Attention**	**Simple**	**BiLSTM/LSTM**	**BiLSTM/LSTM+Attention**
LSTM	LSTM	BiLSTM	BiLSTM	CNN	CNN	CNN
Dropout	Attention	Dropout	Attention	CNN	CNN	MaxPooling
Dense	Dropout	Dense	Dropout	Dropout	MaxPooling	CNN
	Dense		Dense	MaxPooling	Flatten	MaxPooling
				Flatten	BiLSTM/LSTM	Flatten
				Dense	Dense	BiLSTM/LSTM
				Dense		Attention
						Dropout
						Dense

2 METHODOLOGY

2.1 Applied Models

Table 1 shows the number, type, and order of layers of proposed models that are applied to fulfill the emotion intensity detection task. The parameter settings are as follows: convolution layers are all 1D and have 64 filters and kernel size of three. ReLU activation function is applied for adding non-linearity. Dropout layers are used as regularizers and their ratio is set to 0.1. 1D max-pooling layers, with a kernel size of four are used to introduce sparsity in the network parameters and to learn deep feature representations. Dense layers are used with the activation functions of sigmoid for finding the predicted binary distribution of the target class. The number of epochs is selected as 250 and the batch size is set to 128. The number of units in applied LSTM and BiLSTM networks is five.

2.2 Dataset

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) [20] is used to train and test the proposed models. RAVDESS contains videos which provide both facial and speech features. Each video lasts approximately three seconds and contains frontal face poses of twelve female and twelve male, all north American actors and actresses, speaking and singing two lexically-matched sentences while expressing six basic emotions plus calmness, and neutral. In this study, we only used speaking records. Further, expressed emotions in RAVDESS are categorized into two levels of normal and strong intensities, which makes it a good option for emotion intensity detection.
The dataset is partitioned, in a subject independent manner, into the train and test sets, i.e., the videos related to the first eighteen actors (nine female, nine male) are used for training, and the videos of the next six subjects (three female, three male) are used for testing. Since videos are recorded with 30 fps, 30 images are extracted per second for facial analysis. However, as each video starts from a neutral state, reaches an apex, and goes back to a neutral state, the first and last twenty obtained frames per video are discarded.
Since the expressions of neutral are not categorized as normal or strong, this emotional state is not considered for emotion intensity detection. Additionally, the normal and strong expressions of calmness are only marginally different, therefore, they are omitted from the emotion intensity detection.

2.3 Features and Data Pre-processing

To obtain facial and speech features, two open-source toolkits, i.e., OpenFace [3] and openEAR [9], are used. OpenFace is able to return different features including facial landmarks, head pose, facial action units activity, and eye-gaze from both video and image inputs. The applied features in this study are facial landmarks, facial action units activity, and face rigid and non-rigid shape parameters leading to a vector of 378 elements [5]. The obtained feature values are normalized between zero and one.

[5] Other features provided by OpenFace were also investigated, however, the mentioned features resulted in the highest classification accuracy.

Table 2
The obtained accuracies for emotion intensity detection by proposed models on RAVDESS dataset based on facial and speech features (without neutral and calmness expressions). The results were obtained over ten repetitions.

(a) Facial features.

Data	LSTM		BiLSTM		CNN			CNN	
	Simple	Attention	Simple	Attention	Simple	LSTM	BiLSTM	LSTM+Attention	BiLSTM+Attention
Female and Male	53.34	52.46	53.27	54.13	55.96	55.06	54.79	55.31	**56.24**
Female	51.67	50.31	50.14	51.12	52.50	53.52	51.72	50.63	**54.72**
Male	54.24	53.72	52.81	53.66	58.38	56.45	54.26	56.97	**58.31**

(b) Speech features.

Data	LSTM		BiLSTM		CNN			CNN	
	Simple	Attention	Simple	Attention	Simple	LSTM	BiLSTM	LSTM+Attention	BiLSTM+Attention
Female and Male	63.5	61.70	67.65	69.03	69.45	68.54	69.46	60.53	**73.53**
Female	68.51	67.62	67.41	69.52	72.61	72.45	72.43	59.83	**75.67**
Male	57.36	55.30	55.97	55.21	59.73	59.18	59.72	58.87	**65.5**

openEAR is the open-source toolkit that is used for speech feature extraction. It analyses the speech signals and returns three different sets of features based on the applied configuration, i.e., INTERSPEECH 2009, emobase, and INTERSPEECH 2013. In this study, the INTERSPEECH 2009 (emo-IS09) [29] configuration is used, which leads to 384 features including minimum, maximum, and mean values of different speech features. In this study, only the MFCC and PCM set of features are used, which lead to a vector of 156 elements [6]. The obtained feature values are normalized between zero and one.

3 EXPERIMENTAL SCENARIOS AND OBTAINED RESULTS

3.1 Experiment I: Emotion Intensity Detection

As the main goal of this study is to identify the intensity of an expressed emotion by a user, the facial and speech related features of all subjects are extracted and pre-processed (as explained in Section 2.3) and are given to the proposed models (Table 1). The obtained results in Table 2 show that speech features lead to higher accuracy in emotion intensity detection than facial features. Further, Table 1 shows the combination of convolutional layer with the BiLSTM and Attention layers (CNN+BiLSTM+Att) achieves the highest performance, i.e., 73.53%, which is higher than state-of-the-art, i.e., 70.4% [12] (Table 4).
Although speech features lead to higher accuracy than facial features, Table 2.a shows that the accuracy of the models in identifying the intensity of the expressed emotions by males is higher than expressed emotions by females when facial features are used, i.e., 58.31% vs 54.72%. In comparison, when speech features are used, the obtained accuracy for females is higher than for males, i.e., 75.67% vs 65.5% (Table 2.b). La Mura [16] showed some of the speech features related to emotion recognition are related to the subject's gender. Thus, one explanation can be that females convey more details about the intensity of their emotions through their speech. To verify this hypothesis, the CNN+BiLSTM+Att model is separately applied to both the facial and speech features of each individual subject. The obtained results (Table 3.a) show that for males obtaining the emotion intensity by facial expressions is more accurate than for females, i.e., the minimum and maximum accuracies for males are 63.92% and 78.79%, respectively, while the corresponding values for females are 58.26% and 71.15%. On the other hand, Table 3.b shows finding emotion intensity via speech features for females is more accurate than males, i.e., the minimum and maximum accuracies for females are 71.49% and 95.83% while the corresponding values for males are 59.29% and 85.71%, respectively.

[6] Different combinations of features were used, however, the highest accuracy was obtained for the the applied feature set, thus, we did not use the other features. In addition, since openEAR is able to analyze a file, we did not trim the videos into smaller intervals, which reduced the running time remarkably.

Table 3
Accuracy of emotion intensity detection based on facial and speech data for males and females. The results obtained over ten repetitions.

(a) Facial features.

Male	Acc	STD	Female	Acc	STD
Sub#1	64.97	2.3	Sub#2	68.36	1.6
Sub#3	77.53	2.5	Sub#4	66.05	2.5
Sub#5	78.79	1.3	Sub#6	69.46	1.4
Sub#7	75.85	2.6	Sub#8	62.17	5.3
Sub#9	72.62	1.8	Sub#10	67.55	1.2
Sub#11	72.71	2.1	Sub#12	68.42	4.7
Sub#13	63.92	3.1	Sub#14	69.19	2.2
Sub#15	66.03	2.5	Sub#16	63.11	3.6
Sub#17	67.78	2.1	Sub#18	71.15	2.1
Sub#19	73.23	2.3	Sub#20	69.32	2.8
Sub#21	74.91	1.9	Sub#22	61.62	1.4
Sub#23	73.84	2.7	Sub#24	58.26	1.9

(b) Speech features.

Male	Acc	STD	Female	Acc	STD
Sub#1	82.86	8.3	Sub#2	88.46	4.2
Sub#3	84.21	3.0	Sub#4	87.85	4.8
Sub#5	59.29	4.8	Sub#6	85.57	4.1
Sub#7	80.71	5.8	Sub#8	75.49	4.9
Sub#9	85.71	6.3	Sub#10	71.49	3.3
Sub#11	70.49	7.3	Sub#12	78.57	6.7
Sub#13	70.01	11.0	Sub#14	74.29	3.6
Sub#15	60.00	7.6	Sub#16	79.23	5.1
Sub#17	85.71	6.7	Sub#18	95.83	4.3
Sub#19	74.29	6.9	Sub#20	67.17	8.8
Sub#21	67.50	7.3	Sub#22	72.5	9.6
Sub#23	83.50	9.8	Sub#24	95.38	5.3

Table 4
Comparison between the proposed model and the state-of-the-art for emotion intensity detection over RAVDESS on speech features in a subject independent manner.

Research	Architecture	Accuracy
Jalal [12]	CNN + BiLSTM + CapsuleNet	70.4%
Proposed model	CNN + BiLSTM + Attention	**73.53%**

3.2 Experiment II: Gender Detection

Since the obtained accuracies for emotion intensity detection for males and females are noticeably different, in this experiment we investigated the speech and facial features for the task of gender detection. As the results of the proposed models (Table 1) by using facial features were not promising, a new model was designed for this experiment. The new proposed model uses raw images of 200×200 pixels as input and consists of four layers, wherein each a 2D convolutional layer is followed by a max-pooling and a dropout layer. The kernel size of the convolution layers is 3×3, with the same padding size, and ReLU is used as the activation function. The max-pooling layer is 2×2 and dropout rates in different layers are set to 0.6, 0.4, 0.2, and 0.2, respectively. The batch size during the train and test is set to 32. The first eighteen subjects are used for training and the last six subjects are used for testing (subject independent and gender balance). The obtained accuracy of this model is 70.46%.

Repeating the experiment with the speech features led to higher accuracy for gender detection via the proposed models in Table 1. More specifically, CNN+BiLSTM+Att model obtained an accuracy of 89.8% for gender detection using the MFCC and PCM feature sets, which is the highest obtained accuracy in comparison with the other proposed models in Table 1. Table 5 shows the obtained confusion matrix by the proposed model for gender detection. We noticed that 20 of the female samples that are wrongly predicted as male belong to one subject.

A straightforward comparison between the proposed model and the state-of-the-art for gender detection task, using speech signals of RAVDESS dataset, is difficult. For instance, Singh et al. [31] performed gender detection in each individual emotion class assuming the emotion class is known. Bansal et al. [4] used only four expressions of RAVDESS for gender detection and obtained an accuracy of 94.12%, and Shaqra et al. [30] considered six emotions and obtained an accuracy of 98.67%, while a gender detection model should be robust to various emotions. Thus, in this study, we used all expressed emotional states in RAVDESS dataset, i.e., eight emotional states, for the task of gender detection. Table 6 compares the obtained accuracy by the proposed model with the state-of-the-art. Although the proposed model could not beat the state-of-the-art, it is more robust since it considers more emotional states.

Table 5
Confusion matrix for gender detection.

	Predicted Female	Predicted Male
Actual Female	143	25
Actual Male	9	159

Table 6
Comparison between the proposed model and the state-of-the-art for gender detection over RAVDESS on speech features in a subject independent manner.

Research	Model	Accuracy
Bansal et al. [4] (four emotional states)	SVM	94.12%
Shaqra et al. [30] (six emotional states)	MLP	98.67%
Proposed model (eight emotional states)	CNN + BiLSTM + Attention	89.8%

4 CONCLUSION

In this study, we designed different deep neural network based models for emotion intensity and gender detection using features obtained by open-source toolkits. The RAVDESS dataset was used to evaluate the proposed models because it is, to the best of our knowledge, the only dataset that categorizes emotions based on their intensity.
The obtained results showed a difference between the obtained accuracy of emotion intensity detection for females and males based on the applied feature set, i.e., using facial features led to more accurate results for males than for females, while using speech features led to higher accuracy for females' emotion intensity detection. Additionally, the results showed that the MFCC and PCM feature sets led to higher accuracy than facial features in emotion intensity detection. Further, we used the proposed models for gender detection task using facial and speech features. The obtained results showed that gender detection is also more accurate by using speech features than facial features for the RAVDESS dataset. In addition, the obtained results showed that the proposed model is comparable with the state-of-the-art while it is more robust in terms of handling more emotional states.

ACKNOWLEDGEMENTS

The work leading to these results has received funding from Flanders Make Proud (PROgramming by User Demonstration) and the Flemish Government under the program Onderzoeksprogramma Artificiele Intelligentie (AI) Vlaanderen.

Bibliography

[1] Ambadar, Z., Cohn, J.F., Reed, L.I.: All smiles are not created equal: Morphology and timing of smiles perceived as amused, polite, and embarrassed/nervous. Journal of nonverbal behavior **33**(1), 17–34 (2009)

[2] Bagheri, E., Bagheri, A., Esteban, P.G., Vanderborgth, B.: A novel model for emotion detection from facial muscles activity. In: Iberian Robotics conference. pp. 237–249. Springer (2019)

[3] Baltrusaitis, T., Zadeh, A., Lim, Y.C., Morency, L.P.: Openface 2.0: Facial behavior analysis toolkit. In: Automatic Face & Gesture Recognition (FG 2018), 2018 13th IEEE International Conference on. pp. 59–66. IEEE (2018)

[4] Bansal, M., Sircar, P.: Phoneme based model for gender identification and adult-child classification. In: 2019 13th International Conference on Signal Processing and Communication Systems (ICSPCS). pp. 1–7. IEEE (2019)

[5] Caridakis, G., Castellano, G., Kessous, L., Raouzaiou, A., Malatesta, L., Asteriadis, S., Karpouzis, K.: Multimodal emotion recognition from expressive faces, body gestures and speech. In: IFIP International Conference on Artificial Intelligence Applications and Innovations. pp. 375–388. Springer (2007)

[6] Cohn, J.F., Kanade, T., Li, C.C.: Subtly different facial expression recognition and expression intensity estimation. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition. pp. 853–859 (1998)

[7] Cohn, J.F., Schmidt, K.: The timing of facial motion in posed and spontaneous smiles. In: Active Media Technology, pp. 57–69. World Scientific (2003)

[8] Ekman, P.: Facial action coding system (facs). A human face (2002)

[9] Eyben, F., Wöllmer, M., Schuller, B.: Opensmile: the munich versatile and fast open-source audio feature extractor. In: Proceedings of the 18th ACM international conference on Multimedia. pp. 1459–1462 (2010)

[10] Hamm, J., Kohler, C.G., Gur, R.C., Verma, R.: Automated facial action coding system for dynamic analysis of facial expressions in neuropsychiatric disorders. Journal of neuroscience methods **200**(2), 237–256 (2011)

[11] Han, K., Yu, D., Tashev, I.: Speech emotion recognition using deep neural network and extreme learning machine. In: Fifteenth annual conference of the international speech communication association (2014)

[12] Jalal, M.A., Loweimi, E., Moore, R.K., Hain, T.: Learning temporal clusters using capsule routing for speech emotion recognition. In: Proc. Interspeech. vol. 2019, pp. 1701–1705 (2019)

[13] Jeni, L.A., Girard, J.M., Cohn, J.F., De La Torre, F.: Continuous au intensity estimation using localized, sparse facial feature space. In: 2013 10th IEEE international conference and workshops on automatic face and gesture recognition (FG). pp. 1–7. IEEE (2013)

[14] Jin, Q., Li, C., Chen, S., Wu, H.: Speech emotion recognition with acoustic and lexical features. In: 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP). pp. 4749–4753. IEEE (2015)

[15] Kim, Y., Provost, E.M.: Emotion classification via utterance-level dynamics: A pattern-based approach to characterizing affective expressions. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 3677–3681. IEEE (2013)

[16] La Mura, M., Lamberti, P.: Human-machine interaction personalization: a review on gender and emotion recognition through speech analysis. In: 2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT. pp. 319–323. IEEE (2020)

[17] Lee, J., Tashev, I.: High-level feature representation using recurrent neural network for speech emotion recognition. In: Sixteenth annual conference of the international speech communication association (2015)

[18] Lim, W., Jang, D., Lee, T.: Speech emotion recognition using convolutional and recurrent neural networks. In: 2016 Asia-Pacific signal and information processing association annual summit and conference (APSIPA). pp. 1–4. IEEE (2016)

[19] Liu, M., Li, S., Shan, S., Chen, X.: Au-aware deep networks for facial expression recognition. In: 2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG). pp. 1–6. IEEE (2013)

[20] Livingstone, S.R., Russo, F.A.: The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english. PloS one **13**(5), e0196391 (2018)

[21] Mehrabian, A.: Nonverbal communication. Routledge (2017)

[22] Mirsamadi, S., Barsoum, E., Zhang, C.: Automatic speech emotion recognition using recurrent neural networks with local attention. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 2227–2231. IEEE (2017)

[23] Mower, E., Mataric, M.J., Narayanan, S.: A framework for automatic human emotion classification using emotion profiles. IEEE Transactions on Audio, Speech, and Language Processing **19**(5), 1057–1070 (2010)
[24] Mustaqeem, M., Kwon, S., et al.: A cnn-assisted enhanced audio signal processing for speech emotion recognition. Sensors **20**(1), 183 (2020)
[25] Neumann, M., Vu, N.T.: Attentive convolutional neural network based speech emotion recognition: A study on the impact of input features, signal length, and acted speech. arXiv preprint arXiv:1706.00612 (2017)
[26] Nwe, T.L., Foo, S.W., De Silva, L.C.: Speech emotion recognition using hidden markov models. Speech communication **41**(4), 603–623 (2003)
[27] Pitaloka, D.A., Wulandari, A., Basaruddin, T., Liliana, D.Y.: Enhancing cnn with preprocessing stage in automatic emotion recognition. Procedia computer science **116**, 523–529 (2017)
[28] Rozgić, V., Ananthakrishnan, S., Saleem, S., Kumar, R., Vembu, A.N., Prasad, R.: Emotion recognition using acoustic and lexical features. In: Thirteenth Annual Conference of the International Speech Communication Association (2012)
[29] Schuller, B., Steidl, S., Batliner, A.: The interspeech 2009 emotion challenge. In: Tenth Annual Conference of the International Speech Communication Association (2009)
[30] Shaqra, F.A., Duwairi, R., Al-Ayyoub, M.: Recognizing emotion from speech based on age and gender using hierarchical models. Procedia Computer Science **151**, 37–44 (2019)
[31] Singh, R., Puri, H., Aggarwal, N., Gupta, V.: An efficient language-independent acoustic emotion classification system. Arabian Journal for Science and Engineering **45**(4), 3111–3121 (2020)
[32] Trigeorgis, G., Ringeval, F., Brueckner, R., Marchi, E., Nicolaou, M.A., Schuller, B., Zafeiriou, S.: Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network. In: 2016 IEEE international conference on acoustics, speech and signal processing (ICASSP). pp. 5200–5204. IEEE (2016)

The algorithm versus the chimps: On the minima of classifier performance metrics

Joep Burger[1][0000−0002−7298−5561] and
Quinten Meertens[2,3,4][0000−0002−3485−8895]

[1] Statistics Netherlands, Research and Development, CBS-weg 11, PO Box 4481, 6401 CZ Heerlen, the Netherlands
j.burger@cbs.nl
[2] Center for Nonlinear Dynamics in Economics and Finance, University of Amsterdam, Roetersstraat 11, PO Box 15867, 1001 NJ Amsterdam, the Netherlands
[3] Leiden Centre of Data Science, Leiden University, the Netherlands
[4] Statistics Netherlands, Business Statistics (The Hague), the Netherlands
q.a.meertens@uva.nl

Abstract. In this paper we seek the minima of performance metrics for binary classification to facilitate comparison between metrics and applications, and to assess the quality of inferential statistics made from non-probability samples. We use these minima to min-max normalize the performance metrics so that they can be interpreted as a percentage of the perfect classifier relative to the proverbial chimps at the zoo[†] guessing at random. We compare our results with the balanced metrics that have been introduced recently, which are corrected for bias due to class imbalance.

Keywords: Inferential statistics · Non-probability samples · Statistical learning · Supervised machine learning · Binary classification.

1 Introduction

Imagine you have to do a school exam. The test consists of one hundred multiple choice questions. At each question you can choose from four possible answers (A, B, C or D). In the Netherlands, students pass when they score 6 points or more on a scale from 0 to 10. Is it sufficient to answer sixty questions correctly to pass the test? Maybe not, because the proverbial chimps at the zoo that choose answers randomly will on average answer twenty five questions correctly. The teacher could correct for that minimum by grading twenty five correct answers a 0 and then scaling linearly to maintain a perfect score when no mistakes are made by the student, Then, a student answering sixty questions correctly fails the test, having a score of 4.7 on the scale from 0 to 10. The student would now have to answer at least seventy questions correctly to pass the test. From a statistical point of view, an advantage of this so-called min-max normalization,

[†] Inspired by Swedish physician Hans Rosling (1948–2017).

is that the students' grades are now comparable with the grades at a second school where the students can choose between two possible answers (A or B). The proverbial chimps at the zoo that answer randomly will then answer fifty question correctly, on average. After min-max normalization, a student at that second school will only pass the test when answering eighty or more questions correctly.

In supervised machine learning, algorithms instead of students are trained to find the right answer to a multiple choice question. The algorithm learns, for example, that in the case of a drawing the subject is a 'moon', 'rose' or 'fish'. (These are the first words Dutch children learn to read. Many algorithms are still in elementary school and it is nice to use something else than pictures of cats and cars as an example.) What final grade does an algorithm get for its answers to new drawings? For this purpose a considerable list of performance metrics has been developed (see, e.g., [7]). For some of them it is unclear what the score of the control group would be: how well would the chimps at the zoo perform?

In this paper, we will provide an answer to that question. The answer is important from a statistical point of view: as students' grades should preferably be comparable between subjects and schools, we would like the performance of algorithms to be comparable between metrics and applications. Moreover, we would like to have a statistical interpretation of the actual value of a performance metric, such that the interpretation is independent of the metric and application, similar to the min-max normalized grade of a multiple-choice test: a grade equal to 6 can always be interpreted as 60 percent between guessing and perfection. That statistical interpretation is essential when employing supervised machine learning algorithms at national statistical institutes, such as Statistics Netherlands, to produce official statistics.

Official statistics provide quantitative information about the status and development of well-defined populations such as businesses or households. One of the challenges is to produce such information at reasonable accuracy, cost and time. A burning question in official statistics is how to make inference from non-probability (NP) samples [8]. NP samples like social media messages or sensor data can offset some disadvantages of questionnaires sent to units in a probability sample, such as response burden, high costs and a considerable time lag between data collection and dissemination [1]. However, not all units in the population of interest have a positive and known probability of being included in an NP sample. This rules out design-based estimators from sampling theory.

Alternatively, the data-generating mechanism of NP samples can be deduced by modeling the relationship between features and the target variable in the NP sample and use it to predict the missing data, assuming they are missing at random. Statistical models could then be deployed, but more often machine learning algorithms are used, because they are designed for prediction or extrapolation and scale better with the number of features.

To assess the quality of the extrapolations, the quality of the predictions are assessed on test sets for which the actual value is known. A range of performance

metrics exists to this end [7]. However, as noted before, it remains unclear what the proverbial chimps at the zoo would achieve by randomly guessing the value of the target variable. A typical example is high accuracy in imbalanced datasets: if a class has a relative frequency of 95%, it is easy to obtain a seemingly impressive accuracy of 95% by always 'predicting' the most common class.

In this paper we seek the minima of performance metrics for binary classification to facilitate comparison between metrics and to assess the quality of inferential statistics made from NP samples. We use these minima to min-max normalize the performance metrics so that they can be interpreted as percentage of perfection relative to the performance of the proverbial chimps at the zoo. We compare our results with balanced metrics [6], which have been corrected for bias due to class imbalance. In our view, the paper is a methodological contribution with preliminary simulation results that encourage a more thorough experimental study in the future.

2 Imbalanced performance metrics

We assume that we have a test set of n data points, n_1 of which are labeled positive: $y_i = 1$, where y_i is the observed class of instance i. The fraction $\alpha = n_1/n$ is referred to as the base rate. The algorithm trained on a training set of $N - n$ data points predicts for all n instances the probability that instance i belongs to the positive class: $\hat{p}_i = \mathbb{P}(y_i = 1)$. By choosing a cutoff $0 \leq c \leq 1$ above which the probability $\hat{p}_i$ is assigned to the positive class, a 2×2 contingency table or confusion matrix can be constructed (Table 1). Optimizing cutoff c is discussed in Section 4.

Table 1. Confusion matrix for cutoff c. Cells highlighted in gray can be used to derive all other cells and metrics.

	Predicted Positive	Negative	$\sum$	
Actual Positive	X_c	$n_1 - X_c$	n_1	$TPR_c = \frac{X_c}{n_1}$
Negative	Y_c	$n_2 - Y_c$	$n_2 := n - n_1$	$TNR_c = \frac{n_2 - Y_c}{n_2} = 1 - \frac{Y_c}{n_2}$
$\sum$	$X_c + Y_c$	$n - X_c - Y_c$	n	
	$PPV_c = \frac{X_c}{X_c + Y_c}$	$NPV_c = \frac{n_2 - Y_c}{n - X_c - Y_c}$	$\alpha = \frac{n_1}{n}$	

From this confusion matrix the following well-known performance metrics are derived (left two columns of Table 2). Accuracy (ACC_c) is the fraction of all cases that is predicted correctly. The true positive rate (TPR_c), also known as sensitivity or recall, is the fraction of positively labeled cases that is predicted correctly and the true negative rate (TNR_c), also known as specificity, is the fraction of negatively labeled cases that is predicted correctly. The positive predictive value (PPV_c), also known as precision, is the fraction of predicted

positive cases that is actually labeled 'positive' and the negative predictive value (NPV_c) is the fraction of predicted negative cases that is actually labeled 'negative'. The receiver operating characteristic curve, or ROC curve, plots TPR_c against the complement of TNR_c for $0 \leq c \leq 1$. The area under the ROC curve (AUC) is used as a performance metric. Note that TPR_c will decrease with c whereas TNR_c will increase with c. This trade-off is captured by Youden's J index (J_c) or Peirce Skill Score, which is also the vertical distance between the ROC curve and the diagonal. Note also that PPV_c will become unstable at higher values of c, whereas NPV_c will become unstable at lower values of c, because the respective denominators decrease there. This trade-off is captured by markedness (MRK_c). The Matthews correlation coefficient (MCC_c) is the correlation between the actual and predicted binary classifications. The positive F_1 score (PF_{1c}) is the harmonic mean of TPR_c and PPV_c. Analogously, the negative F_1 score (NF_{1c}) is the harmonic mean of TNR_c and NPV_c. The harmonic mean is more sensitive to one of the values being low than the arithmetic mean.

Table 2. Imbalanced performance metrics and their expected value when randomly guessing the positive class with probability g.

Metric Q	Definition [7]	$\mathbb{E}[Q(g)]$
ACC_c	$\frac{n_2+X_c-Y_c}{n}$	$\alpha g + (1-\alpha)(1-g)$
TPR_c	$\frac{X_c}{n_1}$	g
TNR_c	$1-\frac{Y_c}{n_2}$	$1-g$
PPV_c	$\frac{X_c}{X_c+Y_c}$	$\alpha + O(\frac{1}{n^2})$
NPV_c	$\frac{n_2-Y_c}{n-X_c-Y_c}$	$1-\alpha+O(\frac{1}{n^2})$
AUC	$\int_{c=0}^{1} TPR_c dTNR_c$	$\frac{1}{2}$
J_c	$TPR_c + TNR_c - 1$	0
MRK_c	$PPV_c + NPV_c - 1$	$0+O(\frac{1}{n^2})$
MCC_c	$\frac{n_2X_c-n_1Y_c}{\sqrt{n_1n_2(X_c+Y_c)(n-X_c-Y_c)}}$	$0+O(\frac{1}{n^2})$
PF_{1c}	$\frac{2}{\frac{1}{TPR_c}+\frac{1}{PPV_c}}$	
	$=\frac{2X_c}{n_1+X_c+Y_c}$	$2\alpha g\left(\frac{1}{\alpha+g}-\frac{\alpha(1-g)}{n(\alpha+g)^3}\right)+O\left(\frac{1}{n^2}\right)$
NF_{1c}	$\frac{2}{\frac{1}{TNR_c}+\frac{1}{NPV_c}}$	
	$=\frac{2(n_2-Y_c)}{n+n_2-X_c-Yc}$	$2(1-\alpha)(1-g)\left(\frac{1}{2-\alpha-g}-\frac{(1-\alpha)g}{n(2-\alpha-g)^3}\right)+O\left(\frac{1}{n^2}\right)$

We will now formally introduce how to model the outcome of the predictions made by the proverbial chimps at the zoo. To that end, let g be the probability that a chimp at the zoo predicts the positive class. We assume that the chimps will all guess according to one and the same strategy out of the following three: they may toss a fair coin, throw a dice with n sizes, n_1 of which are labeled 'positive', or always guess the most common class (the mode), i.e.:

$$g^{\text{unif}} = \frac{1}{2}$$
$$g^{\text{prop}} = \alpha$$
$$g^{\text{mode}} = \begin{cases} 1 & \text{if } \alpha > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

Then, let X and Y be independently distributed random variables with binomial distributions $X \sim \text{Bin}(n_1, g)$ and $Y \sim \text{Bin}(n_2, g)$. Each evaluation metric is a random variable as well. Table 3 gives the expected confusion matrix. The third column of Table 2 shows how to compute the expected value of each performance metric. The proofs are provided in Appendix A.1. Five of the metrics are linear functions in the random variables X and Y, hence, it is trivial to compute their expected value. The expected value of the other six metrics take the form $\mathbb{E}[f(X, Y)]$ for a nonlinear, real-valued function f. If n is very small, these expectations could in theory be computed by the closed form expression

$$\mathbb{E}[f(X,Y)] = \sum_{l=0}^{n_1} \sum_{m=0}^{n_2} f(l,m)\mathbb{P}(X=l)\mathbb{P}(Y=m).$$

In practice, however, it might take a relatively long time to evaluate this expression if n gets large. So, unless n is small, the approximations given in Table 2 should be used. In addition, note that both X and Y have a (very small, but strictly) positive probability of being 0, in which case f might not be defined (e.g., for PPV we find 0/0). Mathematically, the correct way to deal with this is to exclude the event by conditioning the expectations on its complement. In practice, in particular for larger values of n, the obtained value will be very close to simply skipping the terms in the summation where f is not defined.

Table 3. Expected confusion matrix when randomly guessing the positive class with probability g.

		Predicted Positive	Negative	$\sum$
Actual	Positive	$n_1 g$	$n_1(1-g)$	n_1
	Negative	$n_2 g$	$n_2(1-g)$	n_2
	$\sum$	ng	$n(1-g)$	n

3 Balanced performance metrics

Some performance metrics are biased due to class imbalance. Balanced performance metrics are obtained by rewriting the imbalanced performance metrics

as a function of the imbalance coefficient $\delta = 2\alpha - 1$ and setting δ to 0, i.e. α to $\frac{1}{2}$ [6]. Table 4 shows the balanced metrics and an approximation of their expected value when randomly guessing the positive class with probability g. The derivations of the formulas in the third column can be found in Appendix A.2.

Table 4. Balanced performance metrics and their expected value when randomly guessing the positive class with probability g.

Metric Q^b	Definition [6]	$\mathbb{E}[Q^b(g)]$
ACC_c^b	$\frac{TPR_c+TNR_c}{2}$	$\frac{1}{2}$
TPR_c^b	TPR_c	g
TNR_c^b	TNR_c	$1-g$
PPV_c^b	$\frac{TPR_c}{TPR_c-TNR_c+1}$	$\frac{1}{2} + \frac{\delta(1-g)}{2n(1+\delta)(1-\delta)g} + O\left(\frac{1}{n^2}\right)$
NPV_c^b	$\frac{TNR_c}{TNR_c-TPR_c+1}$	$\frac{1}{2} - \frac{\delta g}{2n(1+\delta)(1-\delta)(1-g)} + O\left(\frac{1}{n^2}\right)$
AUC^b	$2AUC - 1$	0
J_c^b	J_c	0
MRK_c^b	$PPV_c^b + NPV_c^b - 1$	$\frac{\delta(1-2g)}{2n(1+\delta)(1-\delta)g(1-g)} + O\left(\frac{1}{n^2}\right)$
MCC_c^b	$\frac{TPR_c+TNR_c-1}{\sqrt{(TPR_c-TNR_c+1)(TNR_c-TPR_c+1)}}$	$\frac{\delta(1-2g)}{2n(1+\delta)(1-\delta)\sqrt{g(1-g)}} + O\left(\frac{1}{n^2}\right)$
PF_{1c}^b	$\frac{2TPR_c}{TPR_c-TNR_c+2}$	$2g\left(\frac{1}{1+2g} - \frac{2\left(1-\delta(1+2g)\right)(1-g)}{n(1+\delta)(1-\delta)(1+2g)^3}\right) + O\left(\frac{1}{n^2}\right)$
NF_{1c}^b	$\frac{2TNR_c}{TNR_c-TPR_c+2}$	$2(1-g)\left(\frac{1}{3-2g} - \frac{2\left(1+\delta(3-2g)\right)g}{n(1+\delta)(1-\delta)(3-2g)^3}\right) + O\left(\frac{1}{n^2}\right)$

4 Min-max normalization and optimization

After establishing $\mathbb{E}[Q(g)]$, min-max normalization can be applied to rescale each metric so that the proverbial chimps at the zoo score 0, on average:

$$Q_c^{mmn}(g) = \frac{Q_c - \mathbb{E}[Q(g)]}{1 - \mathbb{E}[Q(g)]}. \tag{1}$$

Note that $ACC^{mmn}(g)$ equals the Heidke Skill Score [4] or Cohen's κ [3] if g is set to $\frac{X+Y}{n}$. This g is, however, not a random guessing probability. The Heidke Skill Score min-max normalizes accuracy with the expected cell frequencies, which depend on the model.

Through K-fold cross validation or bootstrapping, we obtain the quality of K classifiers trained on different partitions or bootstrap samples of the data. We propose to first average Q_{kc}^{mmn} per cutoff to determine the overall optimal cutoff c^*, that is:

$$c^* = \arg\max_c \overline{Q}_c^{mmn}, \tag{2}$$

in which

$$\overline{Q}_c^{mmn} = \frac{1}{K}\sum_{k=1}^{K} Q_{kc}^{mmn}. \tag{3}$$

Then, we propose to use the distribution of $Q_k^{mmn}(c^*)$ as a proxy for the quality of the predictions when applied to unlabeled data for making statistical inference.

5 Example: normalized F_1 scores

Figure 1 shows the F_1 performance of a fictitious binary classifier that predicts 60% of the actual positive instances correctly ($TPR = 0.6$) and 80% of the actual negative cases ($TNR = 0.8$), using g^{unif} for normalization. Similar figures for accuracy and F_1 with g^{prop} can be found in Appendix B. When the test set is balanced ($\delta = 0$, i.e. $\alpha = 0.5$), this classifier scores $PF_1 = \frac{2}{3}$ and $NF_1 = \frac{8}{11}$. The more abundant the positive class relative to the negative class, the higher the classifier scores on PF_1 and the lower on NF_1 (thin red line in left panels). One solution to this sensitivity to class imbalance is to balance the metric (thin red lines in right panels) by correcting for the bias. The alternative we propose is to min-max normalize the metric (thick red line in left panels) by relating it to the expected value when randomly guessing the positive class with probability g (thin blue lines).

Two interesting observations can be made. First, data sets with a different imbalance coefficient can be compared. The classifier performs best at $\delta \approx -0.18$ where $PF_1^{mmn} \approx 0.34$, i.e. 34% from perfection relative to tossing a fair coin. A classifier with the same TPR and TNR in an application with a higher δ scores better on PF_1 (up to $\frac{3}{4}$), the same on PF_1^b but worse on PF_1^{mmn}. Second, metrics can be compared. Before min-max normalization, the classifier scores equally well on PF_1 and NF_1 at $\delta = \frac{1}{7}$ (small white points). After min-max normalization, however, the classifier scores equally well on PF_1^{mmn} and NF_1^{mmn} at $\delta \approx 0.5$ (large white points). Between $\frac{1}{7} < \delta < 0.5$, $PF_1 > NF_1$ but $PF_1^{mmn} < NF_1^{mmn}$.

Note that $\mathbb{E}[F_1]$ is sensitive to sample size n (see Tables 2 and 4), which becomes apparent when the metric is balanced and the sample is highly imbalanced (Fig. 1, right panel, blue line).

6 Conclusion

In this paper, we propose to rescale performance metrics through min-max normalization, where the minimum is set to the expected value when randomly guessing the positive class with probability g. It should be explicitly specified which expected value the algorithm is trying to defeat. Our proposed normalization yields different results than correcting for bias due to class imbalance [6] or balancing the sample [e.g. 2; 5]. The min-max normalized metrics allow for a better comparison between applications and between metrics. Moreover, we propose to use the distribution across test sets of a normalized metric at the overall optimal cutoff as performance metric for inferential statistics. Future research could

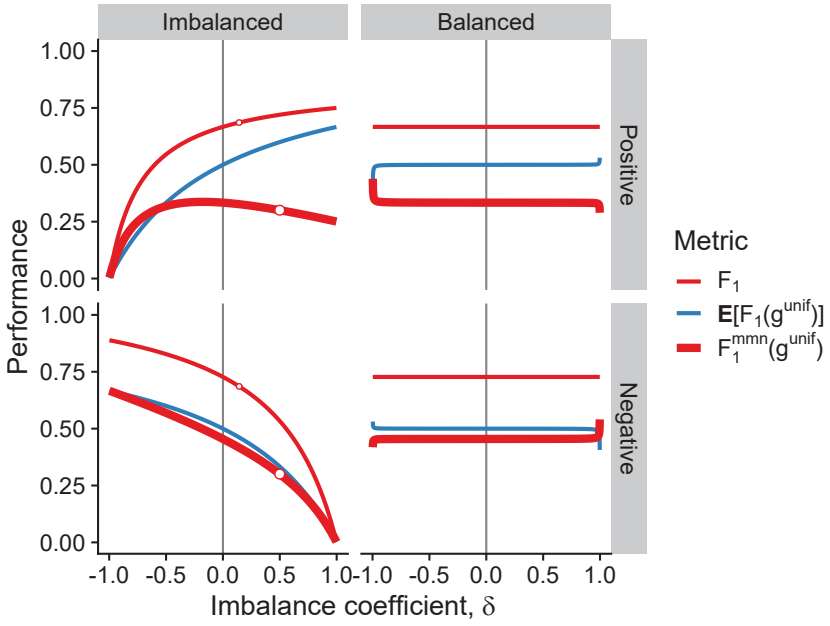


Fig. 1. Performance of a fictitious binary classifier in relation to imbalance coefficient δ. $g = 0.5$, $TPR = 0.6$, $TNR = 0.8$, $n = 1000$. White points show where positive and negative F_1 intersect.

focus on generalizing the results from binary classification to multi-class classification and regression, and on metrics that compare the predicted probability directly with the actual label, without a cutoff for constructing the confusion matrix.

Bibliography

[1] Baker, R., Brick, J.M., Bates, N.A., Battaglia, M., Couper, M.P., Dever, J.A., Gile, K.J., Tourangeau, R.: Summary report of the AAPOR task force on non-probability sampling. Journal of Survey Statistics and Methodology **1**, 90–143 (2013). https://doi.org/doi.org/10.1093/jssam/smt008

[2] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: Synthetic minority over-sampling technique. J. Artif. Int. Res. **16**(1), 321–357 (Jun 2002). https://doi.org/10.5555/1622407.1622416

[3] Cohen, J.: A coefficient of agreement for nominal scales. Educational and Psychological Measurement **20**(1), 37–46 (1960). https://doi.org/10.1177/001316446002000104

[4] Heidke, P.: Berechnung des erfolges und der gute der windstärkevorhersagen im sturmwarnungsdienst (measures of success and goodness of wind force forecasts by the gale-warning service). Geografiska Annaler **8**, 301–349 (1926). https://doi.org/10.1080/20014422.1926.11881138

[5] Krawczyk, B.: Learning from imbalanced data: open challenges and future directions. Progress in Artificial Intelligence **5**, 221–232 (2016). https://doi.org/10.1007/s13748-016-0094-0

[6] Luque, A., Carrasco, A., Martín, A., de las Heras, A.: The impact of class imbalance in classification performance metrics based on the binary confusion matrix. Pattern Recognition **91**, 216–231 (2019). https://doi.org/10.1016/j.patcog.2019.02.023

[7] Powers, D.M.W.: Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. Journal of Machine Learning Technologies **2**(1), 37–63 (2011)

[8] Wu, C., Thompson, M.E.: Non-probability survey samples. In: Sampling Theory and Practice, pp. 319–331. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-44246-0$_1$7

A Appendix - Proof of expected values

This appendix belongs to the paper by Burger & Meertens titled "The algorithm versus the chimps: On the minima of classifier performance metrics". It contains the proofs of the formulas provided in the third column of Table 2 and Table 4.

The key idea is to approximate an expectation of the form $\mathbb{E}[f(X,Y)]$, in which X and Y are random variables and in which f is an infinitely differentiable real-valued function, by inserting the Taylor series of f at $(\mathbb{E}[X], \mathbb{E}[Y])$. More specifically, let $x_0 = \mathbb{E}[X]$ and $y_0 = \mathbb{E}[Y]$ and consider the second-order Taylor series of f at (x_0, y_0):

$$\begin{aligned} f(x,y) \approx\ & f(x_0,y_0) + f_x(x_0,y_0)(x-x_0) + f_y(x_0,y_0)(y-y_0) \\ & + \frac{1}{2} f_{xx}(x_0,y_0)(x-x_0)^2 + \frac{1}{2} f_{yy}(x_0,y_0)(y-y_0)^2 \\ & + f_{xy}(x_0,y_0)(x-x_0)(y-y_0). \end{aligned} \tag{4}$$

We then approximate the expectation $\mathbb{E}[f(X,Y)]$ by taking the expectation of the right-hand side of the above equation. Then, assuming that X and Y are uncorrelated, we find

$$\mathbb{E}[f(X,Y)] \approx f(x_0,y_0) + \frac{1}{2} f_{xx}(x_0,y_0)\operatorname{Var}(X) + \frac{1}{2} f_{yy}(x_0,y_0)\operatorname{Var}(y). \tag{5}$$

In the proofs below we will specify the order of the approximation in terms of the size n of the test dataset.

In the remainder of the appendix, n_1 and n_2 are positive integers that sum up to n and $g \in (0,1)$ represents the probability that class 1 is predicted by the proverbial chimps at the zoo. Moreover, X will be a random variable distributed as $\mathrm{Bin}(n_1, g)$ and Y a random variable distributed as $\mathrm{Bin}(n_2, g)$. The random variables X and Y are assumed to be independent. The expectation and variance are given by

$$x_0 = \mathbb{E}[X] = n_1 g, \quad \operatorname{Var}(X) = n_1 g(1-g), \tag{6}$$

and

$$y_0 = \mathbb{E}[Y] = n_2 g, \quad \operatorname{Var}(Y) = n_2 g(1-g). \tag{7}$$

Finally, we will use the notation $\alpha = n_1/n$ (and hence $1-\alpha = n_2/n$) and $\delta = 2\alpha - 1 = (n_1 - n_2)/n$.

A.1 Expected value of imbalanced metrics

This appendix contains the derivations of the approximations of the expected values of the imbalanced performance metrics, as presented in Table 2 of the main text.

Expected Positive Predictive Value ($\mathbb{E}[\boldsymbol{PPV}]$)

The positive predictive value PPV can be written as $f(X,Y)$ with $f(x,y) = x/(x+y)$. Check that

$$f_{xx}(x,y) = \frac{-2y}{(x+y)^3}, \quad f_{yy}(x_0,y_0) = \frac{2x}{(x+y)^3}. \tag{8}$$

It follows that

$$\mathbb{E}[PPV] \approx \frac{n_1 g}{ng} - \frac{n_2 g}{(ng)^3} \cdot n_1 g(1-g) + \frac{n_1 g}{(ng)^3} \cdot n_2 g(1-g) = \frac{n_1}{n} = \alpha. \tag{9}$$

The higher order terms in the Taylor series of PPV are in $O(1/n^2)$. It can be shown by looking at the terms or order 3 in the Taylor series. Only f_{xxx} and f_{yyy} remain, which are both $O(1/n^3)$ when evaluated at (x_0, y_0), and the third central moment of both X and Y are $O(n)$.

Expected Negative Predictive Value ($\mathbb{E}[\boldsymbol{NPV}]$) The negative predictive value NPV can be viewed as the positive predictive value for the negative class, i.e., to compute NPV we first swap the roles of n_1 and n_2 and replace g by $1-g$ and then compute PPV. It follows that

$$\mathbb{E}[NPV] = 1 - \alpha + O\left(\frac{1}{n^2}\right). \tag{10}$$

Expected Area under the ROC curve ($\mathbb{E}[\boldsymbol{AUC}]$) If the threshold value c is equal to 1, then any coin toss prediction by the chimps is considered as tails, corresponding to the point $(0,0)$ on the ROC curve. Similarly, $c = 0$ corresponds to the point $(1,1)$ on the ROC curve. For any other value of the threshold value c, the predictions by the chimps do not depend on c, and thus we have $TPR_c = X/n_1$ and $1 - TNR_c = Y/n_2$, for any $0 < c < 1$. The ROC curve can be obtained by connected these three points, resulting in (the random variable!)

$$\begin{aligned} AUC &= \frac{1}{2}\frac{Y}{n_2}\frac{X}{n_1} + \left(1 - \frac{Y}{n_2}\right)\frac{X}{n_1} + \frac{1}{2}\left(1 - \frac{Y}{n_2}\right)\left(1 - \frac{X}{n_1}\right) \\ &= \frac{1}{2}\left(\frac{X}{n_1} + 1 - \frac{Y}{n_2}\right). \end{aligned} \tag{11}$$

It then follows that $\mathbb{E}[AUC] = \frac{1}{2}$.

Expected Matthews Correlation Coefficient ($\mathbb{E}[\boldsymbol{MCC}]$) The Matthews Correlation Coefficient (MCC) can be written as $f(X,Y)$ in which

$$f(x,y) = \frac{n_2 x - n_1 y}{\sqrt{n_1 n_2 (x+y)(n-x-y)}}. \tag{12}$$

Introducing the function $D(x,y) = n(x+y) - (x+y)^2$, the above simplifies to

$$f = (n_1 n_2)^{-\frac{1}{2}}(n_2 x - n_1 y)D^{-\frac{1}{2}}. \tag{13}$$

Both first order partial derivatives of D are equal to $n - 2(x+y)$. The identity $n_2 x_0 - n_1 y_0 = 0$ then implies that only the following term remains in $f_{xx}(x_0, y_0)$:

$$\begin{aligned} f_{xx}(x_0,y_0) &= 2 \cdot (n_1 n_2)^{-\frac{1}{2}} \cdot n_2 \cdot (-\tfrac{1}{2}) \cdot D^{-\frac{3}{2}}(x_0,y_0) \cdot (2 - n(x_0+y_0)) \\ &= \frac{-n_2(1-2g)}{n^2\sqrt{n_1 n_2 g^3(1-g)^3}}. \end{aligned} \tag{14}$$

Notice that $f_{xx}(x_0,y_0) = O(1/n^2)$, and hence $f_{xx}(x_0,y_0)\operatorname{Var}(X) = O(1/n)$. Similarly, we obtain

$$\begin{aligned} f_{yy}(x_0,y_0) &= 2 \cdot (n_1 n_2)^{-\frac{1}{2}} \cdot (-n_1) \cdot (-\tfrac{1}{2}) \cdot D^{-\frac{3}{2}}(x_0,y_0) \cdot (2 - n(x_0+y_0)) \\ &= \frac{n_1(1-2g)}{n^2\sqrt{n_1 n_2 g^3(1-g)^3}}. \end{aligned} \tag{15}$$

Interestingly, we have derived that

$$f_{yy}(x_0,y_0)\operatorname{Var}(Y) = -f_{xx}(x_0,y_0)\operatorname{Var}(X). \tag{16}$$

In particular, we have $f_{yy}(x_0,y_0)\operatorname{Var}(Y) = O(1/n)$. Finally, as $f(x_0,y_0) = 0$, we have shown that

$$\mathbb{E}[MCC] = 0 + O\left(\frac{1}{n^2}\right). \tag{17}$$

Expected Positive F_1 ($\mathbb{E}[PF_1]$) The positive F_1 score (PF_1) can be written as $f(X,Y)$ for $f(x,y) = 2x/(n_1 + x + y)$. Check that

$$f_{xx}(x,y) = \frac{-4(n_1+y)}{(n_1+x+y)^3}, \quad f_{yy}(x,y) = \frac{4x}{(n_1+x+y)^3}. \tag{18}$$

We leave it to the reader to check that $f_{xxx}(x_0,y_0) = O(1/n^3)$ and $f_{yyy}(x_0,y_0) = O(1/n^3)$. It follows that

$$\begin{aligned} \mathbb{E}[PF_1] &= \frac{2n_1 g}{n_1+ng} - \frac{2(n_1+n_2 g)}{(n_1+ng)^3} \cdot n_1 g(1-g) + \frac{2n_1 g}{(n_1+ng)^3} \cdot n_2 g(1-g) + O\left(\frac{1}{n^2}\right) \\ &= \frac{2n_1 g}{n_1+ng} - \frac{2n_1^2 g(1-g)}{(n_1+ng)^3} + O\left(\frac{1}{n^2}\right) \\ &= 2n_1 g\left(\frac{1}{n_1+ng} - \frac{n_1(1-g)}{(n_1+ng)^3}\right) + O\left(\frac{1}{n^2}\right) \\ &= 2\alpha g\left(\frac{1}{\alpha+g} - \frac{\alpha(1-g)}{n(\alpha+g)^3}\right) + O\left(\frac{1}{n^2}\right). \end{aligned} \tag{19}$$

Notice that $\mathbb{E}[PF_1(X,Y)] - PF_1(\mathbb{E}[X],\mathbb{E}[Y]) = O(1/n)$ and not $O(1/n^2)$. Moreover, the difference is strictly negative.

Expected Negative F_1 ($\mathbb{E}[NF_1]$) The approximation of the expectation of the negative F_1 score (NF_1) can be obtained from that of PF_1 by first swapping n_1 and n_2 and replacing g by $1-g$. In particular, we find

$$\mathbb{E}[NF_1] = 2(1-\alpha)(1-g)\left(\frac{1}{2-\alpha-g} - \frac{(1-\alpha)g}{n(2-\alpha-g)^3}\right) + O\left(\frac{1}{n^2}\right), \quad (20)$$

Again, notice that $\mathbb{E}[NF_1(X,Y)] - NF_1(\mathbb{E}[X],\mathbb{E}[Y]) = O(1/n)$ and not $O(1/n^2)$, and that the difference is strictly negative.

A.2 Expected value of balanced metrics

This appendix contains the derivations of the approximations of the expected values of the balanced performance metrics, as presented in Table 4 of the main text. The derivations are similar to those in Appendix A.1, although the outcomes are slightly different.

Expected balanced Positive Predictive Value ($\mathbb{E}[PPV^b]$) The balanced positive predictive value PPV^b can be written as $f(X,Y)$ with $f(x,y) = (x/n_1)/(x/n_1 + y/n_2)$. Check that

$$f_{xx}(x,y) = \frac{-2y/n_2}{n_1^2(x/n_1+y/n_2)^3}, \quad f_{yy}(x_0,y_0) = \frac{2x/n_1}{n_2^2(x/n_1+y/n_2)^3}. \quad (21)$$

It follows that

$$\begin{aligned}\mathbb{E}[PPV^b] &= \frac{1}{2} - \frac{n_1 g^2(1-g)}{n_1^2(2g)^3} + \frac{n_2 g^2(1-g)}{n_2^2(2g)^3} + O\left(\frac{1}{n^2}\right)\\ &= \frac{1}{2} + \frac{(n_1-n_2)(1-g)}{8n_1n_2g} + O\left(\frac{1}{n^2}\right)\\ &= \frac{1}{2} + \frac{\delta(1-g)}{2n(1+\delta)(1-\delta)g} + O\left(\frac{1}{n^2}\right). \quad (22)\end{aligned}$$

Observe that $\mathbb{E}[PPV^b(X,Y)] - PPV^b(\mathbb{E}[X],\mathbb{E}[Y]) = O(1/n)$, in contrast to $\mathbb{E}[PPV(X,Y)] - PPV(\mathbb{E}[X],\mathbb{E}[Y]) = O(1/n^2)$. However, the absolute value of the term of order $1/n$ can be bounded from above by $(1-g)/(8g)$.

Expected balanced Negative Predictive Value ($\mathbb{E}[NPV^b]$) The balanced negative predictive value NPV^b can be viewed as the balanced positive predictive value for the negative class, i.e., to compute NPV^b we first swap the roles of n_1 and n_2 and replace g by $1-g$ and then compute PPV^b. It follows that

$$\mathbb{E}[NPV^b] = \frac{1}{2} - \frac{\delta g}{2n(1+\delta)(1-\delta)(1-g)} + O\left(\frac{1}{n^2}\right). \quad (23)$$

Again, observe that $\mathbb{E}[NPV^b(X,Y)] - NPV^b(\mathbb{E}[X],\mathbb{E}[Y]) = O(1/n)$, in contrast to $\mathbb{E}[NPV(X,Y)] - NPV(\mathbb{E}[X],\mathbb{E}[Y]) = O(1/n^2)$. However, the absolute value of the term of order $1/n$ can be bounded from above by $g/(8(1-g))$.

Expected balanced Markedness ($\mathbb{E}[\boldsymbol{MRK^b}]$) The expectation of the balanced markedness (MRK^b) can be approximated as follows:

$$\begin{aligned}\mathbb{E}[MRK^b] &= \mathbb{E}[PPV^b] + \mathbb{E}[NPV^b] - 1 \\ &= \frac{1}{2} + \frac{\delta(1-g)}{2n(1+\delta)(1-\delta)g} + \frac{1}{2} - \frac{\delta g}{2n(1+\delta)(1-\delta)(1-g)} - 1 + O\left(\frac{1}{n^2}\right) \\ &= \frac{\delta(1-2g)}{2n(1+\delta)(1-\delta)g(1-g)} + O\left(\frac{1}{n^2}\right). \end{aligned} \tag{24}$$

It shows that $\mathbb{E}[MRK^b(X,Y)] - MRK^b(\mathbb{E}[X],\mathbb{E}[Y]) = O(1/n)$, in contrast to $\mathbb{E}[MRK(X,Y)] - MRK(\mathbb{E}[X],\mathbb{E}[Y]) = O(1/n^2)$. However, if $g = \frac{1}{2}$, then the term of order $1/n$ is zero. If $g \neq \frac{1}{2}$, then the absolute value of the term of order $1/n$ can be bounded from above by $(1-2g)/(8g(1-g))$.

Expected balanced Matthews Correlation Coefficient ($\mathbb{E}[\boldsymbol{MCC^b}]$) The balanced Matthews Correlation Coefficient (MCC^b) can be written as $f(X,Y)$ in which

$$f(x,y) = \frac{x/n_1 - y/n_2}{\sqrt{(x/n_1 + y/n_2)(2 - x/n_1 - y/n_1)}}. \tag{25}$$

Introducing the function $D(x,y) = 2(x/n_1 + y/n_2) - (x/n_1 + y/n_2)^2$, the above simplifies to

$$f = (x/n_1 - y/n_2)D^{-\frac{1}{2}}. \tag{26}$$

The first order partial derivatives of D are equal to $2/n_1 \cdot (1 - x/n_1 - y/n_2)$. The identity $x_0/n_1 - y_0/n_2 = 0$ then implies that only the following term remains in $f_{xx}(x_0,y_0)$:

$$\begin{aligned} f_{xx}(x_0,y_0) &= 2 \cdot (1/n_1) \cdot (-\tfrac{1}{2}) \cdot D^{-\frac{3}{2}}(x_0,y_0) \cdot 2/n_1 \cdot (1 - x_0/n_1 - y_0/n_2)) \\ &= \frac{-(1-2g)}{4n_1^2\sqrt{g^3(1-g)^3}} \end{aligned} \tag{27}$$

Similarly, we obtain

$$\begin{aligned} f_{yy}(x_0,y_0) &= 2 \cdot (-1/n_2) \cdot (-\tfrac{1}{2}) \cdot D^{-\frac{3}{2}}(x_0,y_0) \cdot 2/n_2 \cdot (1 - x_0/n_1 - y_0/n_2)) \\ &= \frac{(1-2g)}{4n_2^2\sqrt{g^3(1-g)^3}}. \end{aligned} \tag{28}$$

Finally, as $f(x_0,y_0) = 0$, it follows that

$$\begin{aligned} \mathbb{E}[MCC^b] &= \frac{-(1-2g)n_1 g(1-g)}{8n_1^2\sqrt{g^3(1-g)^3}} + \frac{(1-2g)n_2 g(1-g)}{8n_2^2\sqrt{g^3(1-g)^3}} + O\left(\frac{1}{n^2}\right) \\ &= \frac{(n_1-n_2)(1-2g)}{8n_1 n_2\sqrt{g(1-g)}} + O\left(\frac{1}{n^2}\right) \\ &= \frac{\delta(1-2g)}{2n(1+\delta)(1-\delta)\sqrt{g(1-g)}} + O\left(\frac{1}{n^2}\right). \end{aligned} \tag{29}$$

Once again, observe that $\mathbb{E}[MCC^b(X,Y)] - MCC^b(\mathbb{E}[X],\mathbb{E}[Y]) = O(1/n)$, in contrast to $\mathbb{E}[MCC(X,Y)] - MCC(\mathbb{E}[X],\mathbb{E}[Y]) = O(1/n^2)$. However, if $g = \frac{1}{2}$, then the term of order $1/n$ is zero. If $g \neq \frac{1}{2}$, then the absolute value of the term of order $1/n$ can be bounded from above by $(1-2g)/(8\sqrt{g(1-g)})$.

Expected balanced Positive F_1 ($\mathbb{E}[PF_1^b]$) The balanced positive F_1 score (PF_1^b) can be written as $f(X,Y)$ for $f(x,y) = (2x/n_1)/(x/n_1 + y/n_2 + 1)$. Check that

$$f_{xx}(x,y) = \frac{-4(y/n_2+1)}{n_1^2(x/n_1+y/n_2+1)^3}, \quad f_{yy}(x,y) = \frac{4x/n_1}{n_2^2(x/n_1+y/n_2+1)^3}. \quad (30)$$

It follows that

$$\begin{aligned}
\mathbb{E}[PF_1^b] &= \frac{2g}{1+2g} - \frac{2n_1(1+g)g(1-g)}{n_1^2(1+2g)^3} + \frac{2n_2g^2(1-g)}{n_2^2(1+2g)^3} + O\left(\frac{1}{n^2}\right) \\
&= 2g\left(\frac{1}{1+2g} - \frac{\Big(n_2 - (n_1-n_2)g\Big)(1-g)}{n_1n_2(1+2g)^3}\right) + O\left(\frac{1}{n^2}\right) \\
&= 2g\left(\frac{1}{1+2g} - \frac{2\Big(1-\delta(1+2g)\Big)(1-g)}{n(1+\delta)(1-\delta)(1+2g)^3}\right) + O\left(\frac{1}{n^2}\right) \qquad (31)
\end{aligned}$$

The term of order $1/n$ is bounded from above by $2g^2(1-g)/(2g+1)^3$, which is at most $8/243 \approx 0.033$ at $g = 2/5$. Moreover, it is bounded from below by $-2g(1-g^2)/(2g+1)^3$, which is at least $-4/243 \cdot (7\sqrt{7}-10) \approx -0.14$ at $g = (\sqrt{7}-2)/3 \approx 0.22$.

Expected balanced Negative F_1 ($\mathbb{E}[NF_1^b]$) The approximation of the expectation of the balanced negative F_1 score (NF_1^b) can be obtained from that of PF_1^b by first swapping n_1 and n_2 and replacing g by $1-g$. In particular, we find

$$\mathbb{E}[NF_1^b] = 2(1-g)\left(\frac{1}{3-2g} - \frac{2\Big(1+\delta(3-2g)\Big)g}{n(1+\delta)(1-\delta)(3-2g)^3}\right) + O\left(\frac{1}{n^2}\right), \quad (32)$$

The term of order $1/n$ is bounded from above by $2g(1-g)^2)/(2(1-g)+1)^3$, which is at most $8/243 \approx 0.044$ at $g = 3/5$, and bounded from below by $-2(1-g)(1-(1-g)^2)/(2(1-g)+1)^3$, which is at least $-4/243 \cdot (7\sqrt{7}-10) \approx -0.14$ at $g = (5-\sqrt{7})/3 \approx 0.78$.

B Appendix - Performance of fictitious binary classifier

This appendix shows how a fictitious binary classifier with $TPR = 0.6$ and $TNR = 0.8$ performs on accuracy (Figs. 2 and 3) and F_1 (Figs. 1 and 4 when

it is min-max normalized with the expected value when randomly guessing the positive class with probability $g = 0.5$ (Figs. 2 and 1) or $g = \alpha$ (Figs. 3 and 4), as a function of imbalance coefficient δ. Shown are imbalanced metrics (left panels) and balanced metrics (right panels), which have been corrected for bias due to class imbalance.

When $g^{unif} = \frac{1}{2}$ is chosen as control, $\mathbb{E}[ACC(g^{prop})] = \frac{1}{2}$ (Fig. 2, left panel, blue line). When $g^{prop} = \alpha$ is chosen as control, $\mathbb{E}[ACC(g^{prop})]$ is a quadratic function (Fig. 3, left panel, blue line; see Table 2). As a result, the classifier is outperformed ($ACC^{mmn}(g^{prop}) < 0$) by this strategy when imbalance is large (here when $\delta < \frac{-1-\sqrt{41}}{10} \approx -0.74$ or $\delta > \frac{-1+\sqrt{41}}{10} \approx 0.54$).

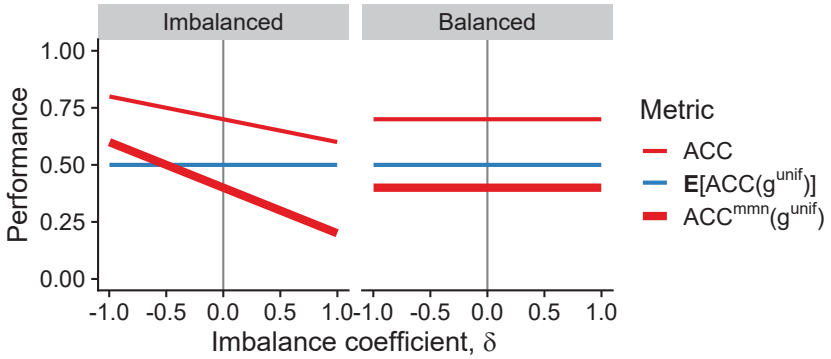


Fig. 2. Accuracy of a fictitious binary classifier in relation to imbalance coefficient δ. $g = 0.5$, $TPR = 0.6$, $TNR = 0.8$, $n = 1000$.

Before min-max normalization, the classifier scores equally well on PF_1 and NF_1 at $\delta = \frac{1}{7}$ (Fig. 4, small white points). After min-max normalization using g^{prop}, however, the classifier scores equally well on PF_1^{mmn} and NF_1^{mmn} at $\delta = -\frac{1}{3}$ (large white points). For $\delta < -\frac{1}{3}$ and $\delta > \frac{1}{7}$, the regular F_1 and normalized $F_1^{mmn}(g^{prop})$ disagree on whether the model performs better on the positive or the negative class.

By definition, the balanced F_1 is insensitive to class imbalance. After min-max normalization with g^{prop}, however, it is sensitive again to class imbalance. The larger the imbalance coefficient, the lower the classifier scores on $PF_1^{mmn,b}(g^{prop})$ and the higher on $NF_1^{mmn,b}(g^{prop})$ (Fig. 4, right panels).

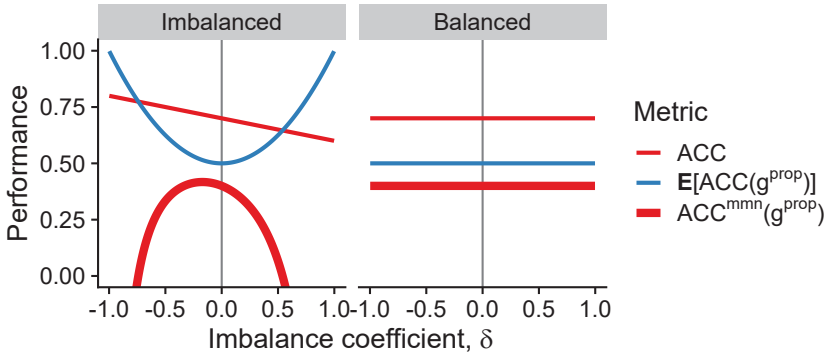


Fig. 3. Accuracy of a fictitious binary classifier in relation to imbalance coefficient δ. $g = \alpha$, $TPR = 0.6$, $TNR = 0.8$, $n = 1000$.

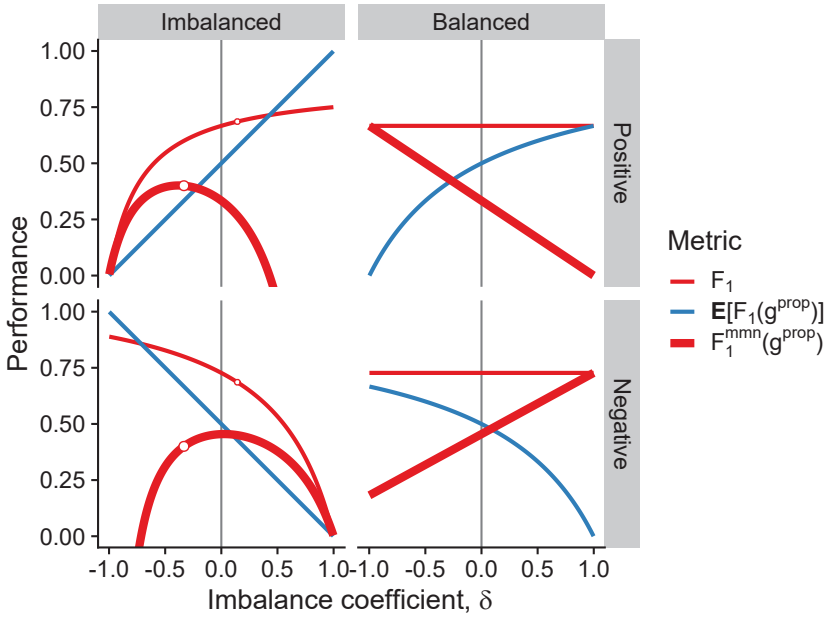


Fig. 4. F_1 of a fictitious binary classifier in relation to imbalance coefficient δ. $g = \alpha$, $TPR = 0.6$, $TNR = 0.8$, $n = 1000$. White points show where positive and negative F_1 intersect.

Philéas: Anomaly Detection for IoT Monitoring

Alberto Franzin[1][0000−0002−4066−0375]*, Raphaël Gyory[1]*, Jean-Charles Nadé[2]*, Guillaume Aubert[2], Georges Klenkle[2], and Hugues Bersini[1]

[1] IRIDIA-CoDE, Université Libre de Bruxelles, Brussels, Belgium
{afranzin,raphael.gyory,bersini}@ulb.ac.be
[2] Degetel Belgium, Brussels, Belgium
jcnade@gmail.com,gaubert@degetel.com,gklenkle@eugeka.com

Abstract. A growing number of private companies and public administrations is adopting Internet of Things (IoT) technologies to monitor resources, spaces, activities and events. Ensuring the required levels of quality of service and security is a key aspect in managing a service based on IoT. We introduce Philéas, a joint project between Degetel Belgium and the IRIDIA laboratory of the Université Libre de Bruxelles to develop a framework to analyze the activity of IoT systems and to identify possible issues via anomaly detection. In this paper we describe our framework, and we present as a demonstration two real cases that have been tackled using this framework.

Keywords: Anomaly Detection · Industrial applications · Internet of Things · Machine Learning · Quality of Service · Security.

1 Introduction

Internet of Things (IoT) devices are increasingly deployed to address a variety of real world tasks. The umbrella term Internet of Things encompasses a variety of technologies that collect, elaborate and transfer data over a network to other devices and servers in an automated and pervasive fashion [2]. They find application in a variety of domains from smart homes [42, 45] to smart cities [35, 50], from agriculture [18, 47] to manufacturing [23, 34], from healthcare [3, 29, 49] to transportation and mobility [17, 43], and many more. Due to the potential impact on society, public administrations also take a great interest in these technologies, from local to international scale [22, 37]. Though estimates on the actual number of devices vary wildly, there is strong consensus on the fact that the exponential growth will only continue in the next years, reaching the tens of billions of devices in the very near future [12, 36].

The growth of IoT technologies is however not without concerns. The complexity and scale of their applications, their ubiquity and interconnection, the heterogeneity and limited capabilities of technologies involved, the collection and treatment of personal and/or sensitive data are all factors that pose serious

*A.F., R.G. and J-C.N. contributed equally to this work.

challenges regarding energy management, security and privacy [15, 19]. IoT technologies offer several vulnerabilities for malicious actors to exploit or, simply, for faults and issues to happen [7]. The timely detection of such issues plays a key role in the management of an IoT network and application.

Big Data technologies and Artificial Intelligence (AI), in particular Machine Learning (ML), techniques are instrumental in assisting human operators in the monitoring, analysis and resolution of such issues [21, 26, 28, 31, 32, 40, 46, 52]. Devices collect a huge amount of data that is sent to other devices and central servers, where it is stored to be processed. While the data collected can be analyzed for the specific application, the metadata consisting in the message headers is useful to understand the state of the network and application. In large amounts of data, patterns of behaviour are likely to emerge, and deviations from them can be an indication of potential problems.

In this work we introduce Philéas, a framework conceived to assist managers and operators of IoT networks and applications in the analysis of IoT data. In particular, while in this project we have analyzed several different cases and applications, our main focus is the detection of anomalies in the metadata received from the devices. Philéas focuses on the analysis of anomalies from a centralized, application perspective, as independent as possible from the technical details of the devices and the network protocol, and not relying on external information about the network status. Philéas is a joint project between Degetel,[1] a consulting and services group specialized in digital transformation in France and Belgium with 15 years of experience in the IoT domain, and IRIDIA, the AI laboratory of the Université Libre de Bruxelles (ULB). Philéas answers specific market demands from the clients of Degetel regarding the management and securitization of IoT infrastructures. The project is funded by Innoviris, the institute of the Brussels Capital Region for technological innovation, with the goal of transferring academic knowledge into the industrial domain, and aims at exploring advanced AI solutions that can accompany more traditional approaches.

In the following Section we review the context of this project, the challenges in IoT systems that we address within this project and some existing relevant approaches. In Section 3 we describe the Philéas framework, including its infrastructure and the algorithms we implemented. In Section 4 we present two real world cases tackled in this project, before concluding in Section 5.

2 Background and related work

2.1 Internet of Things

Internet of Things (IoT) is a set of technologies based on uniquely identifiable devices capable of communicating with each other over a network without human interaction [2, 7]. Though the definition is rather broad, with IoT we usually refer to low power embedded devices with limited computational capabilities devoted to one task, or a specific set of tasks. A common characteristic of IoT devices

[1] https://www.degetel.com

is their pervasivity, that is, the possibility of deploying them in countless places and applications. They enable the collection of huge amounts of data, which is usually collected and analyzed. IoT devices include sensors and smart meters that measure one single value or event (e.g. temperature, humidity, the opening of a door, the failure of a mechanical component in an industrial machine) and transmit it to a central server. The use of IoT in the industry is at the base of the so-called fourth industrial revolution [23]. But in IoT we can also include vehicles capable of communicating with other vehicles and the environment (e.g. road infrastructure) [20]. IoT is also progressively entering private homes, with home automation and intelligent appliances [45].

From an economic perspective, IoT technologies create a new market, whose actors are device manufacturers, network and service providers, and application developers. Public administrations also play a role in this: for example, in Brussels public entities provide support for a smart city initiative[2] and for companies and start-ups to bring AI and IoT solutions to the market. There is also great public interest in the next generation of IoT networks, based on 5G.

Alongside with the many opportunities, the deployment of IoT solutions presents several technical challenges, from the non-interoperability of solutions, to device obsolescence, to the computational challenges of big data analysis [4]. But also the pervasivity of the devices, the possible threats to privacy and security and their implications, even at international level, are key concerns for developers, institutions and regulators [14, 16, 30].

Technical specifications of IoT devices and their interconnection encompass the full stack from the design of the electronic components of a device to the platform and application. Issues can happen at various levels of the IoT system – physical, network, application. Among the several problems that affect IoT systems, here we review the ones that concern the scope of Philéas, and we compile the following list from the management and application perspective, that is, from the central administration of the system.

2.2 Issues and challenges in IoT system administration

Device-related issues A very common situation, especially when using technologies that are inexpensive or with low capabilities, is the malfunctioning of a device, which could fail in some of its parts (e.g. measuring a wrong value, being unable to send or receive messages) or stop working altogether.

Network failure Similarly, a system can fail at the network level, e.g. because of a malfunctioning gateway. Packets can also be lost simply because of a poor network status, caused for example by bad weather conditions. In this case we typically observe a deviation from the usual patterns for a group of devices belonging to the same network, or connected to each other.

[2] https://smartcity.brussels/

Malicious action IoT systems can be the target of criminals with the goal of stealing information, or simply disrupt a service to cause financial damage. Several kinds of attacks are possible on an IoT system, for example, Distributed Denial of Service (DDoS) attacks can compromise a gateway, while the lack of end-to-end message integrity check could be exploited to alter the payload [39].

However, from the perspective of this project, the effect of malicious action results in issues that affect the system at the device or network level, or both. In fact, for both network and device failure a mere log analysis is usually insufficient to distinguish the causes of the failure, whether accidental or caused by malicious actors, and additional knowledge is required to establish the causes of the issue.

System heterogeneity The huge variety in the technologies available makes it very difficult to provide generalized solutions, even for the same kind of task. As an example, and the case that concerns Philéas the most, there are several communication protocols that can be implemented to transmit messages between devices in a network, many of which are proprietary.

2.3 Anomaly detection for IoT system management

The complexity of IoT systems is a perfect application for AI technologies and, in particular, data mining and ML techniques that can be used to process the vast amount of data and metadata collected [26, 28, 46, 52]. A complete review is beyond the scope of this project and of this work; here we limit our discussion to an overview of the techniques that have been applied to monitor the state of IoT systems from the data collected, notably anomaly detection.

An *anomaly* (or *outlier*) is an observation, or a group of observations, that exhibits two characteristics: it differs significantly from the majority of other observations, and it appears rarely in the dataset [5, 25, 27]. Anomalies can take many different forms: the simplest way to define what an anomaly is is therefore to define what *normal* observations (*inliers*) are, and to mark as anomalies all the observations that cannot be considered inliers. The nature of the deviation depends on the particular context and application. We can search for observations that deviate from the regular behaviour in the entire dataset; in this case we are considering *global* anomalies. But anomalies can also occur with respect to a subset of the data, and in this case we identify them as *local* anomalies.

Clustering and neighbourhood-based methods Clustering often is the first step taken to make sense of the data collected. By associating related observations, we can identify groups of devices that exhibit similar behaviour, for some suitable definition of similarity that takes into account relevant features. There are however many possible similarity criteria, and many clustering techniques available, each one possibly entailing different outcomes. Popular techniques include centroid-based algorithms, such as the k-means algorithms, where some observations are chosen as representatives (*centroids*) of the clusters they belong, and the remaining observations are associated to the closest centroid. Another approach is based on *density*, where a cluster is composed by points that have a

minimum amount of neighbouring points under a certain distance; in algorithms from this class, such as DBSCAN, sparse points can be considered outliers. For thorough reviews of clustering algorithms, we refer to [1, 48].

In the case of IoT log analysis, to cluster similar observations we usually need to define a distance function based on a subset of the packet fields. For example, we can group observations by the behaviour they describe, e.g. the number of packets sent by each device in a certain interval of time. But we can also analyze the aggregate of the packets sent by one device, or the devices of a specific client or a certain geographic area.

A notion of proximity between observations is also at the base of the Local Outlier Factor (LOF) algorithm, an anomaly detection technique based on the notion of *local density*, that is, how close each point is to its k neighbours [8]. In a nutshell, LOF classifies as anomalies data points whose local density differs from the local density of its neighbours. LOF is a generic technique that can be applied to various tasks for which we can define a distance between observations.

One-class learning Another approach to anomaly detection is to have a model learn only the "normal" behaviour; this corresponds to a classification task with a single target class. Anomalies are then the observations for which the model performs poorly. Techniques in this family include one-class Random Forests [24] and one-class Support Vector Machines [11]. Isolation Forests exploits the low frequency of outliers to isolate them in leaves of decision trees [33].

A family of artificial neural networks called *autoencoders* is another effective approach to anomaly detection [51]. Autoencoders perform two subsequent actions: first they map (encode) the input to a reduced space of neurons, in order to approximate the input; then they try to re-generate (decode) the input from this approximation. During the training phase they effectively learn a noise-free version of the original model, hence, in general, the reconstruction error will be smaller for observations that match the input model relatively well, rather than for observations that deviate significantly from the majority of the other points in the dataset; the first ones can therefore be considered inliers, while the latter will be identified as outliers.

Time series analysis As devices send packets to the central server either periodically or based on events, the data collected can often be modeled as multivariate time series. A multivariate time series is an ordered set of k-dimensional vectors $\mathbf{X} = \{\mathbf{x}_t\}_{t \in T}$ where each vector $\mathbf{x}_t = \{x_t^1, x_t^2, \ldots, x_t^k\}$ contains the values observed at time t. In our case, the values are the values of the different features we receive, or that we are interested to monitor in the given situation. Anomalies in time series can take different forms. We can look for a point or a sequence of points that deviates from the rest of the points in the time series (*point outliers* and *subsequence outliers*), or for a time series that exhibits a different behaviour from other time series in one or more features (*outlier time series*). Anomaly detection in time series is a rich and active field of research, thanks also to the huge importance of this task for the industry [41], and we refer to [6, 10] for detailed reviews of anomaly detection techniques for time series.

Batch vs real time analysis vs prediction Depending on the context and the specific applications, there are two possible ways of looking for anomalies in log data, batch analysis and real time analysis. Batch analysis processes data about past event, and is performed periodically or occasionally to discover anomalies occurred in the past, e.g. in the context of forensic analysis. Real time analysis is instead applied to the data as it arrives, or in small batches of recent data, and is continuously performed to ensure that potential problems are immediately spotted and taken care of. When a machine learning model is trained on the data available, it can be used to predict the future status of the IoT system, for example the reception of a packet, or the failure of a device or a network. Prediction is always applied to one single instance of the desired target.

Domain expertise Domain expertise is crucial to properly understand and evaluate the results of a data analysis, as the data alone is often not sufficient to fully understand a situation. In particular, in our applications the client needs to be involved in the process and has to analyze the outcome.

3 The Philéas framework

3.1 Scope

In Philéas we implement algorithms to detect anomalies in IoT metadata, processing logs of messages from different sources both in batch and in real time. Philéas provides a framework that can be used not only as stand-alone software, but also to develop specific solutions for different clients. Key elements in the design of the framework are the separation between the data and the AI algorithms and at the identification of common features in the various data sources. Therefore, while we can provide algorithms tailored for specific cases, in general we favour general techniques that can be applied in a variety of contexts. A specific application is then instantiated for each client, selecting the infrastructure and the algorithms that best serve the specific needs for the tasks required. The results of the anomaly detection task are meant to accompany domain expert analyses, to obtain meaningful insights about the status of the IoT system.

3.2 Network protocols

We consider two of the most common options for the communication between the devices and the central infrastructure, Sigfox [53] and LoRaWAN [13, 44]. Both are proprietary technologies, developed in France and available mostly in Europe. They operate in the ISM (Industrial, Scientific and Medical) band, at $867 - 869$ MHz in Europe. The protocols have similar architectures: devices transmit packages to gateway nodes, which in turn communicate with the network server, devoted to manage the data for the various applications. The devices are not associated to a specific gateway, but rather initiate a communication by looking for an available gateway, and continue communicating with a responding one.

Sigfox is a protocol designed to be simple and robust to interference. Its packets contain only nine fields in total including the payload, with additional information about the client, and only the device ID, transmission time and RSSI of the network at the transmission as metadata usable for analysis on the receiving end. It uses an asymmetric link for transmission and reception, so it is a good choice in case of network of sensors that transmit infrequent data (e.g. temperature sensors).

LoRaWAN (Long Range Wide Area Network) is the medium access control and network layer protocol defined in the LoRa standard, designed for long range, low power connectivity: a device can function for over eight years before having to replace its battery. A LoRaWAN packet contains several metadata fields additionally to the payload. These fields include information about the gateway and the antenna, and for both uplink and downlink. LoRaWAN uses a symmetric link, so it is a better choice in case of bidirectional communication.

For more technical details about the specifications of these two protocols, we refer to their official documentations. At the moment we do not consider alternative protocols such as 5G, GPRS or NB-IoT, but the Philéas application is designed to be possibly extended to work with different packet formats.

To provide solutions as general and reusable as possible, we base our analyses on the common fields of both the Sigfox and LoRaWAN packet formats. For Sigfox, the relevant fields include, aside from the payload, the device ID, the client ID, the timestamp, and the RSSI (Received Signal Strength Indicator), a measure of the power of the radio signal, and thus on the quality of the network at the moment of the transmission. Analyzing the content of these fields can already provide several insights on the status of the IoT system. In addition to this, LoRaWAN provides several other information about the network, such as the uplink and downlink gateways, and the connection status, that can be used for deeper analyses when needed.

3.3 Algorithms

Simple rule-based and statistical analyses are very effective in several scenarios. For example, if a device did not send a message in the last x hours, or sent $y\%$ more (or less) messages than the other devices in its network, it may considered a problem. The values of x and y are to be determined by the specific application, either as fixed values provided by the client or after a preliminary data analysis. We can also compare the current behaviour of a device with its past behaviour, to observe whether it changed significantly, according to some threshold values.

In IoT networks the chances of losing a packet are comparatively high with respect to other network technologies, without this necessarily being related to actual problems. Hence, point outliers in metadata, especially multivariate (a single packet received or lost) are at high risk of being false positives. We therefore focus on subsequence outliers and outlier time series, as indications of possible persistent problems.

We use the k-means clustering to identify devices based on their transmission behaviour. For the same task in a big data context we also implement a MapRe-

duce version of the k-center clustering algorithm [9]. The distance function may depend on the specific case, protocol and application, and can therefore be defined with the user. We also use the Local Outlier Factor algorithm to identify devices that have an abnormal frequency of messages, or an abnormal amount of messages received.

While statistical and clustering methods cover most of the needs in our practical cases to characterize, respectively, individual and group behaviour of the devices. However, one of the goals of this project was to investigate more advanced machine learning models for analysis of IoT metadata, for advanced analyses of large batches of data, but also to replace manually-crafted rules and to predict future issues at the device level. We implement autoencoder neural networks, whose hyperparameters are to be set for each specific case.

3.4 Infrastructure note: exchanged place with Algorithms

The database used depends on the amount of data to be collected for each client. We have the option of using the Hadoop infrastructure for managing big data, and PostgreSQL and MongoDB as databases otherwise. Streaming data can be collected using Kafka.

The algorithms of Section 3.3 are built on top of the common Python stack of scientific libraries, based on Spark (Pyspark), Pandas and Scikit-learn for data analysis, feature augmentation and machine learning tasks. We use TensorFlow for implementing deep learning solutions, and Spark for big data analysis. Whenever possible, we use the algorithms available in the Python libraries; we however implement custom algorithms for statistical and time series analyses.

The interface for the application is built using Django and node.js. Communication with the backend is handled by REST services.

4 Use cases

Here we present two examples of issues tackled using the Philéas framework, to showcase the set of algorithms we have currently available. As they refer to specific situations of Degetel clients, the data and some of the specific details are covered by non-disclosure agreements, and we will thus omit from the following presentation any detail that may identify situations, clients or any other party involved unless specifically authorized. We can however present the computational problems, and the approaches we implemented to tackle them.

4.1 Quality of Service

The first case is about Shayp[3], a Brussels-based startup that deploys IoT water telemetry sensors to monitor water consumption in indoor locations. The sensors measure the amount of water used and transmit this value to the central server

[3] http://www.shayp.com

using Sigfox packets, with a frequency of one packet every hour. Some packets are lost, either singularly or in bursts: this normally happens due to poor network conditions. Sometimes packets from a certain device may disappear completely, in case of a faulty device or external intervention (e.g. a device is misplaced after being accidentally hit). To avoid too many false positives, we do not consider a single lost packet as an anomaly; in fact, this can situation can happen for several reasons, and it is not considered problematic in itself. However, two or more consecutive expected packets lost are considered as an anomaly to note.

The dataset for the analysis we report includes anonymized logs of 500 devices for one year of activity, each observation corresponding to one Sigfox packet received ($\sim$ 2.6M packets in total). No information available regarding users is available, and the payload is encrypted. Given the simplicity of the transmission protocol, the relevant information in each packet is only the device ID, the RSSI of the network and the timestamp of the message. The expected periodicity allows us to detect the loss of one or more packets by measuring the time elapsed between two consecutive packets received from the same device.

Monitoring the status of the network and of the devices can be done, in large part, using simple time series and statistical analyses, analyzing the time of each message, and the associated RSSI. We implement rules to detect devices with an anomalous behaviour, with respect to both the other devices in the network and the device expected behaviour.

We use this case also to describe our autoencoder approach to track lost packets. The relevant information for each message is: (i) the device ID, (ii) the RSSI value measured, and (iii) the elapsed time since the previous packet from the same device. Starting from these information, for each device we build two sequences, R_n with the n last RSSI values measured for the device (normalized in the $[0, 1]$ interval, relative to the entire dataset), and T_n, the (normalized) elapsed time between each of the last n packets received. For the autoencoder to learn the "normal" behaviour, we include in the training set only data that corresponds to packages that have at most one packet lost among its predecessors.

The input features for the autoencoder are the two sequences R_n and T_n. The autoencoder has a symmetrical architecture with an input and an output layer of $2n$ nodes, a first and last hidden layer of n nodes and a third and fourth hidden layer of $\lceil n/2 \rceil$ nodes. In our experiments we used $n = 5$, for a total of ten input features. More precisely, the network architecture is the following one:

input layer 10 nodes with ReLu activation;
first hidden layer fully connected, 5 nodes, ReLu with ℓ_1 regularization;
second hidden layer fully connected, 3 nodes, ReLu activation;
third hidden layer fully connected, 3 nodes, ReLu activation;
fourth hidden layer fully connected, 5 nodes, ReLu activation;
output layer fully connected, 10 nodes, ReLu activation.

The autoencoder then computes the reconstruction error of its input. Sequences corresponding to packets considered having normal behaviour have a lower reconstruction error than packets belonging to sequences where many previous packets have been lost. We can thus fix a threshold for the reconstruction error

Fig. 1. Daily summary of the network status, with an overview of the main network statistics and a list of the problematic devices (top) and a plot with the hourly amount of lost packets (bottom).

to maximize the correct separation of inliers and outliers. We set the threshold experimentally on the training and validation data. This approach obtains results comparable to statistical and rule-based methods.

From the dashboard of the application the user can monitor the performance of the autoencoder, and choose to modify the error threshold, or to retrain the neural network with more recent data or different time horizons (last week, last month, etc.) should the performance decreases significantly. Philéas allows the users to filter training/validation/test data, include the features they need and set the threshold. In Figure 4.1 we show two examples of information accessible form the Philéas dashboard; the first one is a daily summary of the overall

network status, with the main statistics and a list of the devices that lost too many packets, while the second one is a plot that reports, for a given day, the hourly amount of packets expected but not received.

4.2 Distributed Denial of Service

The second case we present comes from a French company, a major actor in the local IoT market with thousand of clients throughout the entire country. They deploy LoRa sensors for a variety of tasks; with no regular frequency, these devices transmit the information they collect to the central server via LoRaWAN packets. The company requested an analysis of their logs, following an occasional service failure experienced by several of their clients on a certain day d_a, whose devices were unable to connect to the network. The company reported that, overall, nearly 45% of the connections failed, contrarily to a circa 1% of probability of connection failure under normal circumstances. The hypothesis to verify is that this was a case of DDoS [38]. Here we report the analysis on the beginning of the attack.

Due to energy considerations, devices in LoRaWAN networks are not continuously connected to the network, but rather they send a join request to the network when they have to transmit data. If the request is accepted by the server, the device will be assigned a private token to be used during the communication, that will be checked by the gateway. The connection is closed when they stop transmitting, or if they get disconnected from the network. Join requests, both accepted and rejected, are normally received and stored by the central server. Gateways are configured to cap the number of join requests they can handle in a given amount of time; when the limit is exceeded, the gateway will reject all the new incoming join requests, to preserve the central server from the additional load. The recommended practice is therefore to minimize the number of connections, and to avoid repeated retries when a join request fails or in case of a network failure, in order to minimize the load on a network. Unfortunately, IoT protocols only enforce limited secure practices by design, so it is relatively easy for a malicious actor to disrupt a service by making some devices perform an excessive amount of join requests. When this exceeds the network capacity, also non-infected devices are impacted, experiencing more join failures than usual.

We were provided six months of anonymized LoRaWAN logs, for a total of approximately two terabytes of data. The LoRaWAN packets are composed of 12 downlink fields and 60 uplink ones, only one of which is the actual payload; the other ones include many accessory information that is not necessarily useful in many contexts. Moreover, several fields have been anonymized before giving us access to the data, so only partial information was available to us. The relevant fields for this task are the device ID, the client to which the device is associated, the timestamp, and the success status. The first step is to count the packages received by each device, and by the devices of each client. We use Spark to aggregate records in the dataset by device ID, day, and client, to count daily connections. Additionally, the aggregated data is now manageable without big data algorithms or technologies.

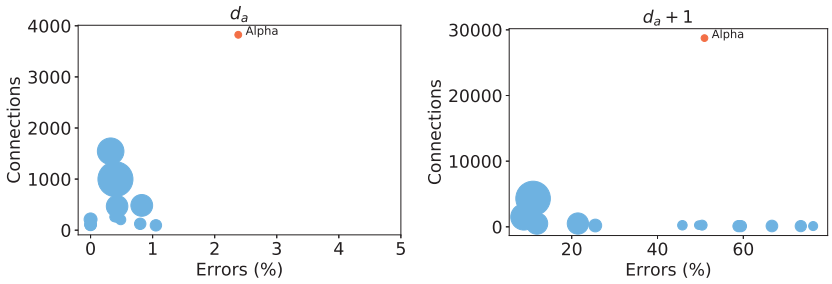


Fig. 2. Number of connections and connection error percentage for the main fifteen clients, on days d_a (the day the DDoS attack started, left plot) and $d_a + 1$ (when the attack continued, right plot). Each circle represent a client, the size of the circle is proportional to the number of devices controlled by that client. The colors indicate the outcome of the Local Outlier Factor analysis: in blue the clients that are considered inliers, in orange the client with anomalous behaviour (conventionally called Alpha).

The analysis by client shows clearly how the attack on one client impacted the other clients of the network. In Figure 4.2 we report the number of connections (on the y axis) and the error percentage (on the x axis) experienced by each client. The client that we call Alpha is the one hit by the attack and on day d_a it starts requesting an unusually high number of connections; on this day, the network load is still under control, and the other clients do not experience any particular issue. However, as the attack continues on day $d_a + 1$ and the network load increases to an excessive level, not only Alpha experiences a higher ratio of rejected joins, but also other clients become affected. In fact, all the clients experience, to various extents, an increase in the number of rejected connections. A consequence is the need for the devices to issue more join request than usual, in order to be able to transmit the information, even if, respecting the protocol recommendations, they do not issue nearly as many new join requests as the compromised client. The same outcome is observed with a Local Outlier Factor on the number of connections, normalized in the $[0, 1]$ interval, which confirms the abnormal behaviour of the devices of client Alpha.

This ex-post analysis serves also as a blueprint for the periodic monitoring of the network status. We can in fact periodically aggregate the data and spot anomalous behaviour by one or more clients; thanks to the reduced size of the aggregated dataset this can be done almost in real time.

5 Conclusions

With the growing interest in IoT technologies and applications, there is also a growing request in the market for data-based solutions to monitor IoT services. We introduced Philéas, a framework to analyze IoT logs to find anomalies in the

metadata, as an indication of potential problems in an IoT network. In Philéas we implemented several machine learning and anomaly detection techniques, and we have applied them to real-world cases of Degetel clients.

The framework can be used to implement custom solutions for clients with particular requirements; to this purpose, and depending on the requests, we are also going to include additional anomaly detection techniques, and network protocols.

Acknowledgements

The work has been made possible by the Innoviris project 2018-SHAPE-25a "PHILEAS: smart monitoring par détection de comportements anormaux appliquée aux objets connectés". M. Wattez contributed to the graphic interface and part of the implementation of the Philéas framework. We thank Shayp for the concession of using their data and use case in this paper. We thank Prof. G. Bontempi, J. De Stefani and G. Buroni for precious discussions and suggestions.

References

1. Ahmad, A., Khan, S.S.: Survey of state-of-the-art mixed data clustering algorithms. IEEE Access **7**, 31883–31902 (2019)
2. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Computer networks **54**(15), 2787–2805 (2010)
3. Baker, S.B., Xiang, W., Atkinson, I.: Internet of things for smart healthcare: Technologies, challenges, and opportunities. IEEE Access **5**, 26521–26544 (2017)
4. Banafa, A.: Three major challenges facing IoT. IEEE Internet of things (2017)
5. Ben-Gal, I.: Outlier detection. In: Data mining and knowledge discovery handbook, pp. 131–146. Springer (2005)
6. Blázquez-García, A., Conde, A., Mori, U., Lozano, J.A.: A review on outlier/anomaly detection in time series data. arXiv preprint arXiv:2002.04236 (2020)
7. Borgia, E.: The internet of things vision: Key features, applications and open issues. Computer Communications **54**, 1–31 (2014)
8. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: Identifying density-based local outliers. SIGMOD Rec. **29**(2), 93–104 (2000)
9. Ceccarello, M., Pietracaprina, A., Pucci, G.: Solving k-center clustering (with outliers) in mapreduce and streaming, almost as accurately as sequentially. arXiv preprint arXiv:1802.09205 (2018)
10. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM computing surveys (CSUR) **41**(3), 1–58 (2009)
11. Chen, Y., Qian, J., Saligrama, V.: A new one-class SVM for anomaly detection. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 3567–3571 (2013)
12. Cisco, C.V.N.I.: The zettabyte era—trends and analysis, 2015–2020. (2016)
13. Committee, L.A.T., et al.: Lorawan 1.1 specification. LoRa Alliance, Standard (2017)
14. Conti, M., Dehghantanha, A., Franke, K., Watson, S.: Internet of things security and forensics: Challenges and opportunities (2018)

15. Covington, M.J., Carskadden, R.: Threat implications of the internet of things. In: 2013 5th International Conference on Cyber Conflict (CYCON 2013). pp. 1–12. IEEE (2013)
16. for Cybersecurity, E.U.A.: Good practices for security of IoT. Tech. rep., European Union (02 2019)
17. Din, S., Paul, A., Hong, W.H., Seo, H.: Constrained application for mobility management using embedded devices in the internet of things based urban planning in smart cities. Sustainable Cities and Society **44**, 144 – 151 (2019)
18. Elijah, O., Rahman, T.A., Orikumhi, I., Leow, C.Y., Hindia, M.N.: An overview of internet of things (IoT) and data analytics in agriculture: Benefits and challenges. IEEE Internet of Things Journal **5**(5), 3758–3773 (2018)
19. Fu, K., Kohno, T., Lopresti, D., Mynatt, E., Nahrstedt, K., Patel, S., Richardson, D., Zorn, B.: Safety, security, and privacy threats posed by accelerating trends in the internet of things. arXiv preprint arXiv:2008.00017 (2020)
20. Gerla, M., Lee, E.K., Pau, G., Lee, U.: Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In: 2014 IEEE world forum on internet of things (WF-IoT). pp. 241–246. IEEE (2014)
21. Ghosh, A., Chakraborty, D., Law, A.: Artificial intelligence in internet of things. CAAI Transactions on Intelligence Technology **3**(4), 208–218 (2018)
22. Gil-Garcia, J.R., Pardo, T.A., Gasco-Hernandez, M.: Internet of Things and the Public Sector, pp. 3–24. Springer International Publishing (2020)
23. Gilchrist, A.: Industry 4.0: the industrial internet of things. Springer (2016)
24. Goix, N., Drougard, N., Brault, R., Chiapino, M.: One class splitting criteria for random forests. In: Zhang, M.L., Noh, Y.K. (eds.) Proceedings of the Ninth Asian Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 77, pp. 343–358. PMLR (2017)
25. Goldstein, M., Uchida, S.: A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. PloS one **11**(4), e0152173 (2016)
26. González García, C., Núñez Valdéz, E.R., García Díaz, V., Pelayo García-Bustelo, B.C., Cueva Lovelle, J.M.: A review of artificial intelligence in the internet of things. International Journal of Interactive Multimedia and Artificial Intelligence (2019)
27. Hodge, V., Austin, J.: A survey of outlier detection methodologies. Artificial intelligence review **22**(2), 85–126 (2004)
28. Hussain, F., Hussain, R., Hassan, S.A., Hossain, E.: Machine learning in iot security: current solutions and future challenges. IEEE Communications Surveys & Tutorials (2020)
29. Islam, S.R., Kwak, D., Kabir, M.H., Hossain, M., Kwak, K.S.: The internet of things for health care: a comprehensive survey. IEEE access **3**, 678–708 (2015)
30. Kaska, K., Beckvard, H., Minárik, T.: Huawei, 5G and China as a security threat. NATO Cooperative Cyber Defence Center for Excellence (CCDCOE) **28** (2019)
31. Kotenko, I.V., Saenko, I., Branitskiy, A.: Applying big data processing and machine learning methods for mobile internet of things security monitoring. J. Internet Serv. Inf. Secur. **8**(3), 54–63 (2018)
32. Lee, J., Stanley, M., Spanias, A., Tepedelenlioglu, C.: Integrating machine learning in embedded sensor systems for internet-of-things applications. In: 2016 IEEE international symposium on signal processing and information technology (ISSPIT). pp. 290–294. IEEE (2016)
33. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation-based anomaly detection. ACM Trans. Knowl. Discov. Data **6**(1) (2012)

34. Manavalan, E., Jayakrishna, K.: A review of internet of things (IoT) embedded sustainable supply chain for industry 4.0 requirements. Computers & Industrial Engineering **127**, 925–953 (2019)
35. Mehmood, Y., Ahmad, F., Yaqoob, I., Adnane, A., Imran, M., Guizani, S.: Internet-of-things-based smart cities: Recent advances and challenges. IEEE Communications Magazine **55**(9), 16–24 (2017)
36. Munirathinam, S.: Industry 4.0: Industrial internet of things (IIOT). In: Raj, P., Evangeline, P. (eds.) The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases, Advances in Computers, vol. 117, pp. 129 – 164. Elsevier (2020)
37. Ponti, M., Micheli, M., Scholten, H., Craglia, M.: Internet of things: Implications for governance (2019)
38. Salim, M.M., Rathore, S., Park, J.H.: Distributed denial of service attacks and its defenses in IoT: a survey. The Journal of Supercomputing pp. 1–44 (2019)
39. Sengupta, J., Ruj, S., Bit, S.D.: A comprehensive survey on attacks, security issues and blockchain solutions for IoT and IIoT. Journal of Network and Computer Applications **149** (2020)
40. Sezer, O.B., Dogdu, E., Ozbayoglu, A.M.: Context-aware computing, learning, and big data in internet of things: a survey. IEEE Internet of Things Journal **5**(1), 1–27 (2017)
41. Shipmon, D.T., Gurevitch, J.M., Piselli, P.M., Edwards, S.T.: Time series anomaly detection; detection of anomalous drops with limited features and sparse examples in noisy highly periodic data. arXiv preprint arXiv:1708.03665 (2017)
42. Soliman, M., Abiodun, T., Hamouda, T., Zhou, J., Lung, C.H.: Smart home: Integrating internet of things with web services and cloud computing. In: 2013 IEEE 5th international conference on cloud computing technology and science. vol. 2, pp. 317–320. IEEE (2013)
43. Solmaz, G., Wu, F., Cirillo, F., Kovacs, E., Santana, J.R., Sanchez, L., Sotres, P., Munoz, L.: Toward understanding crowd mobility in smart cities through the internet of things. IEEE Communications Magazine **57**(4), 40–46 (2019)
44. Sornin, N., Luis, M., Eirich, T., Kramp, T., Hersent, O.: Lorawan specification. LoRa alliance (2015)
45. Stojkoska, B.L.R., Trivodaliev, K.V.: A review of internet of things for smart home: Challenges and solutions. Journal of Cleaner Production **140**, 1454–1464 (2017)
46. Sun, Y., Song, H., Jara, A.J., Bie, R.: Internet of things and big data analytics for smart and connected communities. IEEE access **4**, 766–773 (2016)
47. Tzounis, A., Katsoulas, N., Bartzanas, T., Kittas, C.: Internet of things in agriculture, recent advances and future challenges. Bios. Eng. **164**, 31–48 (2017)
48. Xu, R., Wunsch, D.: Clustering, vol. 10. John Wiley & Sons (2008)
49. Yuehong, Y., Zeng, Y., Chen, X., Fan, Y.: The internet of things in healthcare: An overview. Journal of Industrial Information Integration **1**, 3–13 (2016)
50. Zanella, A., Bui, N., Castellani, A., Vangelista, L., Zorzi, M.: Internet of things for smart cities. IEEE Internet of Things journal **1**(1), 22–32 (2014)
51. Zhou, C., Paffenroth, R.C.: Anomaly detection with robust deep autoencoders. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 665–674. ACM, New York, NY, USA (2017)
52. Zolanvari, M., Teixeira, M.A., Gupta, L., Khan, K.M., Jain, R.: Machine learning-based network vulnerability analysis of industrial internet of things. IEEE Internet of Things Journal **6**(4), 6822–6834 (2019)
53. Zuniga, J.C., Ponsard, B.: Sigfox system description. LPWAN@ IETF97, Nov. 14th **25** (2016)

Evolving Virtual Embodied Agents using External Artifact Evaluations

Lesley van Hoek, Rob Saunders, and Roy de Kleijn

Leiden Institute of Advanced Computer Science, Leiden University, Leiden, the Netherlands
{l.s.van.hoek@umail, r.saunders@liacs, kleijnrde@fsw}.leidenuniv.nl
http://www.cs.leiden.edu

Abstract. We present *neatures*, a computational art system exploring the potential of digitally evolving artificial organisms for generating aesthetically pleasing artifacts. Hexapedal agents act in a virtual environment, which they can sense and manipulate through painting. Their cognitive models are designed in accordance with theory of situated cognition. Two experimental setups are investigated: painting with a narrow- and wide perspective vision sensor. Populations of agents are optimized for the aesthetic quality of their work using a complexity-based fitness function that solely evaluates the artifact. We show that external evaluation of artifacts can evolve behaviors that produce fit artworks. Our results suggest that wide-perspective vision may be more suited for maximizing aesthetic fitness while narrow-perspective vision induces more behavioral complexity and artifact diversity. We recognize that both setups evolve distinct strategies with their own merits. We further discuss our findings and propose future directions for the current approach.

Keywords: aesthetic evaluation · artificial intelligence · artificial life · autonomous behavior · computational creativity · embodied agents · evolutionary art · neural networks · situated action · situated cognition

1 Introduction

Computational systems that produce artworks with high levels of autonomy have always provoked discussion about the definition of art and creativity. Researchers and artists working in the field of evolutionary and generative art cede control to autonomous systems that produce artworks, often intending to eliminate human intervention where possible [17]. Digital evolution is an established algorithmic process that has proven very capable of innovation [18]. In art and design, appropriate implementation of this technique can aid the generation of novel, valuable and surprising artifacts [4][2] that may be deemed creative by unbiased observers [8]. It has also been essential in the field of artificial life (a-life) [26] where researchers have been consistently surprised by creative solutions invented by artificial organisms evolving in computational environments [28]. Naturally, the process of digital evolution merely imitates life itself. The biological mechanism of natural selection is known to find and cause inventive adaptations that

enhance the survival and reproduction of organisms [15]. Consequently, these may lead to the appearance of design without a designer [10]. Adaptations may include changes in behavior. We aim our attention at a particular behavior in some non-human organisms, namely the creation of artifacts.

Several species in the natural world are known to decorate and produce structures that resemble visual art in the sense that they are intended to be attractive to potential mates. This structure creation is an important behavioral characteristic of male bower birds [12] and white-spotted pufferfishes [31]. In this paper, we explore whether artificial organisms could adapt to similar, but digitally induced pressures as a consequence of constructing artifacts. The following sections briefly discuss some challenges related to building such a computational system.

1.1 Computational aesthetic evaluation

Early examples of evolutionary art include the highly influential work of Sims [41] and Latham [47], who used genetic algorithms to mutate symbolic expressions for the composition of unpredictable yet interesting visual shapes and patterns. Both adopted a top-down approach that relies on human aesthetic judgment for the evaluation of artifacts using an interactive genetic algorithm (IGA). This technique facilitates easy exploration of large parameter spaces [44] but suffers from significant limitations: (1) IGAs rely on human evaluation at every iteration and so suffer from the *fitness bottleneck* [46], and (2) human fatigue and inconsistency make it difficult to capture universal measures [44]. Attempts to overcome these limitations have included massively multi-user systems [39] and the application of machine learning to capture user preferences [33].

Challenges in IGA helped inspire the research field of computational aesthetic evaluation (CAE), where people seek computational solutions for the assessment of human aesthetics [23]. Machado and Cardoso [29] created *NEvAr*, an autonomous system that evolves Sims' symbolic expressions with an automated evaluation procedure for images that focuses exclusively on form. Here, a speculative fitness function inspired by the study of *information aesthetics* [35] was designed which favors images that are "simultaneously, visually complex and that can be processed (by our brains) easily". In the science of aesthetics, *NEvAr*'s fitness function indicates a *formalist theory* as it proposes aesthetic experience relies on the intrinsic beauty of the artifact. In contrast, a *conceptual theory* relies on other factors that may be more important for aesthetic preference like socio-cultural contexts of the work and the previous experience of the artists and observers [40]. In a more recent publication, Redies [36] proposes a model of visual aesthetic experience that unifies these two theories. Ultimately, there is currently no agreement on which paradigm offers the most effective computational framework of human aesthetics.

1.2 Embodiment

Theorists in situated cognition view the environment as highly significant to driving human cognitive processes. Clark and Chalmers [7] suggest that the

environment directly influences an agent's behaviors as part of a two-way interaction between action and perception. Here, embodiment is key because it allows us to manipulate it to our needs. Biological brains have evolved to take advantage of the environment by offloading cognition to it through the body. Simultaneously, our visual systems evolved to rely on it more. This perspective supports the view of *externalism*, in which the cognitive process is considered something that occurs in- and outside of the mind [7]. In this context, embodiment is key to the creation of art and can be imagined as a feedback loop of action and perception occurring through a body. Brinck [5] states that the production (and consumption) of visual art can be accounted for by the theory of situated cognition [6]: "Artist and canvas form a coupled system. Artistic practice starts with gaze, and then comes the gesture that accomplishes itself when the artist is in touch with the piece [they are] working on."[5]

Experiments in the use of embodied artificial organisms and situated cognition for computational art and creativity have largely been unexplored. Thus, we present *neatures*: a prototype for an autonomous art system that simulates artificial organisms capable of producing visual art in their environment.

2 Related work

There have been several interesting art and research projects involving the use of embodied agents to create visual art. Jean Tinguely experimented with mechanical drawing machines in the 50s, exploring notions of automated artists and artificial creative processes [13]. Influences to his work can be seen in the field of swarm painting, which involves the simulation of agents supplied with some form of cognition producing emergent artworks. *Robotic Action Painter* [34] is an autonomous abstract art system based on behavioral studies of ants and other social insects. An artwork is created by employing several small wheeled robots that leave colored lines (pheromone) as they travel. A color detection sensor on each robot recognizes these lines in the environment and triggers specified behaviors for particular colors—a process analogous to *stigmergy*; a form of self-organization [14]. The result is a painting with chaotic structures that are free from preconceptions and merely represent the actions themselves. McCormack developed similar experiments using biological processes of niche construction to enhance the diversity and variation of agents' behaviors in his art system [32].

Drawing machines that take a more anthropomorphic approach can be classified as robot painters. *eDavid* [11] is an industrial robot that simulates the human painting process using a visual feedback loop to explore painterly rendering on a real canvas. Explorations in expanding its artistic skill demonstrated the possibility of expressing a given collection of images in a different style [48]. With *neatures*, we take inspiration from the flexibility of robot painters and the emerging complexity of swarms to explore the effects of aesthetic selection pressures in an evolutionary art system.

3 Implementation

Neatures[1] is a prototype computational visual art system that was developed in an attempt to employ artificial organisms for the production of visual artifacts. The current implementation is heavily inspired by the seminal work of Sims (1994), Evolving Virtual Creatures, in which a genetic algorithm was used to guide the evolution of specific abilities such as locomotion and jumping. *Neatures*' artificial organisms 'live' in three-dimensional space and are subject to physically plausible simulation. This is achieved using the Bullet physics engine [9]. The software comprises of a controller server which stores the population and commands the complete evolutionary process. A simulator client can connect to a controller and receive queries for queued rollouts. This component features a graphical user interface, allowing the user to observe the virtual organisms in real-time. The following sections briefly cover the system implementation.

3.1 Agent morphology

Virtual organisms situated in physically plausible environments are subject to strict laws of physics and, like real organisms, require an appropriate body to fulfill their purpose. Designing such a body is a difficult task, and perhaps best suited for an evolutionary process to solve. Sims [41] used a genotypic encoding of nodes and connections for the morphology of his creatures, and genetic operators, allowing for the evolution of morphology alongside control policy. In this system, a genotypic encoding scheme is used to generate a hexapod at the start of a simulation and remains fixed. The reason for this is that evolutionary optimization of morphology dramatically increases the complexity of the search landscape and is incompatible with fixed-topology neural network architectures.

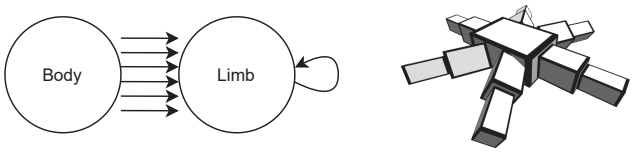


Fig. 1: Agent morphology genotype (left) and phenotype (right).

Each element stores some information about their phenotypic transformations such as size, attachment points, and node or joint type. A phenotype generation algorithm recursively traverses the graph and builds a hierarchical

[1] *Neatures* is open-source and available at https://github.com/lshoek/creative-evo-simulator

structure of boxes connected to each other by joints. Fig. 1 depicts the morphology encoding and phenotype of a hexapod. The algorithm in the current work was implemented after Krčah's example [25] with some alterations tailored to suit this work's purpose. One notable difference, for instance, is that we use a single degree of freedom per joint for simplicity.

3.2 Agent control policy

In every simulation rollout, agents are tasked to produce an artifact in their environment. In order to achieve this in *neatures*, we chose to implement a painting system. Each agent is equipped with a single brush-type node capable of applying virtual ink drops to the canvas; a specified surface area in the environment that the agent can sense and manipulate. Four invisible walls are located at a specified distance from the canvas edges to prevent agents from moving too far away from the center. Ink is only released under the conditions that the brush node is in contact with the canvas, and the agent has decided to activate it.

An agent's decision-making process and behavior are determined by its control policy. This is defined by a neural controller that continuously accepts sensory data as input, and based on this data, outputs a set of activation values. Agents sense their environment through two types of sensors: (1) a proprioceptive sensor, implemented by tracking the current joint angles and storing these in $a \in \mathbb{R}^j$, where j is equal to the number of joints in the agent's morphology and (2) a vision sensor capturing a 64x64px grayscale bitmap representation of the current canvas' content. The data of both sensors is appended to form an observation to be fed to the neural controller at regular time intervals. The physics engine and control policy are updated 60 and 20 times per second of simulated time, respectively. Fig. 2 presents the complete cognitive model of an agent.

The neural controller involves two cognitive modules; a vision model V for processing visual data inside the incoming observation, and an action model C to generate the agent's next action. V is a *convolutional variational autoencoder* (CVAE), pre-trained to compress the canvas data to a latent vector $z \in \mathbb{R}^{32}$. C

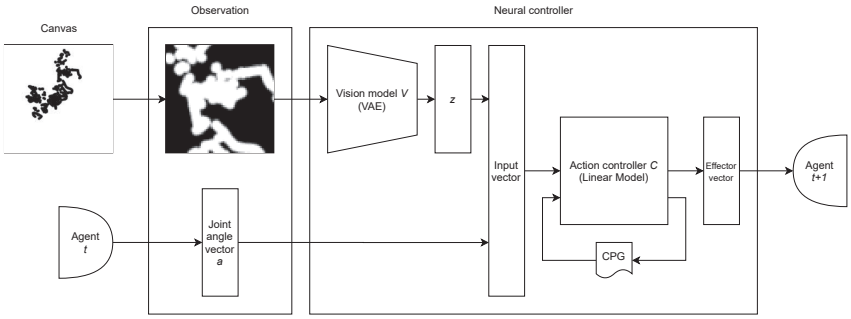


Fig. 2: Cognitive model of an agent.

is a simple linear model that takes as input a combination of latent vector z, a joint angle vector, and an additional value to stimulate continuous movement. Compression of the visual data allows the action controller to be kept small, which alleviates the credit assignment problem in difficult reinforcement learning (RL) tasks and tends to iterate faster [19]. The output layer of C uses a *tanh* activation function to output to produce a vector of effector values, including target joint angles used to update the motor parameters of the agent's joints and a value indicating the stroke width of the brush. Finally, a stimulation output value connects to a central pattern generator (CPG) after which a feedback connection to the corresponding action model input is made for the next time the neural controller is queried [24]. This minimal recurrent network structure is set up this way to evoke changing joint angle outputs. Without it, the agent would cease to move in cases where its observations remain unchanged over multiple frames and its body incidentally has zero momentum. Additionally, as sensory input drives neural excitation, it grants C control over the agent's movement speed, which could bring about more interesting behaviors. Section 4.3 describes the training procedure for V and C.

4 Experiment

We carry out two experiments where an artificial organism is evolved by optimizing for the aesthetic quality of its artifacts. The artistic medium of expression chosen for this task is painting. The main reason for this is that there exists a multitude of interesting theories and evaluation techniques of visual human aesthetics—suitable for two-dimensional content—that could be pursued to design an acceptable fitness function [16].

Fig. 3: The *neatures* simulator showing an agent painting.

As stated in Section 3.1, we decided to exclude morphology from evolutionary optimization, meaning we must formulate an appropriate body design for

the current experiment ourselves. We take inspiration from behavioral robotics research, where it has long been common practice to use biologically based robot designs to study artificial organisms [1]. As a matter of course, the insect-like hexapod was chosen for the current task. This design is a popular benchmark that we suppose will allow for an adequate degree of flexibility required to explore the possibilities of the virtual environment. Fig. 3 shows a screenshot of the agent as it appears in the simulator client of the system.

4.1 Setup

The following is a brief description of the realized experiments. In the first setup, the agent is supplied with a wide-perspective vision sensor. This is defined as a 64x64px grayscale bitmap representation of the environment that is equal to the size of the canvas. The orientation of this representation is at all times aligned with the facing direction of the agent and centered around the point where it last touched the canvas with its brush node. Fig. 4a shows an example of how the canvas is sensed with this perspective. The second setup supplies the agent with a narrow-perspective vision sensor, encompassing 6,5% of the canvas area as shown in Fig. 4b.

(a) **(b)**

Fig. 4: The mapping from canvas (left) to visual field (right), marked in red, for wide-perspective (4a) and narrow-perspective (4b).

The vision capabilities of the agent exist in a separate conceptual space from the one it is situated in. Agents' visual capabilities exist in *artifact space*, whereas their neural controllers output actions in *effector space*. The former is a two-dimensional representation of the environment, cultivated by the agent itself. The latter relates to objects in the three-dimensional virtual environment. Other than muscle memory (the action controller parameters), an agent has no other capabilities of memorization. As a result, the environment is the only cognitive resource to the agent by which an approximate model of situated cognition is realized. The key idea to this experiment is that, under the given conditions, a mapping between these two may be learned. If successful, the creature would be able to produce an aesthetically pleasing artifact in *artifact space* by means of its motor function in *effector space*.

4.2 Measuring aesthetic quality

After a rollout has ended, the resulting artifact is queued for fitness evaluation. In our computational environment, the fitness function is a proxy for natural selection pressures that cause the evolution of adaptations [15]. As outsiders to this virtual world, we can design this function externally, and observe what behaviors emerge from evolutionary optimization. Taking inspiration from some animal species' mate selection indicators that are attributed to external artifacts, we intentionally ignore any behavioral aspects of an agent's existence. Our fitness function is designed to evaluate images in accordance with speculative visual aesthetic theory, essentially assuming the role of an art critic.

To measure the aesthetic quality of an artifact, we use a metric closely related to Birkhoff's [3] formalist aesthetic measure, defining the formula $M = O/C$, where M is the aesthetic effectiveness, O is the degree of order and C is the degree of complexity. Birkhoff theorizes that aesthetic response to an object is stronger when the degree to which psychological effort is required to perceive it—induced by its complex features—is met with a higher degree of tension being released as the perception is realized—originating from orderly features such as symmetry and self-similarity. This formula has been disputed early and is generally regarded as inaccurate [49]. Scha & Bod [37], for instance, note that it penalizes complexity too considerably and is better suited as a measure of the degree of self-similarity. Galanter [16], however, notes that at least two aspects of Birkhoff's work remain legitimate today; the intuitive connection between aesthetic value and order/complexity relationships, and the search for a neurological base of aesthetic behavior. These aspects are reflected in the fitness function of Machado & Cardoso [29], defined in Eq. 1. Inspired by information aesthetics [35], Machado & Cardoso speculate an image's intrinsic aesthetic value to be equal to the ratio of image complexity IC to processing complexity PC.

$$reward_{aesthetic} = \frac{IC}{PC} \tag{1}$$

PC is measured at two temporal instances ($t0$ and $t1$) in the time it takes to perceive an image and provide Eq. 2. The processing complexity is maximized as PC_{t1} and PC_{t0} approach each other.

$$PC = (PC_{t0}PC_{t1})^a \left(\frac{PC_{t1} - PC_{t0}}{PC_{t1}} \right)^b \tag{2}$$

In order to find PC_{t0} and PC_{t1}, we calculate the inverse of the root mean square error (RMSE) between the original image i, and the same image after fractal compression $Fractal(i)$, as shown in Eq. 3.

$$PC_{tn} = \frac{1}{RMSE(Fractal(i),\ i)} \tag{3}$$

Machado et al. [30] compared several complexity measures with human ratings across a selected set of images in five distinct stylistic categories. Among

the results of their feature extraction experiments, their JPEG-Sobel method was found to correlate the most with human ratings, especially those related to the abstract artistic category. We calculate IC following this method as shown in Eq. 4. First, the Sobel [42] edge detection operator is applied to i horizontal and vertical directions, after which the resulting gradients are averaged. Then, JPEG compression is performed on the edges. In the dividend, size defines the total number of bytes required to store the image data.

$$IC = \frac{RMSE(Sobel(i),\ JPEG(Sobel(i)))}{size(Sobel(i))size(JPEG(Sobel(i)))} \tag{4}$$

Taylor et al. [45] note the fractal qualities of late-period action paintings by Jackson Pollock and suggests their fractal dimensions are correlated with their aesthetic qualities. Therefore, we decided to parameterize Eq. 2 using $a = 0.6$ and $b = 0.3$, increasing bias towards artifacts with more orderly features with respect to the reference implementation [29]. We argue that this suits the current experimental setup by countering excessive levels of image complexity in the artifacts due to the generally chaotic nature of agents' behaviors that generate complex and incidental painting patterns by default.

In early experiments, we found that additional encouragement to act through an easily attainable coverage reward could help agents to advance faster in early generations. This has the added benefit that a minimum specified amount of content is imposed on the artifacts. Eq. 5 defines $reward_{coverage}(x)$, where x is the mean of all normalized pixel intensities of the artifact and p is the peak coverage rate. It is essentially a smooth interpolation between x and p, ensuring a result of 1 when $x \geq p$.

$$reward_{coverage}\,(x) = 1 - sin\left(\pi \frac{\frac{1}{p}x + 1}{2}\right)^4 \tag{5}$$

with initial condition

$$x = min(x,\ p) \tag{6}$$

In our experiments, we use $p = 0.0625$, meaning that the maximum coverage reward is already reached when 6,25% of the canvas area is painted. Finally, the total artifact fitness is calculated as defined in Eq. 7. This shows the aesthetic reward is proportional to the coverage reward until peak reward p is reached, thus penalizing paintings that have little content. Table 1 presents a set of images and their fitness values.

$$fitness = 100\ reward_{coverage} + reward_{aesthetic} \times reward_{coverage} \tag{7}$$

We find these results to be satisfactory for our purposes. Although the fitness function is arguably too generous on Gaussian noise (Table 1d), such an artifact is practically impossible for an agent to produce. The Pollock-snippet (Table 1e) is evaluated far more positively and represents a more plausible result.

Table 1: A set of images and their fitness: (a) perfect symmetry, (b) an early-generation artifact with little variability in stroke width, (c) an early-generation artifact with high variability in stroke width, (d) Gaussian noise, (e) a contrast-enhanced snippet of No. 26A: Black and White by Jackson Pollock (1948).

	(a)	(b)	(c)	(d)	(e)
Fitness	101.2	116.2	145.9	399.5	871.5
Coverage	27.7%	15.5%	12.4%	18.2%	46.0%
IC	0.0697	0.4042	0.8046	44.269	48.443
PC	0.0561	0.0250	0.0175	0.0148	0.0063

4.3 Training procedure

Before any control policies can be evolved, visual model V must be pre-trained to discern between visual observations. First, 20,000 artifact samples (256x256 grayscale bitmaps) were collected in a preliminary run using an untrained visual model V. Then, a new dataset was generated by applying random affine transformations to each collected sample. This new dataset is more representative of an agent's visual observations. Finally, using the updated dataset, V was trained to encode visual observations into latent vector $z \in \mathbb{R}^{32}$ for 200 epochs.

Agents' control policies are optimized through evaluation of the quality of their work, rather than the means by which it was achieved. This indirect correspondence between goal and action may reduce credit assignment accuracy of gradient-based numerical optimization algorithms as adaptations to action controller C could have unanticipated effects on an artifact's fitness. Therefore, gradient-free methods such as evolution strategies [38] might be best suited for solving this problem. *Neuroevolution* methods have a long history of success with evolutionary robotics and have recently increased in popularity as they have been found to perform considerably well on deep RL tasks [43]. With this in consideration, we chose *covariance matrix adaptation evolution strategy* (CMA-ES) [20] for the optimization of C's parameters. Evidence shows that the algorithm performs relatively well on deceptive landscapes or sparse-reward functions up to a couple of thousands of parameters [22]. We use an open-source Python implementation of the algorithm by Hansen [21].

At the start of every evolution process, the weights of every action controller C in the population are randomly initialized with $\mu = 0$ and $\sigma = 0.1$. A population size of 32 is used, where each candidate's behavior is determined by their corresponding C, comprising 658 trainable parameters each. Every generation, one rollout is performed per agent and results in 32 artifacts. A rollout is defined as 240 seconds of simulated time an agent spends in the environment. Evaluations occur immediately after each rollout in a separate process. After all rollouts and evaluations are finished, CMA-ES uses the collected fitness values to update

each candidate's action controller parameters for the next generation. Both experiments are performed using an evolutionary process of 350 generations.

Our training setup marks several notable limitations. Foremost, the experiments are carried out separately on two mid-range laptops (i7-7700HQ/GTX1050 and i7-8750H/GTX1070), each running a single simulator client and controller server at the same time. The most significant bottleneck comes from the fractal compression procedure required for each artifact evaluation. In the current setup, we simulate two populations of 32 candidates for 350 generations and takes about 40 hours to complete. More reliable results could be collected by increasing the population size and averaging fitness over multiple rollouts for a more representative metric of the agent's general painting strategy. This is however outside of the scope of this research.

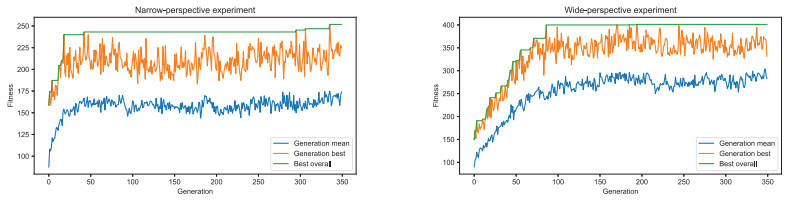


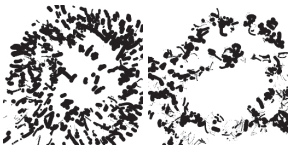
Fig. 5: Fitnesses of the narrow- (left) and wide-perspective populations (right).

5 Results

Fig. 5 presents the fitness results of the narrow- and wide-perspective vision experiments. Here, we see that the narrow-perspective population's mean fitness starts with a steep positive trend and converges towards a local optimum before the 50th generation. The wide-perspective population's mean fitness improves gradually up to around the 100th generation before a local optimum is reached. We also see that the wide-perspective population is generally about 150 points ahead of the narrow-perspective population. From these results, it is evident that the wide-perspective population performs better in terms of fitness. However, it barely shows any signs of improvement after a local optimum has been reached, until the final generation of the simulation. This is unlike the narrow-perspective population, which shows a slight upward trend around the 300th generation, and some new best-ever artifacts of the population. Table 2 presents the highest-rated artifacts of both experiments along with some key statistics. Almost every artifact shows a clear trajectory on the canvas that is telling of the strategy that was used to produce it. Fig. 6 below shows the highest-rated artifacts of the first 64 generations of both populations. We see that the sort of artifacts produced by both populations can easily be distinguished from approximately the 40th generation. From there on, we see that nearly all artifacts of the wide-vision population

Table 2: The highest-rated artifacts of all generations of wide-perspective and narrow-perspective populations and their key statistics.

Pers.	*Fitness*	*Cov.*	*IC*	*PC*	*Gen.*
Wide	401.05	36.05%	2.8339	0.0094	194
Narrow	251.82	21.33%	1.7714	0.0116	335



Wide Narrow

indicate a circular movement strategy, with little diversity among paintings. The fitness results and artifacts of this population show that this strategy is further exploited in subsequent generations, likely because of its effective contribution to maximizing fitness. In contrast, the narrow-perspective population struggles to escape a local optimum early on but demonstrates far more diversity among its artifacts in all generations. This suggests that potentially fit strategies are being explored rather than being exploited.

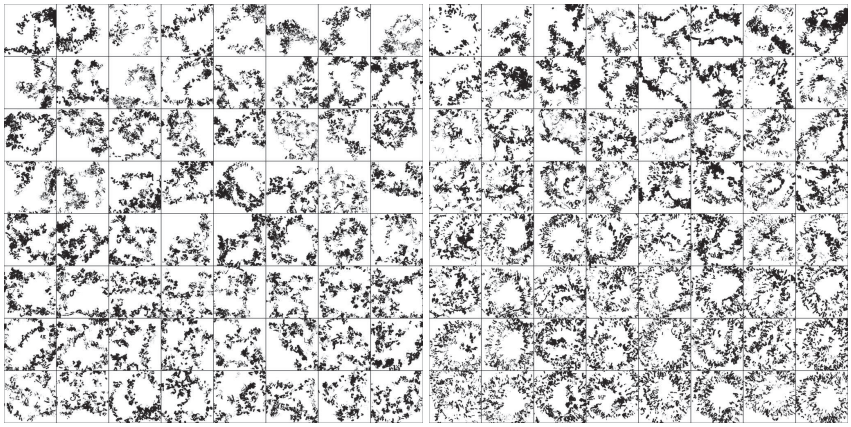


Fig. 6: The best artifacts of the first 64 generations (top-left to bottom-right) of the narrow- (left) and the wide-perspective population (right).

The discrepancy between the fitness results and the type of artifacts produced by both populations led us to believe that coverage and fitness may be strongly positively correlated. To investigate, we plotted coverage against fitness (Fig. 7) and observed that coverage is an accurate predictor of fitness in the wide-perspective population, but not necessarily for the narrow-perspective population.

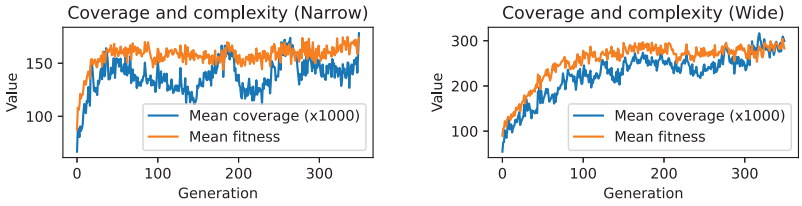


Fig. 7: Mean coverage and fitness in narrow- (left) and wide-perspective populations (right).

6 Discussion

Our results of the current experiment demonstrate a notable distinction between the narrow- and wide-perspective setup. In our experiment, we observe that virtual organisms with narrow-perspective vision trigger explorative search of the fitness landscape by the evolutionary algorithm and demonstrate more complex and distinct behavior. We also see that this is not necessarily in the interest of maximizing fitness. One explanation for this could be that relatively small adaptations to a narrow-perspective controller's weights lead to greater variations in the emerging painting strategy. In the agent's cognitive model, perception and action are closely coupled together. Therefore, distinct actions may be more likely to be triggered when visual observations are more volatile, as is the case with the narrow-perspective agents. This is in line with Brinck's [5] argument that art creation is a situated activity, noting that what the artist perceives is directly transformed into action. We further observe that narrow-perspective agents generally appear more sensitive to the environment in their painting strategies than wide-perspective agents. Narrow-perspective agents show more effective corrective behavior such as turning near the edge of the canvas. This is not as apparent in wide-perspective agents who barely appear to discernibly change their behavior near edges. Little response to edges is likely induced by the exploitation of circular movement patterns—evidently an effective strategy for painting highly fit artifacts. We further think that the widespread coverage of paint in the environment reinforces an agent's behavioral pattern. This may be due to the relatively poor compression quality of global features in visual observations of developed circular patterns, leading to similar encodings of z. Incidentally, this fact may have greatly contributed to finding the circular movement strategy.

From our observations, we theorize that volatile visual information, as demonstrated by the narrow-perspective experiment, considerably complicates the shape of the fitness landscape. For instance, a consistent circular movement strategy would be much more difficult to sustain over the length of a rollout, and over multiple generations, with narrow-perspective vision than with wide-perspective vision. Even more so, this automatically concerns any potential strategy. Although volatile visual information may impede the evolution of consistent action and perception, it does have creative merit in the sense that it elicits greater

behavioral complexity in agents. Hence, the narrow-perspective population has explored the greatest *artifact space*. This is demonstrated in Fig. 8 which presents two random selections of artifacts created in both populations.

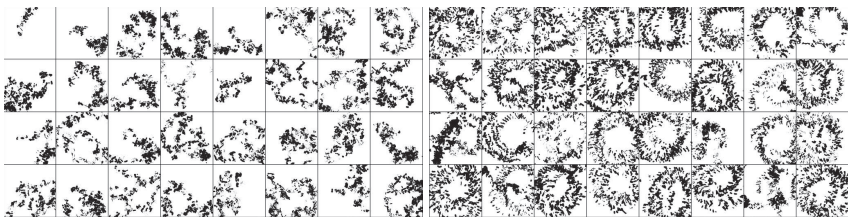


Fig. 8: Two random selections of artifacts drawn from all of the narrow-perspective population (left) and the wide-perspective population (right).

Considering our evaluation procedure; if we, hypothetically, consider Pollock's work as an aesthetic benchmark for this system (Table 1e), we consider the current fitness function helpful at guiding agents' technique towards this aesthetic up to a certain point. Fig. 7 however suggests a possible perverse instantiation problem; at least one strategy exists in which coverage can be exploited to maximize fitness. However, we believe an adjustment to the fitness function would be premature. This is because, as the fitness function is based on complexity, coverage cannot be positively correlated with fitness as it approaches 50%. The highest recorded coverage of all artifacts in both populations is 36%, whereas the coverage of our Pollock example (Table 1e) is measured at 46%. We are confident that under the current time pressure of 240 seconds, it is physically not possible for agents to cover a significantly greater part of the canvas. Therefore, we believe that agents should be assigned sufficient time so that 50% coverage could be achieved. After this is explored, we believe that a worthwhile addition to the fitness function would be a novelty reward term to overcome local optima by encouraging exploration [27].

In our experiment, we see that a proxy for selection pressures based in aesthetic properties of an external artifact can evolve a virtual organism with some success. Our agents' artificially emergent and autonomous behaviors resemble those of simple biological organisms in some ways on a superficial level and are rather interesting to observe. Whether some of the resulting artifacts are aesthetically pleasing is up to the beholder. Their chaotic patterns and compositions certainly parallel abstract expressionist action paintings to some degree. The agents' paintings share an interesting connection to this art movement as all brushstrokes represent nothing but the actions themselves. With that, one could argue for their artistic value.

6.1 Future work

We briefly propose future directions for the current research. Foremost, the system would highly benefit from a more robust visual model, as emphasized by the poor reconstruction quality of wide-perspective visual observations. This can be achieved by using a larger dataset of intermediate visual observations. Future work could assess whether granting a virtual organism continuous agency over its visual perspective, approximating the cognitive process of *attention*, is a worthwhile approach. This feature is trainable and could explore the nuance between the benefits of the demonstrated visual perspectives.

The morphology and environmental setup we chose for the task of painting is by no means the most suitable. We recommend that future work in embodied agent art should keep exploring the evolution of morphologies. This prevents authors from making predisposed choices about the most suitable body for a given task. A significant downside to this is that it requires a flexible network structure for the action controller model that is significantly more difficult to train. A search algorithm for appropriate morphology choice is another separate topic that could be further explored in the context of art-producing artificial organisms [27]. Furthermore, agents in the current work are limited to a single type of brush, paint color, and environment to explore. Therefore, future extensions could try implementing physically based painting systems, color palettes, and varying environments, each of which could bring about interesting new artifacts and behaviors. Ultimately, painting is only one method of artistic practice, and by no means the most suitable for embodied agents to practice. Computational organisms and environments allow for other artistic modes of expression to be explored such as sculpture, dance, music, poetry, etc. The possibilities are far-reaching and may one day perhaps exceed our imagination.

7 Conclusion

We have demonstrated that virtual organisms can be evolved to make aesthetically pleasing paintings using selection pressures based on aesthetic properties of the painting. The results from our experiments show notable behavioral differences between agents employed with wide-perspective and narrow-perspective vision. The wide-perspective population achieved the best results in terms of fitness by evolving a circular movement strategy effective at maximizing fitness early on, but later showing barely any signs of improvement. The narrow-perspective population performed worse and did not evolve an exploitable strategy. Instead, it brought about a diverse set of artifacts across all generations. From this we conclude that the wide-perspective setup may be more suited for maximizing aesthetic fitness while the narrow-vision setup induces more behavioral complexity and artifact diversity. Although, the scope of this research is limited, our results provided some interesting insights and discussions which provide directions for future applications of computational art systems employing virtual organisms.

References

1. Beer, R.D., Chiel, H.J., Sterling, L.S.: A biological perspective on autonomous agent design. Robotics and Autonomous Systems **6**(1-2), 169–186 (1990)
2. Bentley, P.J.: Is evolution creative. In Proc. of the AISB **99**, 28–34 (1999)
3. Birkhoff, G.D.: Aesthetic measure (p. 226). Cambridge, Mass (1933)
4. Boden, M.A.: The Creative Mind: Myths and Mechanisms. Psychology Press, Hove, UK (1990)
5. Brinck, I.: Situated cognition. On artistic creativity and aesthetic experience, dynamic systems, and art (2007)
6. Clancey, W.J.: Situated cognition: On human knowledge and computer representations. Cambridge Univ. Press (1997)
7. Clark, A., Chalmers, D.: The extended mind. Analysis **58**(1), 7–19 (1998)
8. Colton, S., Wiggins, G.A., et al.: Computational creativity: The final frontier? In: Ecai. vol. 12, pp. 21–26. Montpelier (2012)
9. Coumans, E., et al.: Bullet physics library. Open source: bulletphysics. org **15**(49), 5 (2013)
10. Dennett, D.C., Dennett, D.C.: Darwin's Dangerous Idea: Evolution and the Meanins of Life. Simon and Schuster (1996)
11. Deussen, O., Lindemeier, T., Pirk, S., Tautzenberger, M.: Feedback-guided stroke placement for a painting machine. CAe **8** (2012)
12. Diamond, J.: Animal art: variation in bower decorating style among male bowerbirds amblyornis inornatus. Proc. of the National Academy of Sciences **83**(9), 3042–3046 (1986)
13. Dohm, K., Stahlhut, H., Hoffmann, J.: Kunstmaschinen Maschinenkunst. Kehrer Verlag (2007)
14. Dorigo, M., Bonabeau, E., Theraulaz, G.: Ant algorithms and stigmergy. Future Generation Computer Systems **16**(8), 851–871 (2000)
15. Futuyma, D.J.: Natural selection and adaptation. Evolution pp. 279–301 (2009)
16. Galanter, P.: Computational aesthetic evaluation: past and future. In: Computers and creativity, pp. 255–293. Springer (2012)
17. Galanter, P.: Generative art theory. A Companion to Digital Art pp. 146–180 (2016)
18. Goldberg, D.E.: The race, the hurdle, and the sweet spot. Evolutionary design by computers pp. 105–118 (1999)
19. Ha, D., Schmidhuber, J.: World models. arXiv preprint arXiv:1803.10122 (2018)
20. Hansen, N.: The cma evolution strategy: A tutorial. arxiv (2016), preprint
21. Hansen, N., Akimoto, Y., Baudis, P.: Cma-es/pycma on github. Zenodo, doi **10** (2019)
22. Hansen, N., Auger, A., Ros, R., Finck, S., Pošík, P.: Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In: GECCO. pp. 1689–1696 (2010)
23. Hoenig, F.: Defining computational aesthetics. The Eurographics Assoc. (2005)
24. Hülse, M., Wischmann, S., Manoonpong, P., Von Twickel, A., Pasemann, F.: Dynamical systems in the sensorimotor loop: On the interrelation between internal and external mechanisms of evolved robot behavior. In: 50 years of artificial intelligence, pp. 186–195. Springer (2007)
25. Krčah, P.: Evolution and Learning of Virtual Robots. Ph.D. thesis, Univerzita Karlova (2016)
26. Langton, C.G.: Artificial life: An overview. Mit Press (1997)

27. Lehman, J., Stanley, K.O.: Abandoning objectives: Evolution through the search for novelty alone. Evolutionary computation **19**(2), 189–223 (2011)
28. Lehman, J., Clune, J., Misevic, D., Adami, C., Altenberg, L., Beaulieu, J., Bentley, P.J., Bernard, S., Beslon, G., Bryson, D.M., et al.: The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. Artificial Life **26**(2), 274–306 (2020)
29. Machado, P., Cardoso, A.: All the truth about nevar. Applied Intelligence **16**(2), 101–118 (2002)
30. Machado, P., Romero, J., Nadal, M., Santos, A., Correia, J., Carballal, A.: Computerized measures of visual complexity. Acta psychologica **160**, 43–57 (2015)
31. Matsuura, K.: A new pufferfish of the genus torquigener that builds "mystery circles" on sandy bottoms in the ryukyu islands, japan (actinopterygii: Tetraodontiformes: Tetraodontidae). Ichthyological research **62**(2), 207–212 (2015)
32. McCormack, J.: Niche constructing drawing robots. In: EvoMUSART. pp. 201–216. Springer (2017)
33. McCormack, J., Lomas, A.: Understanding aesthetic evaluation using deep learning. In: EvoMUSART. pp. 118–133. Springer (2020)
34. Moura, L.: A new kind of art: The robotic action painter. X Generative Art Conf., Politecnico di Milano Univ. (2007)
35. Nake, F.: Information aesthetics: An heroic experiment. EvoMUSART **6**(2-3), 65–75 (2012)
36. Redies, C.: Combining universal beauty and cultural context in a unifying model of visual aesthetic experience. Frontiers in human neuroscience **9**, 218 (2015)
37. Scha, R., Bod, R.: Computationele esthetica. Informatie en Informatiebeleid **11**(1), 54–63 (1993)
38. Schwefel, H.P.: Numerical optimization of computer models. John Wiley & Sons, Inc (1981)
39. Secretan, J., Beato, N., D'Ambrosio, D.B., Rodriguez, A., Campbell, A., Folsom-Kovarik, J.T., Stanley, K.O.: Picbreeder: A case study in collaborative evolutionary exploration of design space. Evolutionary computation **19**(3), 373–403 (2011)
40. Shimamura, A.P., Palmer, S.E.E.: Aesthetic science: Connecting minds, brains, and experience. OUP USA (2012)
41. Sims, K.: Artificial evolution for computer graphics. In: PACMCGIT. vol. 18, pp. 319–328 (1991)
42. Sobel, I.: An isotropic 3× 3 image gradient operator, machine vision for three-dimensional scenes (h. freeman editor) (1990)
43. Such, F.P., Madhavan, V., Conti, E., Lehman, J., Stanley, K.O., Clune, J.: Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arxiv (2017), preprint
44. Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. Proc. of the IEEE **89**(9), 1275–1296 (2001)
45. Taylor, R.P., Micolich, A.P., Jonas, D.: Fractal analysis of pollock's drip paintings. Nature **399**(6735), 422–422 (1999)
46. Todd, P.M., Werner, G.M.: Frankensteinian methods for evolutionary music. Musical networks: parallel distributed perception and performace pp. 313–340 (1999)
47. Todd, S., Latham, W.: Evolutionary art & computers. Academic Press, Inc. (1994)
48. Tresset, P., Deussen, O.: Artistically skilled embodied agents. In: AISB (2014)
49. Wilson, D.J.: An experimental investigation of birkhoff's aesthetic measure. The Journal of Abnormal and Social Psychology **34**(3), 390 (1939)

Continuous surrogate-based optimization algorithms are well-suited for expensive discrete problems

Rickard Karlsson, Laurens Bliek, Sicco Verwer, and Mathijs de Weerdt

Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands

Abstract. One method to solve expensive black-box optimization problems is to use a surrogate model that approximates the objective based on previous observed evaluations. The surrogate, which is cheaper to evaluate, is optimized instead to find an approximate solution to the original problem. In the case of discrete problems, recent research has revolved around surrogate models that are specifically constructed to deal with with discrete structures. A main motivation is that literature considers continuous methods, such as Bayesian optimization with Gaussian processes as the surrogate, to be sub-optimal (especially in higher dimensions) because they ignore the discrete structure by e.g. rounding off real-valued solutions to integers. However, we claim that this is not true. In fact, we present empirical evidence showing that the use of continuous surrogate models displays competitive performance on a set of high-dimensional discrete benchmark problems, including a real-life application, against state-of-the-art discrete surrogate-based methods. Our experiments on different discrete structures and time constraints also give more insight into which algorithms work well on which type of problem.

Introduction

A principal challenge in optimization is to deal with black-box objective functions. The objective function is assumed to be unknown in this case, in contrast to traditional optimization that often utilizes an explicit formulation to compute the gradient or lower bounds. Instead, we assume to have an objective $y = f(\mathbf{x}) + \epsilon$ with some unknown function $f(\mathbf{x})$ together with additive noise ϵ. Furthermore, $f(\mathbf{x})$ can be expensive to evaluate in terms of time or another resource which restricts the number of evaluations allowed.

One type of method to solve these black-box optimization problems is the use of surrogate models. Surrogate-based algorithms approximate the objective function in search of the optimal solution, with the benefit that the surrogate model is cheaper to evaluate. Bayesian optimization [22] is an example of such a surrogate-based algorithm.

An active field of research is how to deal with discrete black-box optimization problems with an expensive objective function. There are many real-world examples of this, such as deciding on the architecture of a deep neural network [7] or designing molecules with desirable properties [15]. Furthermore, optimization over structured domains was highlighted as an important problem to address from the NIPS 2017 workshop on Bayesian optimization [10].

Discrete optimization problems can be solved with a continuous surrogate model, e.g. Bayesian optimization with Gaussian processes [22], by ignoring the discrete structure and rounding off the real-valued input to discrete values. However, literature in this field generally considers this to be a sub-optimal approach [1, 8]. Therefore, research has revolved around inherently discrete models such as density estimators or decision trees, e.g. HyperOpt [2] or SMAC [12]. Another approach is to use continuous models that guarantee discrete optimal solutions, such as the piece-wise linear model IDONE [4].

In contrast to common belief, we present an empirical study that displays that continuous surrogate models, in this case Gaussian processes and linear combinations of rectified linear units, show competitive performance on expensive discrete optimization benchmarks by outperforming discrete state-of-the-art algorithms. Firstly, we will introduce the problem, the related work, and the considered benchmark problems. Then, in the remainder of the paper we 1) perform a benchmark comparison between continuous and discrete surrogate-based algorithms on optimization problems with different discrete structures (including one real-life application), 2) investigate why continuous surrogate models perform well by transforming the different discrete problem structures and visualizing the continuous surrogate models, and 3) perform a more realistic analysis that takes the time budget and evaluation time into account when comparing the algorithms. We conclude that continuous surrogates applied to discrete problems should get more attention, and leave some questions for interesting directions of future research in the domain of discrete expensive black-box optimization.

Problem Description

Consider the following class of d-dimensional discrete optimization problems:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathbb{Z}^d \\ & l_i \leq \mathbf{x}_i \leq u_i,\ i = 1, \ldots, d \end{aligned} \tag{1}$$

where l_i and u_i are the lower and upper bound for each integer-valued decision variable x_i. For black-box optimization problems, we assume to have no closed form expression for $f : \mathbb{Z}^d \rightarrow \mathbb{R}$. The only information which can be gathered about f comes from observing the output when evaluating $f(\mathbf{x})$ given some input $\mathbf{x}$. However, in many real-world applications we will also have to deal with some noise $\epsilon \in \mathbb{R}$ such that we are given the output $y = f(\mathbf{x}) + \epsilon$. Obtaining an evaluation is also assumed to be expensive: it could require large computational power, human interaction with the system or time consuming simulations. Therefore it is of interest to obtain a solution within a limited amount of evaluations B, also known as the budget.

One way of solving this class of problems is to make use of a so called surrogate model. A surrogate model is an auxiliary function M that approximates the objective function based on the points evaluated so far. This model is cheaper to evaluate in comparison to the original black-box objective function. Given a number m of already evaluated points, the surrogate model is constructed using the evaluation history

$H = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}$. The surrogate can be utilized to predict promising points to evaluate next on. The next feasible solution $\mathbf{x}^{(m+1)}$ to evaluate on can be chosen based on this prediction. These steps, which are also described in Algorithm 1, are repeated until the budget B is reached.

Typically, an acquisition function $A(M, \mathbf{x})$ is used to propose the next point $\mathbf{x}^{(m+1)}$ to evaluate with the objective function. It predicts how promising a new point $\mathbf{x}$ is, based on a trade-off of exploitation (searching at or near already evaluated points that had a low objective) and exploration (searching in regions where the surrogate has high uncertainty). In general, the next point is chosen by finding the global optimum $\mathbf{x}^{(m+1)} = \underset{\mathbf{x}}{\operatorname{argmax}}\, A(M, \mathbf{x})$.

Algorithm 1 Surrogate-based optimization

Require: budget B, surrogate model M, acquisition function A
1: Initialize $\mathbf{x}^{(1)}$ randomly and an empty set H
2: **for** $m = 1 : B$ **do**
3: $\quad y^{(m)} \leftarrow f(\mathbf{x}^{(m)}) + \epsilon$
4: $\quad H \leftarrow H \cup \{(\mathbf{x}^{(m)}, y^{(m)}))\}$
5: $\quad M \leftarrow$ fit surrogate model using H
6: $\quad \mathbf{x}^{(m+1)} \leftarrow \underset{\mathbf{x}}{\operatorname{argmax}}\, A(M, \mathbf{x})$
7: **end for**
8: **return** optimal $(\mathbf{x}^*, y^*) \in H$

Related Work

Although discrete problem structures are difficult to handle in black-box optimization, multiple approaches have been proposed. A survey by M. Zaefferer [24] presents different strategies for dealing with discrete structures in surrogate-based algorithms. The first strategy is the naive way by simply ignoring the discrete structure. Another strategy is to use inherently discrete models such as tree-based models [2, 12]. These models can however fail if the problem structure is too complex or if there are both discrete and continuous variables involved [24]. Lastly, discrete structures can be dealt with by using a certain mapping. Although this strategy does not apply directly to a surrogate model, a suitable mapping can make the problem easier. For example, encoding integer solutions with a binary representation can be easier for some regression models to handle.

There are also other strategies such as using problem-specific feature extraction or customizing the model. However, these violate the black-box assumption which is why we will not discuss them.

We now discuss several surrogate-based optimization algorithms that can solve the expensive discrete optimization problem in eq. (1) and that also have their code available online.

Bayesian optimization has a long history of success in expensive optimization problems [13], and has been applied in many domains such as chemical design and hyper-

parameter optimization for deep learning [9, 14]. It typically uses a Gaussian process as a surrogate to approximate the expensive objective. Several acquisition functions exist to guide the search, such as Expected Improvement, Upper Confidence Bound, or Thompson sampling [21], information-theoretic approaches such as Predictive Entropy Search [11], or simply the surrogate itself [5, 18]. Though Gaussian processes are typically used on continuous problems, they can be adapted for problems with discrete variables as well. The authors of [8] suggest three possible approaches, namely rounding to the nearest integer 1) when choosing where to evaluate the objective function, 2) when evaluating the objective function, or 3) inside the covariance function of the Gaussian process. The latter provides the best results but gives an acquisition function that is hard to optimize. The first option leads to the algorithm getting stuck by repeatedly evaluating the same points, although this can be circumvented by carefully balancing exploration and exploitation [17]. In this work, we will consider only the simpler second option, for which we do not need to modify any existing implementations.[1]

BOCS[2] [1] transforms the combinatorial problem into one that can be solved with semi-definite programming. It uses Thompson sampling as the acquisition function. However, it suffers from a large time complexity, which was only recently overcome by using a submodular relaxation called the PSR method[3] [6].

COMBO[4] [23] uses an efficient approximation of a Gaussian process with random features, together with Thompson sampling as the acquisition function. Though this gives increased efficiency, COMBO deals with discrete search spaces by iterating over all possible candidate solutions, which is only possible for small-dimensional problems.

HyperOpt[5] [2] makes use of a tree of Parzen estimators as the surrogate model. It can naturally deal with categorical or integer variables, and even with conditional variables that only exist if other variables take on certain values. The algorithm is known to perform especially well on hyperparameter tuning problems with hundreds of dimensions [3]. This is in sharp contrast with Bayesian optimization algorithms using Gaussian processes, which are commonly used on problems with less than 10 dimensions. A possible drawback for HyperOpt is that each dimension is modeled separately, i.e., no interaction between different variables is modeled. HyperOpt uses the Expected Improvement acquisition function.

SMAC[6] [12] is another surrogate-based algorithm that can naturally deal with integer variables. The main reason for this is that the surrogate model used in this algorithm is a random forest, which is an inherently discrete model. A point of critique for SMAC is that the random forests have worse predictive capabilities than Gaussian processes. Nevertheless, like HyperOpt, SMAC has been applied to problems with hundreds of dimensions [16]. SMAC uses the Expected Improvement acquisition function.

[1] We consider the implementation from https://github.com/fmfn/BayesianOptimization in this work, which uses the Upper Confidence Bound acquisition function.
[2] https://github.com/baptistar/BOCS
[3] https://github.com/aryandeshwal/Submodular_Relaxation_BOCS
[4] https://github.com/tsudalab/combo
[5] https://github.com/hyperopt/hyperopt
[6] https://github.com/automl/SMAC3

IDONE[7] [4] uses a linear combination of rectified linear units as its surrogate model. This is a continuous function, yet it has the special property that any local minimum of the model is located in a point where all variables take on integer values. This makes the method suitable for expensive discrete optimization problems, with the advantage that the acquisition function can be optimized efficiently with continuous solvers. IDONE uses the surrogate model itself as the acquisition function, but adds small perturbations to the optimum of the acquisition function to improve its exploration capabilities. Though the method is not as mature as SMAC or HyperOpt, it also has been applied to problems with more than 100 variables [4].

Benchmark problems

We present the four different benchmark problems that are used to compare the surrogate-based algorithms. The purpose of the benchmarks is to compare the discrete surrogate-based algorithms presented in the previous section and investigate which algorithms are most suited for which type of problem.

The benchmarks have been selected to include binary, categorical and ordinal decision variables but also different discrete structures such as sequential or graph-based structures. Since we assume that the evaluation of the objective functions is expensive, we perform the benchmark with a relatively strict budget of at most 500 evaluations. The objective function is evaluated once per iteration in Algorithm 1. Furthermore, we are testing on relatively large problem sizes, ranging from 44 up to 150 decision variables with search spaces of around $\sim 10^{50}$ possibilities. This range is interesting considering that Bayesian optimization using Gaussian processes is typically applied on problems with less than 10 variables.

On top of that, it has been shown that a large dimensionality reduces the importance of choosing a complicated acquisition function [18], which helps us doing a fair comparison between surrogates.

Moreover, we do an analysis of the performance of each algorithm where we limit the allowed time budget instead of the number of evaluations and simulate different evaluation times of the objective functions. The time budget includes both the total time to evaluate the objective function and the computation time of the optimization algorithm. Thus, it puts emphasis on the computation time of the algorithm in addition to their respective sample efficiency.

We present the four benchmark problems in detail below. Note that we present these problems in detail but that they are treated as black boxes by the optimization algorithms.

The Discrete Rosenbrock problem is a d-dimensional, non-convex function, with a curved valley that contains the global optimum defined by the following function:

$$f(\mathbf{x}) = \sum_{i=1}^{d-1} [100(\mathbf{x}_{i+1} - \mathbf{x}_i^2)^2 + (1 - \mathbf{x}_i)^2] \tag{2}$$

[7] https://bitbucket.org/lbliek2/idone

where $\mathbf{x} \in \mathbb{Z}^d$. In the Rosenbrock problem, finding the valley is simple, but finding the global optimum $[1, 1, \ldots, 1]$ is not. As we are exploring discrete optimization problems, we consider a discrete variant of the problem such that only integer solutions are considered. We have $d = 49$ decision variables and each decision variable x_i is bounded by the range $[-5, 10]$. Thus, the problem's search space is in the order of 10^{59} candidate solutions. Lastly, the additive noise ϵ is normally distributed according to $N(\mu = 0, \sigma = 10^{-6})$.

The Weighted Max-Cut problem is an NP-hard graph cutting problem, defined as follows: For an undirected weighted graph $G = (V, E)$, a *cut* in G creates the subset $S \subseteq V$ and its complement $\overline{S} = V \backslash S$. Then $E(S, \overline{S})$, is defined as the set of edges that have one vertex in S and the other in $\overline{S}$. The Max-Cut problem is to find the cut that maximizes the weight of the edges in $E(S, \overline{S})$. The problem is encoded with a binary string $x \in \{0, 1\}^d$ where either $x_i = 0$ or $x_i = 1$ indicates if node i lies in S or $\overline{S}$ respectively.

For the following experiments, the MaxCut problem instances are randomly generated as weighted graphs, with d nodes, edge probability $p = 0.5$ and a uniformly distributed edge weight in the range $[0, 10]$. The graph generator is initialized with the same random seed for every run, ensuring that all experiments of a given problem size are performed on the same graph. On top of that, the additive noise ϵ added to each evaluation is following a standard normal distribution $N(\mu = 0, \sigma = 1)$. Lastly, we are using a graph with $d = 150$ nodes which means that the size of the problem's search space is $2^{150} \approx 10^{57}$.

The Perturbed Traveling Salesman is a variant of the well-known sequential graph problem where, given a number of cities and the distances between these cities, a shortest path needs to be found that visits all cities and returns to the starting city. We consider the asymmetric case with k cities where the distance between cities is not the same in both directions. Moreover, noise $\epsilon \sim U(0, 1)$ is added to each distance during evaluation. While the perturbation can cause issues for problem-specific solvers, it creates a good benchmark for black-box optimization algorithms. To ensure a robust solution, each proposed route is also evaluated 100 times and the worst-case objective value is returned. Furthermore, we will consider problem instance *ftv44*. This is an instance with 44 cities taken from TSPLIB [20], a library of problem instances for the traveling salesman problem. An instance with 44 cities is chosen to closely match the number of decision variables in the ESP problem which has a fixed number of 49 decision variables.

The problem is encoded as in [4]: after choosing a fixed origin city, there are $d = k - 2$ ordered decision variables x_i for $i = 1, \ldots, d$ such that $x_1 \in \{1, 2, \ldots, k - 1\}$ where each integer represents a city other than the origin city. Then, the next decision variable $x_2 \in \{1, ..., k - 2\}$ selects between the cities that were not yet visited. This is repeated until all cities have been chosen in some order. Since the last decision variable $x_d \in \{1, 2\}$ selects between the two remaining cities, we can deduce afterward the two remaining edges which closes the route since there is one last city to visit before returning to the origin city. Thus, the total number of possible sequences is given by $(d - 1)! \approx 6 \cdot 10^{52}$ for this instance.

The Electrostatic Precipitator problem is a real-world industrial optimization problem first published by Rehbach et al. [19]. The Electrostatic Precipitator (ESP) is a crucial component for gas cleaning systems. It is a large device that is used when solid particles need to be filtered from exhaust gases, such as reducing pollution in fossil fueled power plants. Before gas enters the ESP, it passes through a gas distribution system that controls the gas flow into the ESP. The gas flow is guided by configurable metal plates which blocks the airflow to a varying degree. The configuration of these plates inside the gas distribution system is vital for the efficiency of the ESP. However, it is non-trivial to configure this system optimally.

The objective function is computed with a computationally intensive fluid dynamics simulation, taking about half a minute of computation time every time a configuration is tested. There are 49 slots where different types of plates can be placed or be left empty. In total, there are 8 different options available per slot. This is formalized such that each integer-valued solution $\mathbf{x}$ is subject to the inequality constraint $0 \leq \mathbf{x}_i \leq 7$ for $i = 1, \ldots, 49$. This gives a large solution space in the order of 10^{44} possibilities.

Lastly, the problem has some ordinal structure where the decision variables decides between sizes of holes which are covering the plates. However, as an indication of the complex problem structure we have noted that changing any single variable does not affect the objective function.

Experiments

The goal of this section is to show a benchmark comparison between discrete and continuous surrogate-based algorithms on the discrete optimization problems of the previous section. The compared algorithms are HyperOpt and SMAC as two popular surrogate-based algorithms that make use of a discrete surrogate model if the search space is discrete, and Bayesian optimization as a popular surrogate-based algorithm for continuous problems. Though there exist several other algorithms that can deal with the discrete setting, these three are often used in practice because they are well established, can be used for a wide variety of problems, and have code available online. We also include IDONE in the comparisons as a surrogate-based algorithm that uses a continuous surrogate model but is designed for discrete problems, and random search is included as a baseline.

All experiments were run on the same Unix-based laptop with a Dual-Core Intel Core i5 2.7 GHz CPU and 8 GB RAM. Each algorithm attempted to solve the benchmarks 5 times. The allowed evaluation budget was 500 evaluations for all problems except the ESP problem where 100 evaluations were allowed instead due to it being more computationally expensive.

We are using the default hyperparameters for all algorithms, which are decided by their respective code libraries, with two exceptions. We change the SMAC algorithm to deterministic mode, since it otherwise evaluates the same point several times, which deteriorates its performance significantly. Besides that, the first five iterations of IDONE are random evaluations, which is similar to what happens in the other algorithms. The other algorithms start with their default number of random evaluations (which is 5 for Bayesian optimization and 3 for SMAC and HyperOpt), however for a fair comparison

we make sure that all of these initial random evaluations come from a uniform distribution over the search space. Unfortunately, more extensive hyperparameter tuning than stated above is too time-consuming for expensive optimization problems such as ESP.

In the following section we present the results from the benchmark comparison of the four surrogate-based optimization algorithms. The benchmark consists of the four problems which have varying discrete structures.

Results

In this section we describe the main results from comparing the algorithms on the discrete Rosenbrock, weighted Max-Cut, the travelling salesman and the ESP problems. Figure 2 shows the best average objective value found until a given iteration on each problem as well as their respective computation time. The computation time is the cumulative time up until iteration i which is required to perform the steps on line 5 and 6 in Algorithm 1. Furthermore, we also investigate how the algorithms perform if we introduce a time budget during optimization instead of constraining the number of evaluations.

Ordinal structures We start by comparing the results from the 49-dimensional discrete Rosenbrock problem. In Figure 2a, we see that Bayesian optimization (BO) is the only algorithm that comes close to the optimal objective value of zero. The other algorithms are not performing as well, where HyperOpt (HO) gets the closest to BO. Given that the problem is in fact a discrete version of an inherently continuous problem with ordinal variables, this can be considered to be well suited for continuous model regression. On the other hand, IDONE also uses a continuous surrogate, but it does not perform as well as BO. A possible explanation is that IDONE is less flexible since it is a piece-wise linear model.

To investigate the quality of the surrogates from both BO and IDONE, we visualize their surfaces in Figure 1 for the 2-dimensional case of Rosenbrock. The Gaussian process from BO (which uses a Matérn 5/2 kernel in this case) predicts a smoother surface than IDONE which appears more rugged and uneven. Overall, BO looks more similar to the objective ground truth. We can argue that this is why BO performs well while IDONE does not. BO is likely suitable for the discrete Rosenbrock problem since the problem has an underlying continuous structure with ordinal variables. Meanwhile, this structure could be too complex for the piece-wise linear surrogate in IDONE.

However, we are interested in investigating problems which do not necessarily have a clear continuous structure. Thus, we look at the ESP problem which also happens to have some ordinal structure. The results from this problem are found in Figure 2c. It shows a more even performance among the algorithms compared to the Rosenbrock problem, although BO still returns the best objective on average. This is closely followed by both SMAC and HO, while IDONE is doing worse than random search.

Based on the results from these two problems, it appears that BO works well on ordinal structures. However, this does not seem to hold true for all continuous surrogates considering the performance of IDONE. Still, the naive approach with BO outperforms the other state-of-the-art discrete algorithms on the problems that we have discussed so

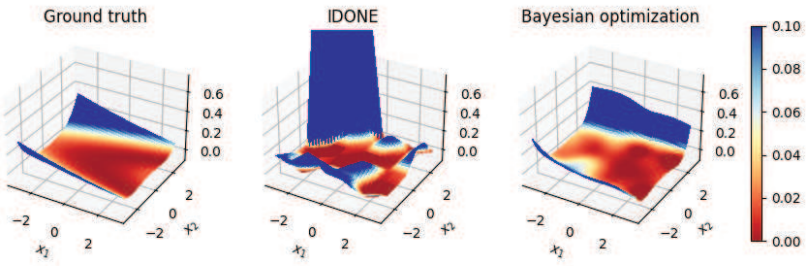
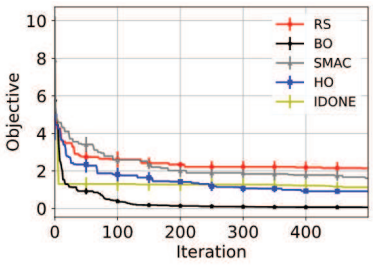


Fig. 1: Visualization of continuous surrogates that approximate the two-dimensional Rosenbrock, namely the linear combination of ReLUs from IDONE and Gaussian processes from BO. These models were picked based on the best performance from 15 different runs with 50 evaluations each. HyperOpt and SMAC are not visualized since this is not supported by their respective code libraries.

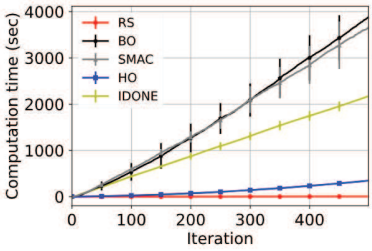
far. This is actually in line with experimental results from [8] on small problems (up to 6 dimensions) with both discrete and continuous parameters, though it was not the main conclusion of the authors. The difference with our work is that we consider purely discrete problems of higher dimensions, from a real-life application, and we include IDONE in the comparison.

Binary structures We will now consider a graph problem, that is the weighted Max-Cut problem. From the results in Figure 2e, we notice that BO clearly outperforms all other algorithms. Meanwhile, IDONE is the second best, followed by SMAC and then HO which performs worse than random search. Compared to the other problems that we have seen so far, a major difference is the binary decision variables in the Max-Cut problem. We use this to frame our hypothesis, namely that the good performance of BO on the Max-Cut problem is due to the binary structure of the problem.

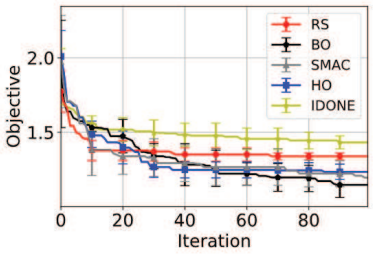
To investigate this hypothesis, we perform an additional experiment by encoding the 49-dimensional, discrete Rosenbrock with binary variables and compare this with the previous results from Figure 2a. The ordinary problem has 49 integer decision variables which lie in the range $[-5, 10]$, this is converted into a total of 196 binary decision variables for the binary-encoded version. Table 1 shows the performance of the algorithms on the binary-encoded, discrete Rosenbrock. Although BO is performing worse on the binarized Rosenbrock, it is still performing the best compared to the other algorithms, even though both SMAC and IDONE perform better on the binarized problem. Thus, we could argue that the binary representation of the Max-Cut problem can not explain



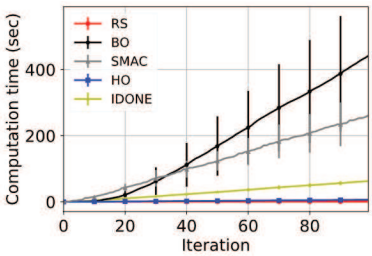
(a) Best average objective value versus iteration on the 49-dimensional discrete Rosenbrock.



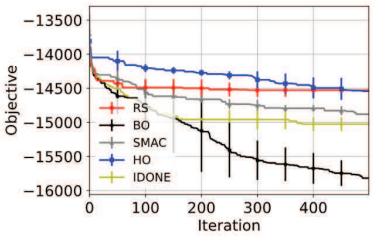
(b) Average computation time versus iteration on the 49-dimensional discrete Rosenbrock.



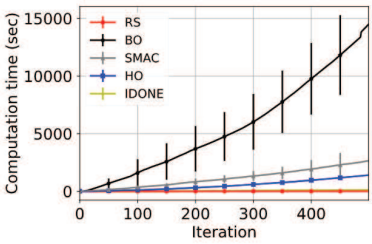
(c) Best average objective value versus iteration on the ESP problem.



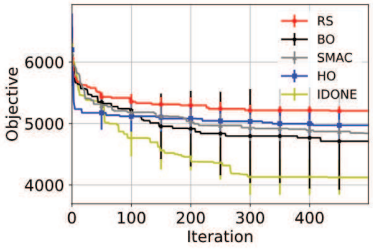
(d) Average computation time versus iteration on the ESP problem.



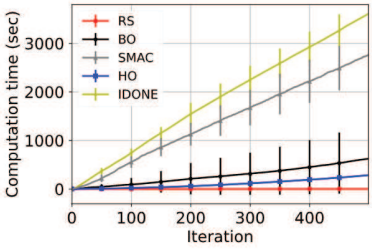
(e) Best average objective value versus iteration on the 150-dimensional weighted Max-Cut.



(f) Average computation time versus iteration on the 150-dimensional weighted Max-Cut.



(g) Best average objective value versus iteration on the TSP with 44 cities.



(h) Average computation time versus iteration on the TSP with 44 cities.

Fig. 2: Comparison of objective value and computation time of Bayesian optimization (BO), SMAC, IDONE, HyperOpt (HO) and random search (RS) on four different benchmark problem. An average is computed from 5 runs and the standard deviation is plotted as the error. The objective value has been negated for Max-Cut since the maximization problem has been turned into a minimization problem. The evaluation budget was 500 evaluations for all problems except the ESP problem which was limited to 100 evaluations due to it being more computationally expensive.

why BO performs well on this problem. There is a possible argument that the binary variables might cause less rounding-off errors since the range of values is simply zero to one with a threshold in the middle. However, a counter-argument is that such a large number of decision variables is typically not well-suited for Gaussian processes regression. This is also indicated by the large computation time of BO on the Max-Cut, see Figure 2f, compared to the other problems as well.

Sequential structures Even though TSP is a graph problem like the Max-Cut problem, there is an important difference. TSP has a sequential structure since the decision variables select an ordering that directly affects the objective value. Moreover, the encoding of the problem, as described in the "Benchmarks problems" section, causes strong interactions between adjacent decision variables.

We continue by looking at the results from TSP in Figure 2g. BO is now outperformed by IDONE even though it still performs better than SMAC and HO on average, although BO has a large variance on this problem. We suspect that the sequential structure is well-suited for IDONE, as it explicitly fits some of its basis functions with adjacent variables in the input vector $(x_1, x_2, \ldots, x_d)$ [4].

To investigate whether this is the case, we test what happens when the order of the decision variables are re-shuffled in TSP such that the sequential structure is removed. This is done by adding to the objective function a mapping that changes the order of the variables in the input vector $(x_1, x_2, \ldots, x_d)$ to a fixed arbitrarily chosen order. From Table 2 we see that IDONE performs worse without the original sequential structure. At the same time, the other algorithms show no large significant difference. However, IDONE returns the best objective on average both with and without shuffling the order of variables. The large variance on BO makes it more difficult to draw any strong conclusions, but since IDONE also uses a continuous surrogate model, we can still conclude that continuous surrogates perform better than the discrete counterparts on this problem.

Algorithm	Non-binary	Binary
BO	**0.067 (0.021)**	**0.37 (0.038)**
SMAC	1.61 (0.18)	1.28 (0.29)
HyperOpt	0.91 (0.13)	0.94 (0.14)
IDONE	1.13 (0.20)	0.61 (0.038)

Table 1: Comparison of results on the 49-dimensional discrete Rosenbrock with and without binary encoding of the decision variables. The final average objective value from 5 runs is presented after 500 evaluations with the standard deviation in parenthesis. The lowest objective value is marked as bold in each column.

Algorithm	Non-shuffled	Shuffled
BO	4713.2 (789.2)	4898.0 (292.4)
SMAC	4841.8 (184.7)	4784.9 (302.7)
HyperOpt	4971.9 (256.5)	4871.8 (221.9)
IDONE	**4122.8 (279.8)**	**4556.4 (175.7)**

Table 2: Comparison of TSP with 44 cities when the input has a sequential structure versus that decision variables' position have been shuffled. The final average objective value from 5 runs is presented after 500 evaluations with the standard deviation in parenthesis. The lowest objective value is marked as bold in each column.

Taking computation time into consideration Although BO performs well on the benchmark comparisons, we are noticing that it is more expensive with respect to compute time compared to the other surrogate-based methods. Figures 2b, 2d, 2f and 2h show the cumulative time on the problems.

In general, BO requires a vast amount of time compared to the other algorithms, especially on Max-Cut where the computations took one to two minutes per iteration. This is not surprising considering that regression with Gaussian processes is computationally intensive: its complexity grows as $O(n^3)$ where n are the number of observations [21]. This can be a big drawback if the evaluation time of the objective function is relatively small.

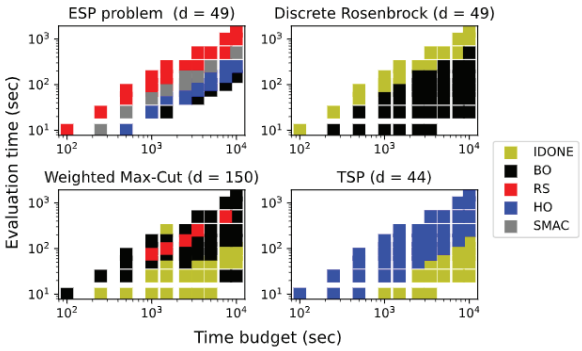


Fig. 3: The best algorithm on average when given a time budget and a fixed evaluation time of the objective function. The results are simulated by adding an artificial evaluation time after running the experiments. For the ESP problem, the actual evaluation time was about $3 \cdot 10^1$ seconds for each function evaluation. The time budget includes both the evaluation time and the compute time of the algorithms.

Meanwhile, the other algorithms share similar computation times which are often less than one second. The only exception is for IDONE which requires more computation time on TSP, see Figure 2h.

So far, we have only considered experiments that restrict the number of evaluations. But in real-life applications, the computation time of an algorithm can be important to take into consideration when limited with some given time budget as well. In particular, the large computation time of BO motivates the question whether it would still perform well under a constrained time budget instead. By keeping track of both the evaluation times of the objective functions, as well as the computation time spent by the algorithms at every iteration, we can investigate the performance of the algorithms in different situations. We artificially adjust the evaluation time in the experiments from Figure 2 to simulate the cost of the objective function. The evaluation time ranges from 10^1 to $1.5 \cdot 10^3$ seconds. Similarly, the time budget varies between 10^2 and 10^4 seconds.

Figure 3 displays which algorithm performs best on average for each problem, depending on the evaluation time and time budget. Notice that the lower triangular shape is caused by the fact that the time budget must be larger than the evaluation time. To ensure a fair comparison, we only present the algorithm with the best final average objective value if the maximum number of evaluations from the previous benchmark experiments was not exceeded within the allocated time budget for all algorithms.

For the ESP problem, the results are mixed. It seems like the best algorithm varies between BO, HO, SMAC and even random search, depending mostly on the ratio between the time budget and the evaluation time. For example, random search performs best when the evaluation time is around the order of 10^1 smaller than the time budget which gives relatively few evaluations. Meanwhile, BO performs best with a much larger ratio. On the discrete Rosenbrock benchmark, BO is clearly the best in almost all cases. The only exception is when the ratio between evaluation time and time budget is very small, in which case IDONE performs better. For the weighted Max-Cut, on the other hand, we notice the opposite of what we see with the Rosenbrock benchmark. Thus, it seems like the growth in compute time of BO, see Figure 2f, sometimes outweighs its good performance which we noted earlier when only taking an evaluation budget into consideration. Lastly, we see that IDONE and HyperOpt outperform other algorithms on TSP when constrained by a time budget.

This experiment gives a better picture of the performance of each algorithm, especially if we may consider it to be more realistic by taking time constraints into consideration. Thus, the experiment from Figure 3 is a good complement to our benchmark comparison. In the following and last section, we summarize the conclusions that can be drawn from all of the above experiments.

Conclusion and Future Work

Based on the results from the benchmark comparison, we can show that the use of continuous surrogate models is a valid approach for expensive, discrete black-box optimization. Moreover, we give insight into what discrete problem structures are well-suited for the different methods.

We have shown that Bayesian optimization (BO) performs better than discrete state-of-the-art algorithms on the four tested, high-dimensional benchmarks problem with either ordinal, sequential or binary structures. IDONE, another continuous surrogate-based algorithm designed for discrete problems, outperforms BO on the benchmark with a sequential structure, but not on the three other benchmarks.

In addition, we have investigated how the different algorithms deal with the different problem structures. Firstly, ordinal structures appear suitable for BO, especially if the objective function has an underlying continuous structure such as the discrete Rosenbrock benchmark. For binary structures, we noticed that BO is negatively affected by binary variables, while IDONE and SMAC benefited from this transformation. However, BO still returned the best solution on the binary Max-Cut problem, even though a big drawback was its computation time. Lastly, we have seen that IDONE outperforms the other algorithms on a problem with sequential decision variables, even after negatively affecting it by changing the ordering.

We also investigated the different algorithms under different time constraints by artificially changing the function evaluation times of the different benchmark problems. For lower time budgets, BO is held back by its large compute time in some cases. Even though BO is a time-intensive method, it mostly showed competitive performance when the evaluation time was relatively low and the time budget high, except for the binary Max-Cut problem. IDONE, HyperOpt, SMAC, and even random search all had specific problems and time budgets where they outperformed other algorithms. Lastly, based on our results, discrete surrogate-based methods could be more relevant in the setting with a limited time budget, in contrast to only limiting the number of evaluations.

Finally, we state some open questions which remain to be answered about continuous surrogates in the topic of expensive, discrete black-box optimization. Considering that we looked at a naive approach of BO, it is still an open question how the more advanced discrete BO variations would fare in the framework where time budgets and function evaluations times are taken into account like in this paper. This same framework would also lead to interesting comparisons between surrogate-based algorithms and other black-box algorithms such as local search or evolutionary algorithms, which are better suited for cheap function evaluations. It also remains unclear why BO performs best on the binary Max-Cut benchmark even though it is negatively affected by binary structures on the Rosenbrock function. Finally, it would be of great practical value if one could decide on the best surrogate-based algorithm in advance, given the time budget and evaluation time of a real-life optimization problem. This research is a first step in that direction.

Acknowledgments

This work is part of the research programme Real-time data-driven maintenance logistics with project number 628.009.012, which is financed by the Dutch Research Council (NWO). The authors would also like to thank Arthur Guijt for helping with the python code.

References

1. Baptista, R., Poloczek, M.: Bayesian optimization of combinatorial structures. In: International Conference on Machine Learning. pp. 471–480 (2018)
2. Bergstra, J., Yamins, D., Cox, D.D.: Hyperopt: A Python library for optimizing the hyperparameters of machine learning algorithms. In: Proceedings of the 12th Python in science conference. pp. 13–20 (2013)
3. Bergstra, J., Yamins, D., Cox, D.D.: Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In: Proceedings of the 30th International Conference on Machine Learning. Jmlr (2013)
4. Bliek, L., Verwer, S., de Weerdt, M.: Black-box combinatorial optimization using models with integer-valued minima. arXiv preprint arXiv:1911.08817 (2019)
5. De Ath, G., Everson, R.M., Rahat, A.A., Fieldsend, J.E.: Greed is good: Exploration and exploitation trade-offs in Bayesian optimisation. arXiv preprint arXiv:1911.12809 (2019)
6. Deshwal, A., Belakaria, S., Doppa, J.R.: Scalable combinatorial Bayesian optimization with tractable statistical models. arXiv preprint arXiv:2008.08177 (2020)

7. Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: A survey. arXiv preprint arXiv:1808.05377 (2018)
8. Garrido-Merchán, E.C., Hernández-Lobato, D.: Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes. Neurocomputing **380**, 20–35 (2020)
9. Griffiths, R.R., Hernández-Lobato, J.M.: Constrained Bayesian optimization for automatic chemical design. arXiv preprint arXiv:1709.05501 (2017)
10. Hernández-Lobato, J.M., Gonzalez, J., Martinez-Cantin, R.: NIPS workshop on Bayesian optimization. https://bayesopt.github.io/, accessed 22-08-2020
11. Hernández-Lobato, J.M., Hoffman, M.W., Ghahramani, Z.: Predictive entropy search for efficient global optimization of black-box functions. In: Advances in neural information processing systems. pp. 918–926 (2014)
12. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: International conference on learning and intelligent optimization. pp. 507–523. Springer (2011)
13. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global optimization **13**(4), 455–492 (1998)
14. Klein, A., Falkner, S., Bartels, S., Hennig, P., Hutter, F.: Fast Bayesian optimization of machine learning hyperparameters on large datasets. In: Artificial Intelligence and Statistics. pp. 528–536 (2017)
15. Korovina, K., Xu, S., Kandasamy, K., Neiswanger, W., Poczos, B., Schneider, J., Xing, E.: Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations. In: International Conference on Artificial Intelligence and Statistics. pp. 3393–3403. PMLR (2020)
16. Lindauer, M., Hutter, F.: Warmstarting of model-based algorithm configuration. arXiv preprint arXiv:1709.04636 (2017)
17. Luong, P., Gupta, S., Nguyen, D., Rana, S., Venkatesh, S.: Bayesian optimization with discrete variables. In: Australasian Joint Conference on Artificial Intelligence. pp. 473–484. Springer (2019)
18. Rehbach, F., Zaefferer, M., Naujoks, B., Bartz-Beielstein, T.: Expected improvement versus predicted value in surrogate-based optimization. arXiv preprint arXiv:2001.02957 (2020)
19. Rehbach, F., Zaefferer, M., Stork, J., Bartz-Beielstein, T.: Comparison of parallel surrogate-assisted optimization approaches. In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 1348–1355. GECCO '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3205455.3205587
20. Reinelt, G.: TSPlib. http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html, accessed 31-07-2020
21. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., de Freitas, N.: Taking the human out of the loop: A review of Bayesian optimization. Proceedings of the IEEE **104**(1), 148–175 (2016)
22. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems. pp. 2951–2959 (2012)
23. Ueno, T., Rhone, T.D., Hou, Z., Mizoguchi, T., Tsuda, K.: Combo: An efficient Bayesian optimization library for materials science. Materials discovery **4**, 18–21 (2016)
24. Zaefferer, M.: Surrogate Models For Discrete Optimization Problems. Ph.D. thesis, Technischen Universität Dortmund (2018)

Comparing Correction Methods to Reduce Misclassification Bias

Kevin Kloos[1,5 [0000−0001−6980−4259]], Quinten Meertens[3,4,5 [0000−0002−3485−8895]], Sander Scholtus[5 [0000−0002−8316−8938]], and Julian Karch[2 [0000−0002−1625−2822]]

[1] Mathematical Institute, Leiden University, the Netherlands
[2] Institute of Psychology, Leiden University, the Netherlands
[3] Leiden Centre of Data Science, Leiden University, the Netherlands
[4] Center for Nonlinear Dynamics in Economics and Finance, University of Amsterdam, the Netherlands
[5] Statistics Netherlands, The Hague, the Netherlands †

Abstract. When applying supervised machine learning algorithms to classification, the classical goal is to reconstruct the true labels as accurately as possible. However, if the predictions of an accurate algorithm are aggregated, for example by counting the predictions of a single class label, the result is often still statistically biased. Implementing machine learning algorithms in the context of official statistics is therefore impeded. The statistical bias that occurs when aggregating the predictions of a machine learning algorithm is referred to as misclassification bias. In this paper, we focus on reducing the misclassification bias of binary classification algorithms by employing five existing estimation techniques, or estimators. As reducing bias might increase variance, the estimators are evaluated by their mean squared error (MSE). For three of the estimators, we are the first to derive an expression for the MSE in finite samples, complementing the existing asymptotic results in the literature. The expressions are then used to compute decision boundaries numerically, indicating under which conditions each of the estimators is optimal, i.e., has the lowest MSE. Our main conclusion is that the calibration estimator performs best in most applications. Moreover, the calibration estimator is unbiased and it significantly reduces the MSE compared to that of the uncorrected aggregated predictions, supporting the use of machine learning in the context of official statistics.‡

Keywords: Bias Correction · Misclassification Bias · Supervised Machine Learning · Classification · Official Statistics

†Corresponding authors: k.kloos@cbs.nl, q.a.meertens@uva.nl

‡The views expressed in this paper are those of the authors and do not necessarily reflect the policy of Statistics Netherlands. The authors would like to thank Arnout van Delden and three anonymous referees for their useful comments on previous versions of this paper.

1 Introduction

Currently, many researchers in the field of official statistics are examining the potential of machine learning algorithms. A typical example is estimating the proportion of houses in the Netherlands having solar panels, by employing a machine learning algorithm trained to classify satellite images [3]. However, as long as the algorithm's predictions are not error-free, the estimate of the relative occurrence of a class, also known as the *base rate*, can be biased [17,18]. This fact is also intuitively clear: if the number of false positives does not equal the number of false negatives, then the estimate of the base rate is biased, even if the false positive rate and false negative rate are both small. The statistical bias that occurs when aggregating the predictions of a machine learning algorithm is referred to as *misclassification bias* [5].

Misclassification bias occurs in a broad range of applications, including official statistics [13], land cover mapping [12], political science [9,21], and epidemiology [8]. The objective in each of these applications is to minimize a loss function at the level of aggregated predictions, in contrast to minimizing a loss function at the level of individual predictions. Within the field of machine learning, learning with that objective is referred to as quantification learning; see [6] for a recent overview. In quantification learning, the idea is not to train a classifier at all, but to directly estimate the base rate from the feature distribution. A drawback of that approach is that relatively large training and test datasets are needed to optimize hyperparameters and to obtain accurate estimates of the accuracy of the prediction, respectively. In the applications referred to before, labelled data are often expensive to obtain and therefore scarce. Hence, in this paper, we focus on what is referred to as quantifiers based on corrected classifiers [6]. In short, it entails that we first aggregate predictions of classification algorithms and then correct the aggregates in order to reduce misclassification bias.

In the literature on measurement error, several methods have been proposed to reduce misclassification bias when aggregating categorical data that is prone to measurement error; see [11] for a technical discussion and [1] for a more recent overview. Based on that literature, we propose a total of five estimators for the base rate that can be derived from the confusion matrix of a classification algorithm. As reducing bias might increase variance, the estimators are evaluated by their mean squared error (MSE). To the best of our knowledge, for three of the five estimators, only asymptotic expressions for the MSE are ever presented in the literature. In this paper, we derive the expressions for the MSE for finite datasets. As a first step, we restrict ourselves to binary classification problems. Nonetheless, we believe that the same proof strategies may be used for multi-class classification problems. The expressions for the MSE enable a theoretical comparison of the five estimators for finite datasets. It allows us, for the first time, to make solid recommendations on how to employ classification algorithms in official statistics and other disciplines interested in aggregate statistics.

The remainder of the paper is organized as follows. First, in Section 2, the five estimators are formally introduced and the mathematical expressions for their MSEs are presented. The derivations are included in the appendix. Then, in

Section 3, the decision boundaries are numerically derived. We can indicate under which condition, like the sensitivity and specificity of the learning algorithm and the size of the test set, each of the estimators has the lowest MSE. Finally, in Section 4, we draw our main conclusion and discuss directions for future research.

2 Methods

Consider a *target population* of N objects and assume that the objects can be separated into two classes. One of the two classes is the *class of interest*. We refer to the relative occurrence of the class of interest in the target population as the *base rate* and we denote that parameter by α. In the example mentioned in Section 1, the objects are houses in the Netherlands and the two classes are whether or not the house has solar panels on the roof [3]. The class of interest is having solar panels and hence α indicates the relative frequency of houses in the country having solar panels.

We assume that the true classifications are only known for objects in a small simple random sample of the target population. In the applications that we consider, these classifications are obtained by manual inspection of the objects in that sample. Objects that belong to the class of interest receive class label 1, the other objects receive class label 0. Then, the sample is split randomly into a training set and a test set. As usual, the training set is used for model selection through cross-validation and is then used to train the selected model. We will consider the result of that part of the process as given. The test set is used to estimate the classification performance of the trained algorithm, which we will discuss in more detail below. Finally, the classification algorithm is applied on the entire target population (minus the small random sample, but we will neglect that small difference) resulting in a predicted label for each object.

As we will encounter in Subsection 2.2, simply computing the relative occurrence of objects predicted to belong to the class of interest will result in a biased estimate of α. That bias is referred to as *misclassification bias* [4]. In this section, five estimators for the base rate parameter α are formally introduced, many of which have been proposed decades ago; see [11] for an extensive discussion. We summarize the formulas for bias and variance that can be found in the literature and complement them with our own derivations.

In order to correct for misclassification bias, we need estimates of the algorithm's (mis)classification probabilities. Following [20], we assume that misclassifications are independent across objects and that the (mis)classification probabilities are the same for each object, conditional on their true class label. With this classification-error model in mind, we denote the probability that the algorithm predicts an object of class 0 correctly by p_{00} and we define p_{11} analogously. Observe that p_{11} and p_{00} correspond to the algorithm's sensitivity and specificity, respectively. The *confusion matrix* $\mathbf{P}$ is then defined as follows:

$$\mathbf{P} = \begin{pmatrix} p_{00} & 1 - p_{00} \\ 1 - p_{11} & p_{11} \end{pmatrix}. \tag{1}$$

Table 1: Contingency tables for test set (left) and target population (right)

(a)

		Estimated class		
		0	1	Tot
True class	0	n_{00}	n_{01}	n_{0+}
	1	n_{10}	n_{11}	n_{1+}
	Tot	n_{+0}	n_{+1}	n

(b)

		Estimated class		
		0	1	Tot
True class	0	N_{00}	N_{01}	N_{0+}
	1	N_{10}	N_{11}	N_{1+}
	Tot	N_{+0}	N_{+1}	N

The classification probabilities p_{00} and p_{11} are not known, but will be estimated using the test set. We write n for the size of the test set and introduce the notation n_{ij} and N_{ij} as depicted in Table 1. The classification probabilities are then estimated without bias by $\hat{p}_{00} = n_{00}/n_{0+}$ and $\hat{p}_{11} = n_{11}/n_{1+}$. (Here, the assumption is needed that the test set is a simple random sample from the target population.) Furthermore, the base rate α for the target population is defined formally as $\alpha = N_{1+}/N$.

Finally, we make the following technical assumptions. We assume that the algorithm is not perfect in predicting either of the classes, but that it is better than guessing for both of the classes, i.e., we assume that $0.5 < p_{ii} < 1$. Because the test set is a small (i.e., $n \ll N$) simple random sample from the population, n_{0+} may be assumed to follow a $Bin(n, \alpha)$-distribution, since α is considered fixed. Moreover, the classification-error model that we assume implies that the elements in the rows in Table 1, conditional on the corresponding row total, follow a binomial distribution as well, with the corresponding classification probability as success probability. For example, to name just two out of the eight entries, $n_{00} \mid n_{0+} \sim Bin(n_{0+}, p_{00})$ and $N_{10} \mid N_{1+} \sim Bin(N_{1+}, 1-p_{11})$. Last, the assumption $n \ll N$ justifies our ultimate technical assumption, which is that the estimators for the entries in $\mathbf{P}$ based on the test set on the one hand and estimators for α based only on the predicted class labels for the target population on the other hand, are independent random variables.

2.1 Baseline estimator - random sample

The baseline estimator for α is the proportion of data points in the test dataset for which the observed class label is equal to 1. The baseline estimator will be denoted by $\hat{\alpha}_a$. Under the assumptions discussed above, it is immediate that $\hat{\alpha}_a$ is an unbiased estimator for α, i.e.:

$$B\left[\hat{\alpha}_a\right] = 0. \tag{2}$$

Since we have assumed that the size n of the test dataset is much smaller than the size N of the population data, we may approximate the distribution of $n\hat{\alpha}_a$ by a binomial distribution with success probability α. The variance, and hence the MSE, of $\hat{\alpha}_a$ is then given by

$$MSE\left[\hat{\alpha}_a\right] = V\left[\hat{\alpha}_a\right] = \frac{\alpha(1-\alpha)}{n}. \tag{3}$$

This MSE will serve as the baseline value for the other estimators we discuss.

2.2 Classify and count

When applying a trained machine learning algorithm on new data, we may simply count the number of data points for which the predicted class equals 1. The resulting estimator for α, which we will denote by $\hat{\alpha}^*$, is referred to as the 'classify-and-count' estimator, see [6]. In general, the classify-and-count estimator is (strongly) biased, and has almost zero variance. More specifically,

$$E\left[\hat{\alpha}^*\right] = \alpha p_{11} + (1-\alpha)(1-p_{00}), \tag{4}$$

and hence

$$B\left[\hat{\alpha}^*\right] = \alpha(p_{11}-1) + (1-\alpha)(1-p_{00}), \tag{5}$$

which is zero only if the point (p_{00}, p_{11}) lies on the line through $(1-\alpha, \alpha)$ and $(1,1)$ in $\mathbb{R}^2$, as shown in [17]. The variance of the classify-and-count estimator is derived in [2] and equals

$$V\left[\hat{\alpha}^*\right] = \frac{\alpha p_{11}(1-p_{11}) + (1-\alpha)p_{00}(1-p_{00})}{N}. \tag{6}$$

If the population size N is large, the variance of $\hat{\alpha}^*$ is low. In some literature, this low variance is misinterpreted as high accuracy, by claiming intuitively that the large size of the dataset implies that the noise cancels out (cf. [16]). However, the nonzero bias is neglected in such arguments. Therefore, we are interested in the MSE because it considers both bias and variance. It equals

$$MSE\left[\hat{\alpha}^*\right] = \Big[\alpha(p_{11}-1) + (1-\alpha)(1-p_{00})\Big]^2 + O\left(\frac{1}{N}\right). \tag{7}$$

Here and below, the notation $O(1/x)$ indicates a remainder term that, for sufficiently large values of $x > 0$, is always contained inside an interval $(-C/x, C/x)$ for some constant $C > 0$; see, e.g., [19, p. 147]. Observe how, in general, the MSE does not converge to 0 as N tends to ∞.

2.3 Subtracting estimated bias

Knowing that the classify-and-count estimator $\hat{\alpha}^*$ is biased (see (5)), we may attempt to estimate that bias and subtract it from $\hat{\alpha}^*$. As briefly mentioned in [17], we may estimate that bias by the plug-in estimator, that is, we substitute the unknown quantities in Equation (5) by their estimates. More precisely, the bias is estimated as

$$\widehat{B}\left[\hat{\alpha}^*\right] = \hat{\alpha}^*(\hat{p}_{00} + \hat{p}_{11} - 2) + (1-\hat{p}_{00}), \tag{8}$$

in which the estimators $\hat{p}_{00}$ and $\hat{p}_{11}$ are based on the test dataset. The resulting estimator $\hat{\alpha}_b$ for α equals

$$\hat{\alpha}_b = \hat{\alpha}^* - \widehat{B}\left[\hat{\alpha}^*\right] = \hat{\alpha}^*(3 - \hat{p}_{00} - \hat{p}_{11}) - (1-\hat{p}_{00}). \tag{9}$$

To the best of our knowledge, the bias and variance of the estimator $\hat{\alpha}_b$ have not been published in the scientific literature. Therefore, we have derived both, up to terms of order $1/n^2$, yielding the following result.

Theorem 1. *The bias of $\hat{\alpha}_b$ as estimator for α is given by*

$$B\left[\hat{\alpha}_b\right] = (1-p_{00})(2-p_{00}-p_{11}) - \alpha(p_{00}+p_{11}-2)^2. \tag{10}$$

The variance of $\hat{\alpha}_b$ equals

$$\begin{aligned} V\left[\hat{\alpha}_b\right] = & \frac{\left[\alpha(p_{00}+p_{11}-1)-p_{00}\right]^2 p_{00}(1-p_{00})}{n(1-\alpha)}\left(1+\frac{\alpha}{n(1-\alpha)}\right) \\ & + \frac{\left[\alpha(p_{00}+p_{11}-1)+(1-p_{00})\right]^2 p_{11}(1-p_{11})}{n\alpha}\left(1+\frac{1-\alpha}{n\alpha}\right) \\ & + O\left(\max\left[\frac{1}{n^3},\frac{1}{N}\right]\right). \end{aligned} \tag{11}$$

Proof. See the Appendix.

In particular, Theorem 1 implies that $B\left[\hat{\alpha}_b\right] = (2-p_{00}-p_{11})B\left[\hat{\alpha}^*\right]$, compare Equations (10) and (5). Hence, $|B\left[\hat{\alpha}_b\right]| \leq |B\left[\hat{\alpha}^*\right]|$, because $1 < p_{00}+p_{11} < 2$.

2.4 Misclassification probabilities

Let $\mathbf{P}$ be the row-normalized confusion matrix of the machine learning algorithm that we have trained, as defined in (1). That is, entry p_{ij} is the probability that the algorithm predicts class j for a data point that belongs to class i. The probabilities p_{ij} are referred to as misclassification probabilities. In the binary setting, we write $\boldsymbol{\alpha}$ for the column vector $(1-\alpha,\alpha)^T$ (similarly for $\hat{\boldsymbol{\alpha}}^*$). Under the assumption that the probabilities p_{ij} are identical for each data point, we obtain the expression $E[\hat{\boldsymbol{\alpha}}^*] = \mathbf{P}^T\boldsymbol{\alpha}$. If the true values of all entries p_{ij} of $\mathbf{P}$ were known and if $p_{00}+p_{11} \neq 1$, then $\hat{\boldsymbol{\alpha}}_p = (\mathbf{P}^T)^{-1}\hat{\boldsymbol{\alpha}}^*$ would be an unbiased estimator for α. Using the plug-in estimator $\hat{\mathbf{P}}$ for $\mathbf{P}$, estimated on the test set, the following estimator for α is obtained:

$$\hat{\alpha}_p = \frac{\hat{\alpha}^* + \hat{p}_{00} - 1}{\hat{p}_{00}+\hat{p}_{11}-1}. \tag{12}$$

It is known that the estimator $\hat{\alpha}_p$ is consistent (asymptotically unbiased) for α, see [1]. In [7], the variance of this estimator is analysed for an arbitrary number of classes. For the binary case, a simple analytic expression for the bias and variance of $\hat{\alpha}_p$ for finite datasets has not been given, as far as we know. Therefore, we have derived the bias and variance for finite datasets, yielding the following result.

Theorem 2. *The bias of $\hat{\alpha}_p$ as estimator for α is given by*

$$B\left[\hat{\alpha}_p\right] = \frac{p_{00} - p_{11}}{n(p_{00} + p_{11} - 1)} + O\left(\frac{1}{n^2}\right). \tag{13}$$

The variance of $\hat{\alpha}_p$ is given by

$$V[\hat{\alpha}_p] = \frac{(1-\alpha)p_{00}(1-p_{00})\left[1 + \frac{\alpha}{n(1-\alpha)}\right] + \alpha p_{11}(1-p_{11})\left[1 + \frac{1-\alpha}{n\alpha}\right]}{n(p_{00} + p_{11} - 1)^2} + O\left(\max\left[\frac{1}{n^2}, \frac{1}{N}\right]\right). \tag{14}$$

Proof. See the Appendix.

2.5 Calibration probabilities

Let $\mathbf{C}$ be the column-normalized confusion matrix of the machine learning algorithm that we have trained. That is, entry c_{ij} is the probability that the true class of a data point is j given that the algorithm has predicted class i. The probabilities c_{ij} are referred to as calibration probabilities [11]. The first element of the vector $\mathbf{C}\hat{\boldsymbol{\alpha}}^*$ is an unbiased estimator for $\boldsymbol{\alpha}$, if $\mathbf{C}$ is known.

Using the plug-in estimator $\hat{\mathbf{C}}$ for $\mathbf{C}$, which is estimated on the test dataset analogously to $\hat{\mathbf{P}}$, the following estimator $\hat{\alpha}_c$ for α is obtained:

$$\hat{\alpha}_c = \hat{\alpha}^* \frac{n_{11}}{n_{+1}} + (1 - \hat{\alpha}^*)\frac{n_{10}}{n_{+0}}, \tag{15}$$

in which each n_{ij} and n_{+j} should be considered as random variables. It has been shown that $\hat{\alpha}_c$ is a consistent estimator for α [1]. Under the assumptions we have made in this paper, it can be shown that $\hat{\alpha}_c$ is in fact an unbiased estimator for α. To the best of our knowledge, we are also the first to give an approximation (up to terms of order $1/n^2$) of the variance of $\hat{\alpha}_c$. Both results are summarized in the following theorem.

Theorem 3. *The calibration estimator $\hat{\alpha}_c$ is an unbiased estimator for α:*

$$B\left[\hat{\alpha}_c\right] = 0. \tag{16}$$

The variance of $\hat{\alpha}_c$ is equal to the following expression:

$$\begin{aligned} V(\hat{\alpha}_c) = &\left[\frac{(1-\alpha)(1-p_{00}) + \alpha p_{11}}{n} + \frac{(1-\alpha)p_{00} + \alpha(1-p_{11})}{n^2}\right] \\ &\times \left[\frac{\alpha p_{11}}{(1-\alpha)(1-p_{00}) + \alpha p_{11}}\left(1 - \frac{\alpha p_{11}}{(1-\alpha)(1-p_{00}) + \alpha p_{11}}\right)\right] \\ &+ \left[\frac{(1-\alpha)p_{00} + \alpha(1-p_{11})}{n} + \frac{(1-\alpha)(1-p_{00}) + \alpha p_{11}}{n^2}\right] \\ &\times \left[\frac{(1-\alpha)p_{00}}{(1-\alpha)p_{00} + \alpha(1-p_{11})}\left(1 - \frac{(1-\alpha)p_{00}}{(1-\alpha)p_{00} + \alpha(1-p_{11})}\right)\right] \\ &+ O\left(\max\left[\frac{1}{n^3}, \frac{1}{Nn}\right]\right). \end{aligned} \tag{17}$$

Proof. See the Appendix.

Hereby, the overview of the five estimators for α is complete. The expressions that we have derived for the bias and variance of these five estimators will now be used to compare the (root) mean squared error of the five estimators, both theoretically as well as by means of simulation studies.

3 Results

The aim of this section is to derive empirically which of the five estimators of α that we presented in Section 2 has the lowest MSE, and under which conditions. For a given population size N, the MSE of each estimator depends on four parameters (i.e, $\alpha, p_{00}, p_{11}, n$), so visualizations would have to be 5-dimensional. To reduce dimensions, we will first present a simulation study in which all four parameters are fixed. For the fixed parameter setting, the sampling distributions of the estimators are compared using boxplots. Second, we will fix several values of α and n and use plots to compare the MSE of the estimators for varying p_{00} and p_{11}. The latter analysis will already be sufficient in order to reach a final conclusion on which estimator has the lowest MSE.[‖]

3.1 Sampling distributions of the estimators

Here, we present two simple simulation studies to gain some intuition for the difference in the sampling distributions of the five estimators. In the first simulation study, we consider a class-balanced dataset, that is, $\alpha = 0.5$, with a small test dataset of size $n = 1000$, a large population dataset $N = 3 \times 10^5$ and a rather poor classifier having classification probabilities $p_{00} = 0.6$ and $p_{11} = 0.7$. We deliberately choose $p_{00} \neq p_{11}$, as otherwise the classify-and-count estimator $\hat{\alpha}^*$ would be unbiased: (p_{00}, p_{11}) would be on the line between $(1-\alpha, \alpha)$ and $(1, 1)$, see also Equation (5).

Table 2 summarizes the bias, variance and root mean squared error (RMSE), computed using the analytic approximations presented in Section 2. The classify-and-count estimator is highly biased and therefore it has a high RMSE, despite having the lowest variance of all estimators. The RMSE of the classify-and-count estimator can indeed be improved by subtracting an estimate of the bias ($\hat{\alpha}_b$). The subtraction reduces the absolute bias and only slightly increases the variance. A further bias reduction is obtained by the misclassification estimator $\hat{\alpha}_p$. However, inverting the row-normalized confusion matrix $\mathbf{P}$ (that is, the misclassification probabilities) for values of p_{00} and p_{11} close to $p_{00}+p_{11} = 1$ significantly increases the variance of the estimator, leading to the highest RMSE of all estimators considered. Finally, the calibration estimator $\hat{\alpha}_c$ is unbiased and has

[‖]The results in this section have been obtained using the statistical software R. All visualizations have been implemented in a Shiny dashboard, which in addition includes interactive 3D-plots of the RMSE surface for each of the estimators. The code can be retrieved from `https://github.com/kevinkloos/Misclassification-Bias`.

the lowest variance among the estimators that make use of the test dataset. In particular, note that the variance is also lower than that of the baseline estimator. In this example, the estimator based on the calibration probabilities has the lowest RMSE, and it is the only estimator with a lower RMSE than the baseline estimator $\hat{\alpha}_a$.

Table 2: A comparison of the bias, variance and RMSE of each of the five estimators for α, where $\alpha = 0.5$, $p_{00} = 0.6$, $p_{11} = 0.7$, $n = 1000$ and $N = 3 \times 10^5$.

Estimator	*Symbol*	Bias $\times 10^{-2}$	Variance $\times 10^{-4}$	RMSE $\times 10^{-2}$
Baseline	$\hat{\alpha}_a$	0.000	2.500	1.581
Classify-and-count	$\hat{\alpha}^*$	5.000	0.000	5.000
Subtracted-bias	$\hat{\alpha}_b$	-3.500	2.244	3.807
Misclassification	$\hat{\alpha}_p$	-0.033	25.025	5.003
Calibration	$\hat{\alpha}_c$	0.000	2.275	1.508

To gain insight in the sampling distribution of the estimators, in addition to the metrics presented in Table 2, we simulated a large number $R = 10000$ of confusion matrices for datasets of size $n = 1000$ and $N = 3 \times 10^5$. Each confusion matrix was created as follows. First, take a random draw from a $Bin(N, \alpha)$-distribution, resulting in a number N_{1+}. Then, take a random draw from a $Bin(N_{1+}, p_{11})$-distribution to obtain N_{11} and a random draw from a $Bin(N - N_{+1}, p_{00})$-distribution to obtain N_{00}. This computes the theoretical confusion matrix for the target population. Use this confusion matrix to draw a sample from a multivariate hypergeometric distribution, with its parameters from the drawn theoretical confusion matrix. These draws precisely give the number of true and false positives and negatives needed to fill a confusion matrix. Each confusion matrix can be used to compute the five estimators. Repeating this procedure $R = 10000$ times gave rise to the sampling distributions of the five estimators as presented in Figure 1. It nicely visualizes the bias and variance of the five estimators, supporting the results in Table 2. In addition, it shows that, due to the bias, the variances of the classify-and-count estimator $\hat{\alpha}^*$ and the subtracted-bias estimator $\hat{\alpha}_b$ cannot be used to obtain reliable confidence intervals for α.

In the second simulation study, we consider a highly imbalanced dataset, namely $\alpha = 0.98$. We again assume that the available test dataset has size $n = 1000$, but we assume a classifier having classification probabilities $p_{00} = 0.94$ and $p_{11} = 0.97$. Table 3 summarizes the bias, variance and RMSE of each of the estimators and Figure 2 shows the sampling distributions of each of the estimators. It can be noticed that subtracted-bias estimator and the misclassification estimator both have estimates of α that exceed 1. It is obvious that such values

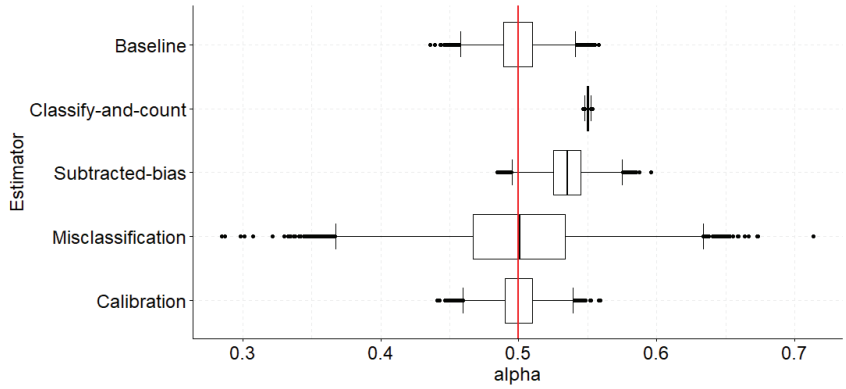


Fig. 1: The boxplots show the sampling distribution of the estimators for α, where $\alpha = 0.5$, $p_{00} = 0.6$, $p_{11} = 0.7$, $n = 1000$ and $N = 3 \times 10^5$. The true value of α is highlighted by a vertical line.

cannot occur in the population. For the method with the misclassification probabilities, this effect gets stronger when $p_{00} + p_{11}$ gets closer to 1. Furthermore, the baseline estimator performs well compared to the other estimators when the dataset is highly imbalanced: its RMSE is slightly higher than the RMSE of the method with calibration probabilities and much lower than the method with the misclassification probabilities. Finally, it is shown that the classify-and-count estimator is highly biased, even though p_{00} and p_{11} are both fairly close to 1.

Table 3: A comparison of the bias, variance and RMSE of each of the five estimators for α, where $\alpha = 0.98$, $p_{00} = 0.94$, $p_{11} = 0.97$, $n = 1000$ and $N = 3 \times 10^5$.

Method	*Symbol*	Bias $\times 10^{-2}$	Variance $\times 10^{-5}$	RMSE $\times 10^{-3}$
Baseline	$\hat{\alpha}_a$	0.000	1.960	4.427
Classify-and-count	$\hat{\alpha}^*$	-2.820	0.000	28.200
Subtracted-bias	$\hat{\alpha}_b$	0.254	3.377	6.342
Misclassification	$\hat{\alpha}_p$	-0.003	3.587	5.989
Calibration	$\hat{\alpha}_c$	0.000	1.289	3.591

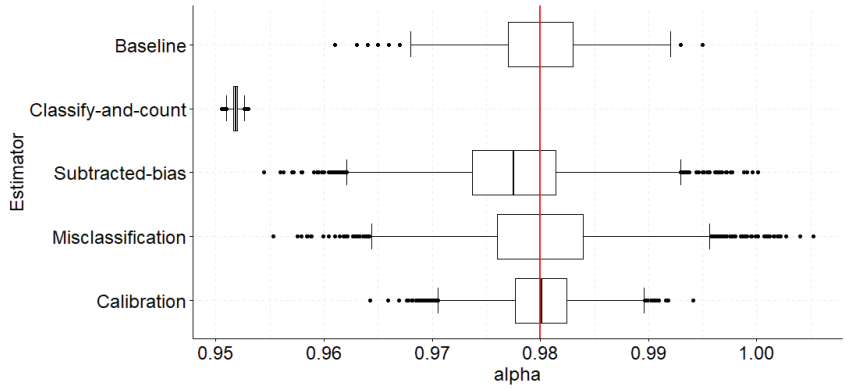


Fig. 2: The boxplots show the sampling distribution of the estimators for α, where $\alpha = 0.98$, $p_{00} = 0.94$, $p_{11} = 0.97$, $n = 1000$ and $N = 3 \times 10^5$. The true value of α is highlighted by a vertical line.

3.2 Finding the optimal estimator

The aim of this subsection is to find the optimal estimator, i.e., the estimator with the lowest RMSE, for every combination of values of the parameters α, p_{00}, p_{11} and n. First, suppose that (p_{00}, p_{11}) is close to the line in the plane through the points $(1-\alpha, \alpha)$ and $(1, 1)$. As noted before, it implies that the classify-and-count estimator $\hat{\alpha}^*$ has low bias. Consequently, the subtracted-bias estimator $\hat{\alpha}_b$ has low bias as well. Thus, these two estimators will have the lowest RMSE in the described region, whose size decreases as n increases. Figure 3 visualizes the described region for $\alpha = 0.2$ and two different values of n. We remark that the biased estimators $\hat{\alpha}^*$ and $\hat{\alpha}_b$ perform worse (relative to the other estimators) when the sample size n of the test dataset increases. The biased methods, like Classify-and-count and Subtracted-bias, perform well when the classification probabilities are high for the largest group.

As we have seen in both Table 2 and Table 3, the calibration estimator $\hat{\alpha}_c$ competes with the baseline estimator in having the lowest RMSE. In general, the calibration estimator will have lower RMSE if the classification probabilities p_{00} and p_{11} are higher, while the baseline estimator does not depend on these classification probabilities. In a neighbourhood of $p_{00} = p_{11} = 0.5$, the baseline estimator will always have lower RMSE than the calibration estimator. However, for every α and n, there must exist a curve in the (p_{00}, p_{11})-plane beyond which the calibration estimator will have lower RMSE than the baseline estimator. The left-hand panels in Figure 4 show this curve for $\alpha = 0.2$ and two different values of n. For larger values of n, the curve where the calibration estimator performs better than the baseline estimator gets closer to $p_{00} = p_{11} = 0.5$ and therefore covers a larger area in the (p_{00}, p_{11})-plane.

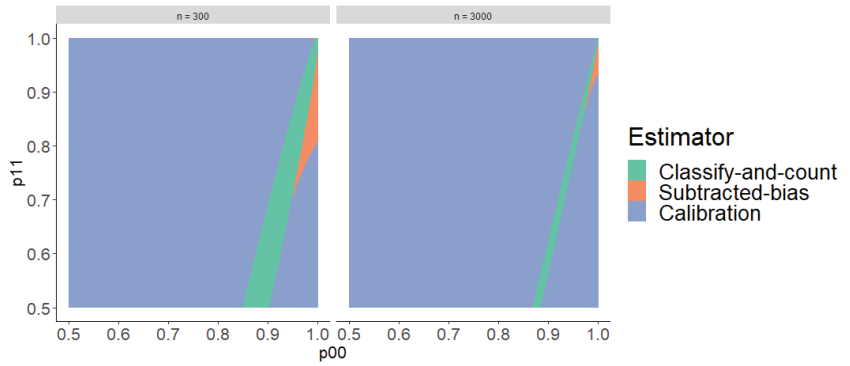


Fig. 3: For each coordinate (p_{00}, p_{11}), the depicted color indicates which estimator has the lowest RMSE, considering only the classify-and-count estimator (green), the subtracted-bias estimator (orange) and the calibration estimator (purple). In the left panel , we have set $\alpha = 0.2$ and $n = 300$, whereas $\alpha = 0.2$ and $n = 3000$ in the right panel. The red and green regions are smaller in the right panel, as the variance of the calibration estimator is decreasing in n, while the bias of the classify-and-count estimator and of the subtracted-bias estimator do not depend on n.

Table 2 and Table 3 have shown that the misclassification estimator only performs well if p_{00} and p_{11} are high, which is confirmed by the expression of the bias and variance: both have a singularity at $p_{00} + p_{11} = 1$, see Equations (13) and (14). The right-hand panels in Figure 4 show, for $\alpha = 0.2$ and two different values of n, the curve in the (p_{00}, p_{11})-plane beyond which the misclassification estimator has lower RMSE than the baseline estimator. Observe that an increase in the size n of the test dataset does not have much impact on the position of the curve. The reason is that the misclassification estimator has a singularity at $p_{00} = p_{11} = 0.5$. The shape of the curve also depends on the value of α. If $\alpha = 0.8$ instead of 0.2, the curves are line-symmetric in the line $p_{00} = p_{11}$. The curve is also line symmetric in $p_{00} = p_{11}$ for $\alpha = 0.5$. The area where the misclassification estimator performs better than the baseline estimator decreases when α gets closer towards 0 or 1. The main reason why this happens is that the variance of the baseline estimator decreases fast when α gets closer towards 0 or 1. Thus, the baseline estimator performs better than the misclassification estimator either if the classifier performs badly in general or performs badly in classifying the largest group.

The final analysis of this paper is to compare the calibration estimator and the misclassification estimator for high values of p_{00} and p_{11}. In Theorem 4 it is proven that, for all possible combinations of α and sufficiently large n, the MSE of the calibration estimator is consistently lower than that of the misclassification estimator.

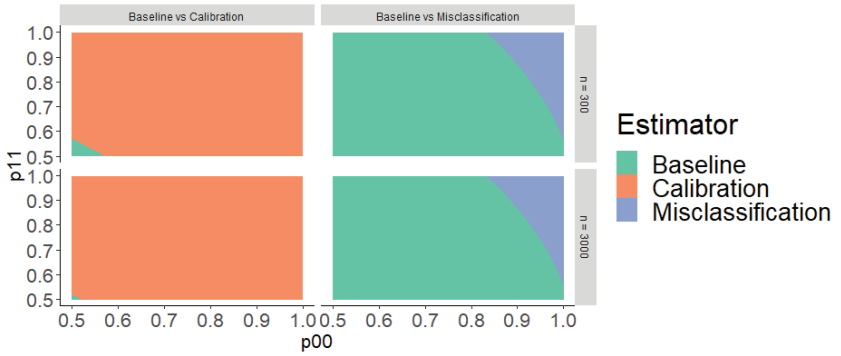


Fig. 4: For each coordinate (p_{00}, p_{11}), the depicted color indicates which estimate has the lowest RMSE, considering only the baseline estimator (green), the calibration estimator (orange) and the misclassification estimator (purple). The top-row panels consider $\alpha = 0.2$ and $n = 300$, while the bottom-row panels consider $\alpha = 0.2$ and $n = 3000$.

Theorem 4. *Let $\widetilde{MSE}[\hat{\alpha}_p]$ and $\widetilde{MSE}[\hat{\alpha}_c]$ denote the approximate mean squared errors, up to terms of order $1/n$, of the misclassification estimator and the calibration estimator, respectively. It holds that:*

$$\widetilde{MSE}[\hat{\alpha}_p] - \widetilde{MSE}[\hat{\alpha}_c] = \frac{\Big[(1-\alpha)p_{00}(1-p_{00}) + \alpha p_{11}(1-p_{11})\Big]^2}{(p_{00}+p_{11}-1)^2\beta(1-\beta)}, \tag{18}$$

in which $\beta \coloneqq (1-\alpha)(1-p_{00}) + \alpha p_{11}$.

Proof. See the Appendix.

Thus, neglecting terms of order $1/n^2$ and higher, the result implies that the calibration estimator has a lower mean squared error than the misclassification estimator, except that both are equal if and only if $p_{00} = p_{11} = 1$. (Note that $0 < \beta < 1$.)

We do remark that the difference in MSE is large in particular for values of p_{00} and p_{11} close to $\frac{1}{2}$. More specifically, it diverges when $p_{00} + p_{11} \to 1$. It is the result of the misclassification estimator having a singularity at $p_{00} + p_{11} = 1$ (see Equation (14)), while the variance of the calibration estimator is bounded. An unpleasant consequence of the singularity at $p_{00} + p_{11} = 1$ is that, for fixed n and α, the probability that $\hat{\alpha}_p$ takes values outside the interval $[0, 1]$ increases as $p_{00} + p_{11} \to 1$; see [14] for a discussion and a possible solution.

4 Conclusion and Discussion

In this paper, we have studied the effect of classification errors on five estimators of the base rate parameter α that are obtained from machine learning algorithms.

In general, a straightforward classify-and-count estimator will lead to biased estimates and some form of bias correction should be considered. As reducing bias might increase variance, we evaluated the (root) mean squared error (MSE) of the five estimators, both theoretically as well as numerically.

From our results we may draw the following main (three-part) conclusion regarding which estimator for α has lowest mean squared error. First, when dealing with small test datasets and rather poor algorithms, that is p_{00} and p_{11} both close to 0.5, the baseline estimator $\hat{\alpha}_a$ has the lowest MSE. Second, when dealing with algorithms for which the classification probabilities p_{00} and p_{11} are in a small neighbourhood around the line $(p_{11} - 1)\alpha + (1 - p_{00})(1 - \alpha) = 0$ in the (p_{00}, p_{11})-plane, the classify-and-count estimator and the subtracted-bias estimator will have the lowest MSE. As the size of the test dataset increases, the size of that neighbourhood decreases. Third, in any other situation, the calibration estimator will have the lowest MSE. In practice, the test dataset will have to be used to determine which of the three scenarios applies to the data and the algorithm at hand. It is an additional estimation problem that we have not discussed in this paper.

We would like to close the paper by pointing out three interesting directions for future research. First, the results could be generalized to multi-class classification problems. The theoretical derivations of the bias and variance are more complicated and involve matrix-vector notation, but the proof strategy is similar. However, it is more challenging to compare the MSE of the five estimators visually in the multi-class case.

Second, the assumptions that we have made could be relaxed. In particular, a trained and implemented machine learning model is, in practice, often used over a longer period of time. A shift in the base rate parameter α, also known as prior probability shift [15], is then inevitable. Consequently, we may no longer assume that the conditional distribution of the class label given the features in the test dataset is similar to that in the population. It implies that the calibration estimator is no longer unbiased, which might have a significant effect on our main conclusion.

Third and finally, a combination of estimators might have a substantially lower MSE than that of the individual estimators separately. Therefore, it might be interesting to study different methods of model averaging applied to the problem of misclassification bias. It could be fruitful especially when the assumptions that we have made are relaxed.

References

1. Buonaccorsi, J.P.: Measurement Error: Models, Methods, and Applications. Chapman & Hall/CRC, Boca Raton, FL (2010)
2. Burger, J., Delden, A.v., Scholtus, S.: Sensitivity of Mixed-Source Statistics to Classification Errors. Journal of Official Statistics **31**(3), 489–506 (2015)
3. Curier, R., De Jong, T., Strauch, K., Cramer, K., Rosenski, N., Schartner, C., Debusschere, M., Ziemons, H., Iren, D., Bromuri, S.: Monitoring spatial sustainable

development: Semi-automated analysis of satellite and aerial images for energy transition and sustainability indicators. arXiv preprint arXiv:1810.04881 (2018)
4. Czaplewski, R.L.: Misclassification bias in areal estimates. Photogrammetric Engineering and Remote Sensing. 58(2): 189-192 **58**(2), 189–192 (1992)
5. Czaplewski, R.L., Catts, G.P.: Calibration of remotely sensed proportion or area estimates for misclassification error. Remote Sensing of Environment **39**(1), 29–43 (1992)
6. González, P., Castaño, A., Chawla, N.V., Coz, J.J.D.: A Review on Quantification Learning. ACM Computing Surveys **50**(5), 74:1–74:40 (2017). https://doi.org/10.1145/3117807
7. Grassia, A., Sundberg, R.: Statistical Precision in the Calibration and Use of Sorting Machines and Other Classifiers. Technometrics **24**(2), 117–121 (1982)
8. Greenland, S.: Sensitivity Analysis and Bias Analysis. In: Ahrens, W., Pigeot, I. (eds.) Handbook of Epidemiology. Springer, New York, NY (2014)
9. Hopkins, D.J., King, G.: A Method of Automated Nonparametric Content Analysis for Social Science. American Journal of Political Science **54**(1), 229–247 (2010)
10. Knottnerus, P.: Sample survey theory: some Pythagorean perspectives. Springer Science & Business Media (2003)
11. Kuha, J., Skinner, C.J.: Categorical data analysis and misclassification. In: Lyberg, L., Biemer, P., Collins, M., de Leeuw, E., Dippo, C., Schwarz, N., Trewin, D. (eds.) Survey Measurement and Process Quality, pp. 633–670. Wiley (Mar 1997)
12. Löw, F., Knöfel, P., Conrad, C.: Analysis of uncertainty in multi-temporal object-based classification. ISPRS Journal of Photogrammetry and Remote Sensing **105**, 91–106 (2015)
13. Meertens, Q.A., Diks, C.G.H., Herik, H.J.v.d., Takes, F.W.: A data-driven supply-side approach for estimating cross-border Internet purchases within the European Union. Journal of the Royal Statistical Society: Series A (Statistics in Society) **183**(1), 61–90 (2020). https://doi.org/10.1111/rssa.12487
14. Meertens, Q., Diks, C., van den Herik, H., Takes, F.: A Bayesian Approach for Accurate Classification-Based Aggregates. Society for Industrial and Applied Mathematics, Philadelphia, PA (2019)
15. Moreno-Torres, J.G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N.V., Herrera, F.: A unifying view on dataset shift in classification. Pattern Recognition **45**(1), 521–530 (2012). https://doi.org/10.1016/j.patcog.2011.06.019
16. O'Connor, B., Balasubramanyan, R., Routledge, B., Smith, N.: From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. In: Proceedings of the International AAAI Conference on Weblogs and Social Media. Washington, DC (2010)
17. Scholtus, S., van Delden, A.: On the accuracy of estimators based on a binary classifier (Feb 2020), Discussion Paper, Statistics Netherlands, The Hague
18. Schwartz, J.E.: The Neglected Problem of Measurement Error in Categorical Data. Sociological Methods & Research **13**(4), 435–466 (1985). https://doi.org/10.1177/0049124185013004001
19. Strichartz, R.S.: The Way of Analysis. Jones & Bartlett Learning (2000)
20. Van Delden, A., Scholtus, S., Burger, J.: Accuracy of mixed-source statistics as affected by classification errors. Journal of Official Statistics **32**(3), 619–642 (2016)
21. Wiedemann, G.: Proportional Classification Revisited: Automatic Content Analysis of Political Manifestos Using Active Learning. Social Science Computer Review **37**(2), 135–159 (2019)

Appendix

This appendix contains the proofs of the theorems presented in the paper entitled "Comparing Correction Methods for Misclassification Bias". Recall that we have assumed a population of size N in which a fraction $\alpha \coloneqq N_{1+}/N$ belongs to the class of interest, referred to as the class labelled as 1. We assume that a binary classification algorithm has been trained that correctly classifies a data point that belongs to class $i \in \{0, 1\}$ with probability $p_{ii} > 0.5$, independently across all data points. In addition, we assume that a test set of size $n \ll N$ is available and that it can be considered a simple random sample from the population. The classification probabilities p_{00} and p_{11} are estimated on that test set as described in Section 2. Finally, we assume that the classify-and-count estimator $\hat{\alpha}^*$ is distributed independently of $\hat{p}_{00}$ and $\hat{p}_{11}$, which is reasonable (at least as an approximation) when $n \ll N$.

It may be noted that the estimated probabilities $\hat{p}_{11}$ and $\hat{p}_{00}$ defined in Section 2 cannot be computed if $n_{1+} = 0$ or $n_{0+} = 0$. Similarly, the calibration probabilities c_{11} and c_{00} cannot be estimated if $n_{+1} = 0$ or $n_{+0} = 0$. We assume here that these events occur with negligible probability. This will be true when n is sufficiently large so that $n\alpha \gg 1$ and $n(1-\alpha) \gg 1$.

Preliminaries

Many of the proofs presented in this appendix rely on the following two mathematical results. First, we will use univariate and bivariate Taylor series to approximate the expectation of non-linear functions of random variables. That is, to estimate $E[f(X)]$ and $E[g(X, Y)]$ for sufficiently differentiable functions f and g, we will insert the Taylor series for f and g at $x_0 = E[X]$ and $y_0 = E[Y]$ up to terms of order 2 and utilize the linearity of the expectation. Second, we will use the following conditional variance decomposition for the variance of a random variable X:

$$V(X) = E[V(X \mid Y)] + V(E[X \mid Y]). \tag{19}$$

The conditional variance decomposition follows from the tower property of conditional expectations [10]. Before we prove the theorems presented in the paper, we begin by proving the following lemma.

Lemma 1. *The variance of the estimator $\hat{p}_{11}$ for p_{11} estimated on the test set is given by*

$$V(\hat{p}_{11}) = \frac{p_{11}(1-p_{11})}{n\alpha}\left[1 + \frac{1-\alpha}{n\alpha}\right] + O\left(\frac{1}{n^3}\right). \tag{20}$$

Similarly, the variance of $\hat{p}_{00}$ is given by

$$V(\hat{p}_{00}) = \frac{p_{00}(1-p_{00})}{n(1-\alpha)}\left[1 + \frac{\alpha}{n(1-\alpha)}\right] + O\left(\frac{1}{n^3}\right). \tag{21}$$

Moreover, $\hat{p}_{11}$ and $\hat{p}_{00}$ are uncorrelated: $C(\hat{p}_{11}, \hat{p}_{00}) = 0$.

Proof (of Lemma 1). We approximate the variance of $\hat{p}_{00}$ using the conditional variance decomposition and a second-order Taylor series, as follows:

$$\begin{aligned}
V(\hat{p}_{00}) &= V\left(\frac{n_{00}}{n_{0+}}\right) \\
&= E_{n_{0+}}\left[V\left(\frac{n_{00}}{n_{0+}} \mid n_{0+}\right)\right] + V_{n_{0+}}\left[E\left(\frac{n_{00}}{n_{0+}} \mid n_{0+}\right)\right] \\
&= E_{n_{0+}}\left[\frac{1}{n_{0+}^2}V(n_{00} \mid n_{0+})\right] + V_{n_{0+}}\left[\frac{1}{n_{0+}}E(n_{00} \mid n_{0+})\right] \\
&= E_{n_{0+}}\left[\frac{n_{0+}p_{00}(1-p_{00})}{n_{0+}^2}\right] + V_{n_{0+}}\left[\frac{n_{0+}p_{00}}{n_{0+}}\right] \\
&= E_{n_{0+}}\left[\frac{1}{n_{0+}}\right]p_{00}(1-p_{00}) \\
&= \left[\frac{1}{E[n_{0+}]} + \frac{1}{2}\frac{2}{E[n_{0+}]^3} \times V[n_{0+}]\right]p_{00}(1-p_{00}) + O\left(\frac{1}{n^3}\right) \\
&= \frac{p_{00}(1-p_{00})}{E[n_{0+}]}\left[1 + \frac{V[n_{0+}]}{E[n_{0+}]^2}\right] + O\left(\frac{1}{n^3}\right) \\
&= \frac{p_{00}(1-p_{00})}{n(1-\alpha)}\left[1 + \frac{\alpha}{n(1-\alpha)}\right] + O\left(\frac{1}{n^3}\right).
\end{aligned}$$

The variance of $\hat{p}_{11}$ is approximated in the exact same way.

Finally, to evaluate $C(\hat{p}_{11}, \hat{p}_{00})$ we use the analogue of (19) for covariances:

$$\begin{aligned}
C(\hat{p}_{11}, \hat{p}_{00}) &= C\left(\frac{n_{11}}{n_{1+}}, \frac{n_{00}}{n_{0+}}\right) \\
&= E_{n_{1+},n_{0+}}\left[C\left(\frac{n_{11}}{n_{1+}}, \frac{n_{00}}{n_{0+}} \mid n_{1+}, n_{0+}\right)\right] \\
&\qquad + C_{n_{1+},n_{0+}}\left[E\left(\frac{n_{11}}{n_{1+}} \mid n_{1+}, n_{0+}\right), E\left(\frac{n_{00}}{n_{0+}} \mid n_{1+}, n_{0+}\right)\right] \\
&= E_{n_{1+},n_{0+}}\left[\frac{1}{n_{1+}n_{0+}}C(n_{11}, n_{00} \mid n_{1+}, n_{0+})\right] \\
&\qquad + C_{n_{1+},n_{0+}}\left[\frac{1}{n_{1+}}E(n_{11} \mid n_{1+}), \frac{1}{n_{0+}}E(n_{00} \mid n_{0+})\right].
\end{aligned}$$

The second term is zero as before. The first term also vanishes because, conditional on the row totals n_{1+} and n_{0+}, the counts n_{11} and n_{00} follow independent binomial distributions, so $C(n_{11}, n_{00} \mid n_{1+}, n_{0+}) = 0$.

Note: in the remainder of this appendix, we will not add explicit subscripts to expectations and variances when their meaning is unambiguous.

Subtracted-bias estimator

We will now prove the bias and variance approximations for the subtracted-bias estimator $\hat{\alpha}_b$ that was defined in Equation (9).

Proof (of Theorem 1). The bias of $\hat{\alpha}_b$ is given by

$$\begin{aligned} B(\hat{\alpha}_b) &= E\left[\hat{\alpha}^\star - \hat{B}[\hat{\alpha}^\star]\right] - \alpha \\ &= E[\hat{\alpha}^\star - \alpha] - E\left[\hat{B}[\hat{\alpha}^\star]\right] \\ &= B[\hat{\alpha}^\star] - E\left[\hat{B}[\hat{\alpha}^\star]\right] \\ &= [\alpha(p_{00} + p_{11} - 2) + (1 - p_{00})] - E\left[\hat{\alpha}^\star(\hat{p}_{00} + \hat{p}_{11} - 2) + (1 - \hat{p}_{00})\right]. \end{aligned}$$

Because $\hat{\alpha}^*$ and $(\hat{p}_{00} + \hat{p}_{11} - 2)$ are assumed to be independent, the expectation of their product equals the product of their expectations:

$$\begin{aligned} B(\hat{\alpha}_b) &= \alpha(p_{00} + p_{11} - 2) + (1 - p_{00}) - E[\hat{\alpha}^\star](p_{00} + p_{11} - 2) - (1 - p_{00}) \\ &= (\alpha - E[\hat{\alpha}^\star])(p_{00} + p_{11} - 2) \\ &= B[\hat{\alpha}^\star](2 - p_{00} - p_{11}) \\ &= (1 - p_{00})(2 - p_{00} - p_{11}) - \alpha(p_{00} + p_{11} - 2)^2. \end{aligned}$$

This proves the formula for the bias of $\hat{\alpha}_b$ as estimator for α. To approximate the variance of $\hat{\alpha}_b$, we apply the conditional variance decomposition (19) conditional on $\hat{\alpha}^*$ and look at the two resulting terms separately. First, consider the expectation of the conditional variance:

$$\begin{aligned} E\left[V(\hat{\alpha}_b \mid \hat{\alpha}^*)\right] &= E\left[V(\hat{\alpha}^*(3 - \hat{p}_{00} - \hat{p}_{11}) - (1 - \hat{p}_{00}) \mid \hat{\alpha}^*)\right] \\ &= E\big[V(\hat{\alpha}^*(3 - \hat{p}_{00} - \hat{p}_{11}) \mid \hat{\alpha}^*) + V(1 - \hat{p}_{00} \mid \hat{\alpha}^*) \\ &\qquad - 2C(\hat{\alpha}^*(3 - \hat{p}_{00} - \hat{p}_{11}), 1 - \hat{p}_{00} \mid \hat{\alpha}^*)\big] \\ &= E\big[(\hat{\alpha}^*)^2 V(3 - \hat{p}_{00} - \hat{p}_{11} \mid \hat{\alpha}^*) + V(1 - \hat{p}_{00} \mid \hat{\alpha}^*) \\ &\qquad - 2\hat{\alpha}^* C(3 - \hat{p}_{00} - \hat{p}_{11}, 1 - \hat{p}_{00} \mid \hat{\alpha}^*)\big] \\ &= E\big[(\hat{\alpha}^*)^2\left[V(\hat{p}_{00}) + V(\hat{p}_{11})\right] + V(\hat{p}_{00}) - 2\hat{\alpha}^* V(\hat{p}_{00})\big] \\ &= E\left[(\hat{\alpha}^*)^2\right]\left[V(\hat{p}_{00}) + V(\hat{p}_{11})\right] + V(\hat{p}_{00}) - 2E\left[\hat{\alpha}^*\right] V(\hat{p}_{00}). \end{aligned}$$

In the penultimate line, we used that $C(\hat{p}_{11}, \hat{p}_{00}) = 0$. The second moment $E\left[(\hat{\alpha}^*)^2\right]$ can be written as $E\left[\hat{\alpha}^*\right]^2 + V(\hat{\alpha}^*)$. Because $V(\hat{\alpha}^*)$ is of order $1/N$, it can be neglected compared to $E\left[\hat{\alpha}^*\right]^2$, which is of order 1. In particular, we find that the expectation of the conditional variance equals:

$$\begin{aligned} E\left[V(\hat{\alpha}_b \mid \hat{\alpha}^*)\right] &= E\left[(\hat{\alpha}^*)\right]^2\left[V(\hat{p}_{00}) + V(\hat{p}_{11})\right] + V(\hat{p}_{00}) - 2E\left[\hat{\alpha}^*\right] V(\hat{p}_{00}) + O\left(\frac{1}{N}\right) \\ &= V(\hat{p}_{00})\left[E\left[\hat{\alpha}^*\right] - 1\right]^2 + V(\hat{p}_{11}) E\left[\hat{\alpha}^*\right]^2 + O\left(\frac{1}{N}\right). \end{aligned}$$

Next, the variance of the conditional expectation can be seen to be equal the following:

$$\begin{aligned} V\left[E(\hat{\alpha}_b \mid \hat{\alpha}^*)\right] &= V\left[E(\hat{\alpha}^*(3 - \hat{p}_{00} - \hat{p}_{11}) - (1 - \hat{p}_{00}) \mid \hat{\alpha}^*)\right] \\ &= V\left[\hat{\alpha}^* E(3 - \hat{p}_{00} - \hat{p}_{11} \mid \hat{\alpha}^*) - E(1 - \hat{p}_{00} \mid \hat{\alpha}^*)\right] \\ &= V(\hat{\alpha}^*)(3 - p_{00} - p_{11})^2. \end{aligned}$$

Because $V(\hat{\alpha}^*)$ is of order $1/N$, it can be neglected in the final formula. Furthermore, the variances of $\hat{p}_{00}$ and $\hat{p}_{11}$ can be written out using the result from Lemma 1:

$$\begin{aligned} V(\hat{\alpha}_b) = & \frac{\left[\alpha(p_{00}+p_{11}-1)-p_{00}\right]^2 p_{00}(1-p_{00})}{n(1-\alpha)}\left[1+\frac{\alpha}{n(1-\alpha)}\right] \\ & + \frac{\left[\alpha(p_{00}+p_{11}-1)+(1-p_{00})\right]^2 p_{11}(1-p_{11})}{n\alpha}\left[1+\frac{1-\alpha}{n\alpha}\right] \\ & + O\left(\max\left[\frac{1}{n^3},\frac{1}{N}\right]\right). \end{aligned}$$

This concludes the proof of Theorem 1.

Misclassification estimator

We will now prove the bias and variance approximations for the misclassification estimator $\hat{\alpha}_p$ as defined in Equation (12).

Proof (of Theorem 2). Under the assumption that $\hat{\alpha}^*$ is distributed independently of $(\hat{p}_{00}, \hat{p}_{11})$, it holds that

$$\begin{aligned} E(\hat{\alpha}_p) &= E\left(\frac{\hat{p}_{00}-1}{\hat{p}_{00}+\hat{p}_{11}-1}\right) + E\left[E\left(\frac{\hat{\alpha}^*}{\hat{p}_{00}+\hat{p}_{11}-1}\,\middle|\,\hat{\alpha}^*\right)\right] \\ &= E\left(\frac{\hat{p}_{00}-1}{\hat{p}_{00}+\hat{p}_{11}-1}\right) + E(\hat{\alpha}^*)E\left(\frac{1}{\hat{p}_{00}+\hat{p}_{11}-1}\right). \end{aligned} \tag{22}$$

$E(\hat{\alpha}^*)$ is known from (4). To evaluate the other two expectations, we use a second-order Taylor series approximation. The first- and second-order partial derivatives of $f(x,y) = 1/(x+y-1)$ and $g(x,y) = (x-1)/(x+y-1) = 1-[y/(x+y-1)]$ are given by:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial y} = \frac{-1}{(x+y-1)^2}, \tag{23}$$

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial^2 f}{\partial y^2} = \frac{2}{(x+y-1)^3},$$

$$\frac{\partial g}{\partial x} = \frac{y}{(x+y-1)^2}, \tag{24}$$

$$\frac{\partial g}{\partial y} = \frac{-(x-1)}{(x+y-1)^2}, \tag{25}$$

$$\frac{\partial^2 g}{\partial x^2} = \frac{-2y}{(x+y-1)^3},$$

$$\frac{\partial^2 g}{\partial y^2} = \frac{2(x-1)}{(x+y-1)^3}.$$

Now also using that $C(\hat{p}_{11}, \hat{p}_{00}) = 0$, we obtain for the first expectation:

$$E\left(\frac{1}{\hat{p}_{00}+\hat{p}_{11}-1}\right) = \frac{1}{p_{00}+p_{11}-1} + \frac{V(\hat{p}_{00})+V(\hat{p}_{11})}{(p_{00}+p_{11}-1)^3} + O(n^{-2})$$
$$= \frac{1}{p_{00}+p_{11}-1}\left[1 + \frac{\frac{p_{00}(1-p_{00})}{n(1-\alpha)} + \frac{p_{11}(1-p_{11})}{n\alpha}}{(p_{00}+p_{11}-1)^2}\right] + O(n^{-2}). \tag{26}$$

Here, we have included only the first term of the approximations to $V(\hat{p}_{00})$ and $V(\hat{p}_{11})$ from Lemma 1, since this suffices to approximate the bias up to terms of order $O(1/n)$. Similarly, for the second expectation we obtain:

$$E\left(\frac{\hat{p}_{00}-1}{\hat{p}_{00}+\hat{p}_{11}-1}\right) = \frac{p_{00}-1}{p_{00}+p_{11}-1} + \frac{(p_{00}-1)V(\hat{p}_{11}) - p_{11}V(\hat{p}_{00})}{(p_{00}+p_{11}-1)^3} + O(n^{-2})$$
$$= \frac{p_{00}-1}{p_{00}+p_{11}-1}\left[1 + p_{11}\frac{\frac{1-p_{11}}{n\alpha} + \frac{p_{00}}{n(1-\alpha)}}{(p_{00}+p_{11}-1)^2}\right] + O(n^{-2}). \tag{27}$$

Using (22), (4), (26), and (27), we conclude that:

$$E(\hat{\alpha}_p) = \frac{\alpha(p_{00}+p_{11}-1)-(p_{00}-1)}{p_{00}+p_{11}-1}\left[1 + \frac{\frac{p_{00}(1-p_{00})}{n(1-\alpha)} + \frac{p_{11}(1-p_{11})}{n\alpha}}{(p_{00}+p_{11}-1)^2}\right]$$
$$+ \frac{p_{00}-1}{p_{00}+p_{11}-1}\left[1 + p_{11}\frac{\frac{1-p_{11}}{n\alpha} + \frac{p_{00}}{n(1-\alpha)}}{(p_{00}+p_{11}-1)^2}\right] + O\left(\frac{1}{n^2}\right).$$

From this, it follows that an approximation to the bias of $\hat{\alpha}_p$ that is correct up to terms of order $O(1/n)$ is given by:

$$B(\hat{\alpha}_p) = \frac{\alpha(p_{00}+p_{11}-1)-(p_{00}-1)}{n(p_{00}+p_{11}-1)^3}\left[\frac{p_{00}(1-p_{00})}{1-\alpha} + \frac{p_{11}(1-p_{11})}{\alpha}\right]$$
$$+ \frac{(p_{00}-1)p_{11}}{n(p_{00}+p_{11}-1)^3}\left[\frac{1-p_{11}}{\alpha} + \frac{p_{00}}{1-\alpha}\right] + O\left(\frac{1}{n^2}\right).$$

By expanding the products in this expression and combining similar terms, the expression can be simplified to:

$$B(\hat{\alpha}_p) = \frac{p_{11}(1-p_{11}) - p_{00}(1-p_{00})}{n(p_{00}+p_{11}-1)^2} + O\left(\frac{1}{n^2}\right).$$

Finally, using the identity $p_{11}(1-p_{11}) - p_{00}(1-p_{00}) = (p_{00}+p_{11}-1)(p_{00}-p_{11})$, we obtain the required result for $B(\hat{\alpha}_p)$.

To approximate the variance of $\hat{\alpha}_p$, we apply the conditional variance decomposition conditional on $\hat{\alpha}^*$ and look at the two resulting terms separately. First,

consider the variance of the conditional expectation:

$$\begin{aligned} V\left[E(\hat{\alpha}_p \mid \hat{\alpha}^*)\right] &= V\left[E\left(\hat{\alpha}^* \frac{1}{\hat{p}_{00}+\hat{p}_{11}-1} + \frac{\hat{p}_{00}-1}{\hat{p}_{00}+\hat{p}_{11}-1} \mid \hat{\alpha}^*\right)\right] \\ &= V\left[\hat{\alpha}^* \frac{1}{p_{00}+p_{11}-1}\right] \\ &= \frac{1}{(p_{00}+p_{11}-1)^2} V\left[\hat{\alpha}^*\right] = O\left(\frac{1}{N}\right), \end{aligned} \tag{28}$$

where in the last line we used (6). Note: the factor $1/(p_{00}+p_{11}-1)^2$ can become arbitrarily large in the limit $p_{00}+p_{11} \to 1$. It will be seen below that this same factor also occurs in the lower-order terms of $V(\hat{\alpha}_p)$; hence, the relative contribution of (28) remains negligible even in the limit $p_{00}+p_{11} \to 1$.

Next, we compute the expectation of the conditional variance.

$$\begin{aligned} E\left[V(\hat{\alpha}_p \mid \hat{\alpha}^*)\right] &= E\left[V\left(\hat{\alpha}^* \frac{1}{\hat{p}_{00}+\hat{p}_{11}-1} + \frac{\hat{p}_{00}-1}{\hat{p}_{00}+\hat{p}_{11}-1} \mid \hat{\alpha}^\star\right)\right] \\ &= E\Bigg[V\left(\hat{\alpha}^* \frac{1}{\hat{p}_{00}+\hat{p}_{11}-1} \mid \alpha^\star\right) + V\left(\frac{\hat{p}_{00}-1}{\hat{p}_{00}+\hat{p}_{11}-1} \mid \hat{\alpha}^\star\right) \\ &\qquad + 2C\left(\hat{\alpha}^* \frac{1}{\hat{p}_{00}+\hat{p}_{11}-1}, \frac{\hat{p}_{00}-1}{\hat{p}_{00}+\hat{p}_{11}-1} \mid \hat{\alpha}^\star\right)\Bigg] \\ &= E\left[(\hat{\alpha}^*)^2\right] V\left[\frac{1}{\hat{p}_{00}+\hat{p}_{11}-1}\right] + V\left[\frac{\hat{p}_{00}-1}{\hat{p}_{00}+\hat{p}_{11}-1}\right] \\ &\qquad + 2E\left[\hat{\alpha}^\star\right] C\left[\frac{1}{\hat{p}_{00}+\hat{p}_{11}-1}, \frac{\hat{p}_{00}-1}{\hat{p}_{00}+\hat{p}_{11}-1}\right] \\ &= E\left[\hat{\alpha}^\star\right]^2 \left[1+O\left(\frac{1}{N}\right)\right] V\left[\frac{1}{\hat{p}_{00}+\hat{p}_{11}-1}\right] + V\left[\frac{\hat{p}_{00}-1}{\hat{p}_{00}+\hat{p}_{11}-1}\right] \\ &\qquad + 2E\left[\hat{\alpha}^\star\right] C\left[\frac{1}{\hat{p}_{00}+\hat{p}_{11}-1}, \frac{\hat{p}_{00}-1}{\hat{p}_{00}+\hat{p}_{11}-1}\right]. \end{aligned} \tag{29}$$

To approximate the variance and covariance terms, we use a first-order Taylor series. Using the partial derivatives in (23), (24) and (25), we obtain:

$$\begin{aligned} V\left[\frac{1}{\hat{p}_{00}+\hat{p}_{11}-1}\right] &= \frac{V(\hat{p}_{00})+V(\hat{p}_{11})}{(p_{00}+p_{11}-1)^4} + O(n^{-2}) \\ V\left[\frac{\hat{p}_{00}-1}{\hat{p}_{00}+\hat{p}_{11}-1}\right] &= \frac{V(\hat{p}_{00})(p_{11})^2}{(p_{00}+p_{11}-1)^4} + \frac{V(\hat{p}_{11})(1-p_{00})^2}{(p_{00}+p_{11}-1)^4} + O(n^{-2}) \\ C\left[\frac{1}{\hat{p}_{00}+\hat{p}_{11}-1}, \frac{\hat{p}_{00}-1}{\hat{p}_{00}+\hat{p}_{11}-1}\right] &= \frac{V(\hat{p}_{00})(-p_{11})}{(p_{00}+p_{11}-1)^4} + \frac{V(\hat{p}_{11})(p_{00}-1)}{(p_{00}+p_{11}-1)^4} + O(n^{-2}). \end{aligned}$$

Substituting these terms into Formula (29) and accounting for Formula (28) yields:

$$\begin{aligned}
V(\hat{\alpha}_p) &= \frac{V(\hat{p}_{00})\left[E\left[\hat{\alpha}^\star\right]^2 - 2p_{11}E\left[\hat{\alpha}^\star\right] + p_{11}^2\right]}{(p_{00}+p_{11}-1)^4} \\
&\quad + \frac{V(\hat{p}_{11})\left[E\left[\hat{\alpha}^\star\right]^2 - 2(1-p_{00})E\left[\hat{\alpha}^\star\right] + (1-p_{00})^2\right]}{(p_{00}+p_{11}-1)^4} + O\left(\max\left[\frac{1}{n^2}, \frac{1}{N}\right]\right) \\
&= \frac{V(\hat{p}_{00})\left[E\left[\hat{\alpha}^\star\right] - p_{11}\right]^2}{(p_{00}+p_{11}-1)^4} + \frac{V(\hat{p}_{11})\left[E\left[\hat{\alpha}^\star\right] - (1-p_{00})\right]^2}{(p_{00}+p_{11}-1)^4} + O\left(\max\left[\frac{1}{n^2}, \frac{1}{N}\right]\right) \\
&= \frac{V(\hat{p}_{00})(1-\alpha)^2}{(p_{00}+p_{11}-1)^2} + \frac{V(\hat{p}_{11})\alpha^2}{(p_{00}+p_{11}-1)^2} + O\left(\max\left[\frac{1}{n^2}, \frac{1}{N}\right]\right).
\end{aligned}$$

Finally, inserting the expressions for $V(\hat{p}_{00})$ and $V(\hat{p}_{11})$ from Lemma 1 yields:

$$\begin{aligned}
V(\hat{\alpha}_p) &= \frac{\frac{p_{00}(1-p_{00})}{n(1-\alpha)}\left[1 + \frac{\alpha}{n(1-\alpha)}\right](1-\alpha)^2}{(p_{00}+p_{11}-1)^2} + \frac{\frac{p_{11}(1-p_{11})}{n\alpha}\left[1 + \frac{1-\alpha}{n\alpha}\right]\alpha^2}{(p_{00}+p_{11}-1)^2} \\
&\quad + O\left(\max\left[\frac{1}{n^2}, \frac{1}{N}\right]\right),
\end{aligned}$$

from which expression (14) follows. This concludes the proof of Theorem 2.

Calibration estimator

We will now prove the bias and variance approximations for the calibration estimator $\hat{\alpha}_c$ that was defined in Equation (15).

Proof (of Theorem 3). To compute the expected value of $\hat{\alpha}_c$, we first compute its expectation conditional on the 4-vector $\boldsymbol{N} = (N_{00}, N_{01}, N_{10}, N_{11})$:

$$\begin{aligned}
E(\hat{\alpha}_c \mid \boldsymbol{N}) &= E\left[\hat{\alpha}^* \frac{n_{11}}{n_{+1}} + (1-\hat{\alpha}^*)\frac{n_{10}}{n_{+0}} \mid \boldsymbol{N}\right] \\
&= \hat{\alpha}^* E\left[\frac{n_{11}}{n_{+1}} \mid \boldsymbol{N}\right] + (1-\hat{\alpha}^*)E\left[\frac{n_{10}}{n_{+0}} \mid \boldsymbol{N}\right] \\
&= \hat{\alpha}^* E\left[E\left(\frac{n_{11}}{n_{+1}} \mid \boldsymbol{N}, n_{+1}\right) \mid \boldsymbol{N}\right] \\
&\qquad + (1-\hat{\alpha}^*)E\left[E\left(\frac{n_{10}}{n_{+0}} \mid \boldsymbol{N}, n_{+0}\right) \mid \boldsymbol{N}\right] \\
&= \frac{N_{+1}}{N} E\left[\frac{1}{n_{+1}} n_{+1} \frac{N_{11}}{N_{+1}} \mid \boldsymbol{N}\right] + \frac{N_{+0}}{N} E\left[\frac{1}{n_{+0}} n_{+0} \frac{N_{10}}{N_{+0}} \mid \boldsymbol{N}\right] \\
&= \frac{N_{11}}{N} + \frac{N_{10}}{N} \\
&= \frac{N_{1+}}{N} = \alpha. \qquad (30)
\end{aligned}$$

By the tower property of conditional expectations, it follows that $E[\hat{\alpha}_c] = E\left[E(\hat{\alpha}_c \mid \boldsymbol{N})\right] = \alpha$. This proves that $\hat{\alpha}_c$ is an unbiased estimator for α.

To compute the variance of $\hat{\alpha}_c$, we use the conditional variance decomposition, again conditioning on the 4-vector $\boldsymbol{N}$. We remark that N_{0+} and N_{1+} are deterministic values, but that N_{+0} and N_{+1} are random variables. As shown above in Equation (30), the conditional expectation is deterministic, hence it has no variance: $V(E[\hat{\alpha}_c \mid \boldsymbol{N}]) = 0$. The conditional variance decomposition then simplifies to the following:

$$V(\hat{\alpha}_c) = E\left[V(\hat{\alpha}_c \mid \boldsymbol{N})\right]. \tag{31}$$

The conditional variance $V(\hat{\alpha}_c \mid \boldsymbol{N})$ can be written as follows:

$$\begin{aligned} V[\hat{\alpha}_c \mid \boldsymbol{N}] &= V\left[\hat{\alpha}^* \frac{n_{11}}{n_{+1}} + (1-\hat{\alpha}^*)\frac{n_{10}}{n_{+0}} \mid \boldsymbol{N}\right] \\ &= (\hat{\alpha}^*)^2 V\left[\frac{n_{11}}{n_{+1}} \mid \boldsymbol{N}\right] + (1-\hat{\alpha}^*)^2 V\left[\frac{n_{10}}{n_{+0}} \mid \boldsymbol{N}\right] \\ &\quad + 2\hat{\alpha}^*(1-\hat{\alpha}^*) C\left[\frac{n_{11}}{n_{+1}}, \frac{n_{10}}{n_{+0}} \mid \boldsymbol{N}\right]. \end{aligned} \tag{32}$$

We will consider these terms separately. First, the variance of n_{11}/n_{+1} can be computed by applying an additional conditional variance decomposition:

$$V\left[\frac{n_{11}}{n_{+1}} \mid \boldsymbol{N}\right] = V\left[E\left(\frac{n_{11}}{n_{+1}} \mid \boldsymbol{N}, n_{+1}\right) \mid \boldsymbol{N}\right] + E\left[V\left(\frac{n_{11}}{n_{+1}} \mid \boldsymbol{N}, n_{+1}\right) \mid \boldsymbol{N}\right].$$

The first term is zero, which can be shown as follows:

$$\begin{aligned} V\left[E\left(\frac{n_{11}}{n_{+1}} \mid \boldsymbol{N}, n_{+1}\right)\right] &= V\left[\frac{1}{n_{+1}} E(n_{11} \mid \boldsymbol{N}, n_{+1}) \mid \boldsymbol{N}\right] \\ &= V\left[\frac{1}{n_{+1}} n_{+1} \frac{N_{11}}{N_{+1}} \mid \boldsymbol{N}\right] \\ &= V\left[\frac{N_{11}}{N_{+1}} \mid \boldsymbol{N}\right] = 0. \end{aligned}$$

For the second term, we find under the assumption that $n \ll N$:

$$\begin{aligned} E\left[V\left(\frac{n_{11}}{n_{+1}} \mid \boldsymbol{N}, n_{+1}\right) \mid \boldsymbol{N}\right] &= E\left[\frac{1}{n_{+1}^2} V(n_{11} \mid \boldsymbol{N}, n_{+1}) \mid \boldsymbol{N}\right] \\ &= E\left[\frac{1}{n_{+1}^2} n_{+1} \frac{N_{11}}{N_{+1}}\left(1 - \frac{N_{11}}{N_{+1}}\right) \mid \boldsymbol{N}\right] \\ &= E\left[\frac{1}{n_{+1}} \mid \boldsymbol{N}\right] \frac{N_{11} N_{01}}{N_{+1}^2}. \end{aligned}$$

The expectation of $\frac{1}{n_{+1}}$ can be approximated with a second-order Taylor series:

$$\begin{aligned} V\left[\frac{n_{11}}{n_{+1}} \mid \boldsymbol{N}\right] &= \left[\frac{1}{E[n_{+1} \mid \boldsymbol{N}]} + \frac{1}{2}\frac{2}{E[n_{+1} \mid \boldsymbol{N}]^3} V\left[n_{+1} \mid \boldsymbol{N}\right]\right] \frac{N_{11}N_{01}}{N_{+1}^2} + O(n^{-3}) \\ &= \frac{1}{E[n_{+1} \mid \boldsymbol{N}]}\left[1 + \frac{V\left[n_{+1} \mid \boldsymbol{N}\right]}{E[n_{+1} \mid \boldsymbol{N}]^2}\right] \frac{N_{11}N_{01}}{N_{+1}^2} + O(n^{-3}) \\ &= \frac{1}{n\hat{\alpha}^*}\left[1 + \frac{1-\hat{\alpha}^*}{n\hat{\alpha}^*}\right] \frac{N_{11}N_{01}}{N_{+1}^2} + O(n^{-3}). \end{aligned} \tag{33}$$

The variance of n_{10}/n_{+0} can be approximated in the same way, which yields the following expression:

$$V\left[\frac{n_{10}}{n_{+0}} \mid \boldsymbol{N}\right] = \frac{1}{n(1-\hat{\alpha}^*)}\left[1 + \frac{\hat{\alpha}^*}{n(1-\hat{\alpha}^*)}\right] \frac{N_{00}N_{10}}{N_{+0}^2} + O(n^{-3}). \tag{34}$$

Finally, it can be shown that the covariance in the final term is equal to zero:

$$\begin{aligned} C\left[\frac{n_{11}}{n_{+1}}, \frac{n_{10}}{n_{+0}} \mid \boldsymbol{N}\right] &= E\left[C\left(\frac{n_{11}}{n_{+1}}, \frac{n_{10}}{n_{+0}} \mid \boldsymbol{N}, n_{+0}, n_{+1}\right) \mid \boldsymbol{N}\right] \\ &\quad + C\left[E\left(\frac{n_{11}}{n_{+1}} \mid \boldsymbol{N}, n_{+0}, n_{+1}\right), E\left(\frac{n_{10}}{n_{+0}} \mid \boldsymbol{N}, n_{+0}, n_{+1}\right) \mid \boldsymbol{N}\right] \\ &= E\left[\frac{1}{n_{+0}n_{+1}} C\left(n_{11}, n_{10} \mid \boldsymbol{N}, n_{+0}, n_{+1}\right) \mid \boldsymbol{N}\right] \\ &\quad + C\left[\frac{1}{n_{+1}} E\left(n_{11} \mid \boldsymbol{N}, n_{+0}, n_{+1}\right), \frac{1}{n_{+0}} E\left(n_{10} \mid \boldsymbol{N}, n_{+0}, n_{+1}\right) \mid \boldsymbol{N}\right] \\ &= 0 + C\left[\frac{1}{n_{+1}} n_{+1}\frac{N_{11}}{N_{+1}}, \frac{1}{n_{+0}} n_{+0}\frac{N_{10}}{N_{+0}} \mid \boldsymbol{N}\right] = 0. \end{aligned} \tag{35}$$

Combining Formulas (33), (34) and (35) with (32) gives:

$$\begin{aligned} V[\hat{\alpha}_c \mid \boldsymbol{N}] &= \frac{N_{+1}^2}{N^2}\frac{1}{n\hat{\alpha}^*}\left[1 + \frac{1-\hat{\alpha}^*}{n\hat{\alpha}^*}\right] \frac{N_{11}N_{01}}{N_{+1}^2} \\ &\quad + \frac{N_{+0}^2}{N^2}\frac{1}{n(1-\hat{\alpha}^*)}\left[1 + \frac{\hat{\alpha}^*}{n(1-\hat{\alpha}^*)}\right] \frac{N_{00}N_{10}}{N_{+0}^2} + O(n^{-3}) \\ &= \frac{1}{n\hat{\alpha}^*}\left[1 + \frac{1-\hat{\alpha}^*}{n\hat{\alpha}^*}\right] \frac{N_{11}N_{01}}{N^2} \\ &\quad + \frac{1}{n(1-\hat{\alpha}^*)}\left[1 + \frac{\hat{\alpha}^*}{n(1-\hat{\alpha}^*)}\right] \frac{N_{00}N_{10}}{N^2} + O(n^{-3}). \end{aligned}$$

Recall from Formula (31) that $\mathrm{V}\left[\hat{\alpha}_c\right] = \mathrm{E}\left[V[\hat{\alpha}_c \mid \boldsymbol{N}]\right] = \mathrm{E}\left[\mathrm{E}\left[V[\hat{\alpha}_c \mid \boldsymbol{N}] \mid N_{+1}\right]\right]$. Hence,

$$\begin{aligned} V[\hat{\alpha}_c] = E\Bigg[&\frac{1}{n\hat{\alpha}^*}\left(1 + \frac{1-\hat{\alpha}^*}{n\hat{\alpha}^*}\right) E\left(\frac{N_{11}N_{01}}{N^2} \mid N_{+1}\right) \\ &+ \frac{1}{n(1-\hat{\alpha}^*)}\left(1 + \frac{\hat{\alpha}^*}{n(1-\hat{\alpha}^*)}\right) E\left(\frac{N_{00}N_{10}}{N^2} \mid N_{+1}\right)\Bigg] + O(n^{-3}). \end{aligned} \tag{36}$$

To evaluate the expectations in this expression, we observe that, conditional on the column total N_{+1}, N_{11} is distributed as $Bin(N_{+1}, c_{11})$, where c_{11} is a calibration probability as defined in Section 2.5. Hence,

$$\begin{aligned} E\left[N_{11} \mid N_{+1}\right] &= N_{+1}c_{11} = \frac{N_{+1}\alpha p_{11}}{(1-\alpha)(1-p_{00}) + \alpha p_{11}} \\ V\left[N_{11} \mid N_{+1}\right] &= N_{+1}c_{11}(1-c_{11}). \end{aligned} \tag{37}$$

Similarly, since $N = N_{+1} + N_{+0}$ is fixed,

$$\begin{aligned} E\left[N_{00} \mid N_{+1}\right] &= N_{+0}c_{00} = \frac{N_{+0}(1-\alpha)p_{00}}{(1-\alpha)p_{00} + \alpha(1-p_{11})} \\ V\left[N_{00} \mid N_{+1}\right] &= N_{+0}c_{00}(1-c_{00}). \end{aligned} \tag{38}$$

Using these results, we obtain:

$$\begin{aligned} E\left[\frac{N_{11}N_{01}}{N^2} \mid N_{+1}\right] &= \frac{1}{N^2}E\left[N_{11}N_{01} \mid N_{+1}\right] \\ &= \frac{1}{N^2}E\left[N_{11}(N_{+1} - N_{11}) \mid N_{+1}\right] \\ &= \frac{1}{N^2}\left[N_{+1}E\left[N_{11} \mid N_{+1}\right] - E\left[N_{11}^2 \mid N_{+1}\right]\right] \\ &= \frac{1}{N^2}\left[N_{+1}E\left[N_{11} \mid N_{+1}\right] - V\left[N_{11} \mid N_{+1}\right] - E\left[N_{11} \mid N_{+1}\right]^2\right] \\ &= \frac{1}{N^2}\left[N_{+1}^2 c_{11} - N_{+1}c_{11}(1-c_{11}) - N_{+1}^2 c_{11}^2\right] \\ &= \frac{N_{+1}^2}{N^2}c_{11}(1-c_{11}) + O\left(\frac{1}{N}\right), \end{aligned} \tag{39}$$

and similarly

$$E\left[\frac{N_{00}N_{10}}{N^2} \mid N_{+1}\right] = \frac{N_{+0}^2}{N^2}c_{00}(1-c_{00}) + O\left(\frac{1}{N}\right). \tag{40}$$

Substituting expressions (39) and (40) into (36) and noting that $N_{+1}^2/N^2 = (\hat{\alpha}^*)^2$ and $N_{+0}^2/N^2 = (1-\hat{\alpha}^*)^2$, we obtain:

$$\begin{aligned} V[\hat{\alpha}_c] = E\Bigg[&\frac{\hat{\alpha}^*}{n}\left(1 + \frac{1-\hat{\alpha}^*}{n\hat{\alpha}^*}\right)c_{11}(1-c_{11}) \\ &+ \frac{1-\hat{\alpha}^*}{n}\left(1 + \frac{\hat{\alpha}^*}{n(1-\hat{\alpha}^*)}\right)c_{00}(1-c_{00})\Bigg] + O\left(\max\left[\frac{1}{n^3}, \frac{1}{Nn}\right]\right) \\ = &\left[\frac{E(\hat{\alpha}^*)}{n} + \frac{1-E(\hat{\alpha}^*)}{n^2}\right]c_{11}(1-c_{11}) \\ &+ \left[\frac{1-E(\hat{\alpha}^*)}{n} + \frac{E(\hat{\alpha}^*)}{n^2}\right]c_{00}(1-c_{00}) + O\left(\max\left[\frac{1}{n^3}, \frac{1}{Nn}\right]\right). \end{aligned}$$

Finally, substituting the expressions for $E(\hat{\alpha}^*)$ from (4) and the expressions for c_{11} and c_{00} from (37) and (38), the desired expression (17) is obtained. This concludes the proof of Theorem 3.

Comparing mean squared errors

To conclude, we present the proof of Theorem 4, which essentially shows that the mean squared error (up to and including terms of order $1/n$) of the calibration estimator is lower than that of the misclassification estimator.

Proof (of Theorem 4). Recall that the bias of $\hat{\alpha}_p$ as an estimator for α is given by

$$B\left[\hat{\alpha}_p\right] = \frac{p_{00} - p_{11}}{n(p_{00} + p_{11} - 1)} + O\left(\frac{1}{n^2}\right).$$

Hence, $(B\left[\hat{\alpha}_p\right])^2 = O(1/n^2)$ is not relevant for $\widetilde{MSE}[\hat{\alpha}_p]$. It follows that $\widetilde{MSE}[\hat{\alpha}_p]$ is equal to the variance of $\hat{\alpha}_p$ up to order $1/n$. From (14) we obtain:

$$\widetilde{MSE}[\hat{\alpha}_p] = \frac{1}{n}\left[\frac{(1-\alpha)p_{00}(1-p_{00}) + \alpha p_{11}(1-p_{11})}{(p_{00}+p_{11}-1)^2}\right]. \tag{41}$$

Recall that $\hat{\alpha}_c$ is an unbiased estimator for α, i.e., $B[\hat{\alpha}_c] = 0$. Also recall the notation $\beta = (1-\alpha)(1-p_{00}) + \alpha p_{11}$. It follows from (17) that the variance, and hence the MSE, of $\hat{\alpha}_c$ up to terms of order $1/n$ can be written as:

$$\begin{aligned}\widetilde{MSE}[\hat{\alpha}_c] &= \frac{1}{n}\left[\beta\frac{\alpha p_{11}}{\beta}\left(1 - \frac{\alpha p_{11}}{\beta}\right) + (1-\beta)\frac{(1-\alpha)p_{00}}{1-\beta}\left(1 - \frac{(1-\alpha)p_{00}}{1-\beta}\right)\right] \\ &= \frac{\alpha(1-\alpha)}{n}\left[\frac{(1-p_{00})p_{11}}{\beta} + \frac{p_{00}(1-p_{11})}{1-\beta}\right]. \end{aligned} \tag{42}$$

To prove Expression (18), first note that

$$\frac{(1-p_{00})p_{11}}{\beta} + \frac{p_{00}(1-p_{11})}{1-\beta} = \frac{(1-p_{00})p_{11} + \beta(p_{00}-p_{11})}{\beta(1-\beta)}. \tag{43}$$

The numerator of this equation can be rewritten as follows:

$$\begin{aligned}&(1-p_{00})p_{11} + \beta(p_{00}-p_{11}) \\ &= (1-p_{00})p_{11} + (1-\alpha)p_{00}(1-p_{00}) + \alpha p_{00}p_{11} - (1-\alpha)(1-p_{00})p_{11} - \alpha p_{11}^2 \\ &= (1-\alpha)p_{00}(1-p_{00}) + \alpha p_{00}p_{11} + \alpha(1-p_{00})p_{11} - \alpha p_{11}^2 \\ &= (1-\alpha)p_{00}(1-p_{00}) + \alpha p_{11}(1-p_{11}).\end{aligned}$$

Note that the obtained expression is equal to the numerator of Expression (41). Write $T = (1-\alpha)p_{00}(1-p_{00}) + \alpha p_{11}(1-p_{11})$ for that expression. It follows that

$$\begin{aligned}&\widetilde{MSE}[\hat{\alpha}_p] - \widetilde{MSE}[\hat{\alpha}_c] \\ &= \frac{T}{n(p_{00}+p_{11}-1)^2} - \frac{T\alpha(1-\alpha)}{n\beta(1-\beta)} \\ &= \frac{T}{n(p_{00}+p_{11}-1)^2\beta(1-\beta)}\Big[\beta(1-\beta) - \alpha(1-\alpha)(p_{00}+p_{11}-1)^2\Big].\end{aligned}$$

Writing out the second factor in the last expression gives the following:

$$
\begin{aligned}
&\beta(1-\beta) - \alpha(1-\alpha)(p_{00}+p_{11}-1)^2 \\
&= (1-\alpha)^2 p_{00}(1-p_{00}) + \alpha(1-\alpha)\Big((1-p_{00})(1-p_{11}) + p_{00}p_{11}\Big) + \alpha^2 p_{11}(1-p_{11}) \\
&\qquad - \alpha(1-\alpha)(p_{00}+p_{11}-1)^2 \\
&= (1-\alpha)^2 p_{00}(1-p_{00}) + \alpha(1-\alpha)\Big(p_{00}(1-p_{00}) + p_{11}(1-p_{11})\Big) + \alpha^2 p_{11}(1-p_{11}) \\
&= (1-\alpha)p_{00}(1-p_{00}) + \alpha p_{11}(1-p_{11}) \\
&= T.
\end{aligned}
$$

This concludes the proof of Theorem 4.

Deep, dimensional and multimodal emotion recognition using attention mechanisms

Jan Lucas, Esam Ghaleb, and Stylianos Asteriadis[0000−0002−4298−6870]

Department of Data Science & Knowledge Engineering, Maastricht University,
Paul-Henri Spaaklaan 1, 6229 EN Maastricht, Netherlands
jan-lucas@hetnet.nl
{esam.ghaleb, stelios.asteriadis}@maastrichtuniversity.com

Abstract. Emotion recognition is an increasingly important sub-field in artificial intelligence (AI). Advances in this field could drastically change the way people interact with computers and allow for automation of tasks that currently require a lot of manual work. For example, registering the emotion a subject expresses for a potential advert. Previous work has shown that using multiple modalities, although challenging, is very beneficial. Affective cues in audio and video may not occur simultaneously, and the modalities do not always contribute equally to emotion. This work seeks to apply attention mechanisms to aid in the fusion of audio and video, for the purpose of emotion recognition using state-of-the-art techniques from artificial intelligence and, more specifically, deep neural networks. To achieve this, two forms of attention are used. Embedding attention applies attention on the input of a modality-specific model, allowing recurrent networks to consider multiple input time steps. Bimodal attention fusion applies attention to fuse the output of modality-specific networks. Combining both these attention mechanisms yielded CCCs of 0.62 and 0.72 for arousal and valence respectively on the RECOLA dataset used in AVEC 2016. These results are competitive with the state-of-the-art, underlying the potential of attention mechanisms in multimodal fusion for behavioral signals.

Keywords: Emotion Recognition · Multimodal · Neural Networks · Attention Mechanisms

1 Introduction

Emotion recognition as a field in machine learning tries to automate the identification of emotion in a human subject through various means, including computer vision, signal processing and deep learning. This problem is non-trivial as emotion in itself is an abstract concept that is hard to interpret, and thus many different models have been proposed to describe it [9]. Interpretation of emotional expressivity is hindered by the differences in expression between cultures and even persons. Emotion recognition related experiments and data, often come in the form of two types: acted and spontaneous. In the former category, emotions are many times expressed by an actor, while the spontaneous category

mostly involves video clips of spontaneously expressed emotions. Spontaneous emotion recognition is deemed to be harder since it deals with more genuine expression of emotion, which tends to be more subtle than in the acted case.

The state-of-the-art approaches for emotion recognition make use of multiple modalities. This entails that emotion will be predicted by looking at multiple sources. Emotion can be expressed by, for example, both facial and vocal expression, and taking both of these sources into account leads to better performing models [7, 11, 4]. However, using multiple modalities is challenging. These modalities usually differ in multiple aspects, such as their inherent distributions, synchronization, sampling rate, dimensionality, etc.

This work aims to make use of the effectiveness of transfer learning by utilizing pre-trained networks on multiple modalities and fusing these using attention mechanisms. For this purpose, a new method is proposed, that combines previous work in emotion recognition and neural attention to predict emotions in the valence-arousal emotion spectrum.

1.1 Related Work

Work by Ghaleb et al. [4] proposed a framework, which uses metric learning and combines multiple input modalities that showed an increase in performance for discrete emotion recognition in an acted setting [4]. This work relies on some of the preprocessing techniques developed and tested by Ghaleb et al., since it is expected that they will also be beneficial for continuous emotion recognition. Research in the field of emotion recognition is encouraged by the Audio Visual Emotion Challenge (AVEC), which is a competition that is held yearly as part of the ACM Multimedia conference. The competition features a continuous affect recognition sub-part that sets a benchmark for work in the emotion recognition field. The accompanying baseline uses support vector machines (SVM) for each of the eight used modalities, with modality specific feature extraction and processing. SVMs are subsequently fused with a linear regression model [10]. This work will focus on solving the problem presented in this sub-part using feature embeddings and deep neural networks, which have been shown to be effective in similar works [10, 12, 3, 6]. Furthermore, Wu et al. showed that using only feed-forward networks and attention can achieve results that are comparable to recurrent approaches [11]. The method proposed in this work relies on the architectures used by Zhao et al.[12] and Haifeng et al.[3], in which they show that using stacked LSTMs followed by a dense linear layer per modality provides good results. Brady et al. showed state-of-the-art results on the RECOLA dataset for AVEC 2016 using a Kalman Filter approach to fuse models trained for specific modalities [2].

2 Methods

2.1 Attention

Attention mechanisms consitute a family of techniques that can be applied to selectively focus on parts of a sequence. It was initially proposed by Luong et

al. [8] for the purpose of machine translation using an encoder-decoder network. Here the encoder network encodes the sentence in the source language and the decoder uses this encoding to reproduce the sentence in the target language. The order of the words in the source and target languages are often not aligned, and thus the decoder needs to be able to process the encoding out of order. Attention mechanisms allow this network to selectively focus on relevant parts of the encoding to produce a part of the target sequence. The attention mechanism, specifically the general-dot-product (GDP) variant, calculates an attention vector a_t over each encoder hidden state $\overline{h}_s$, which determines how important each part of the input is. This mechanism is formulated as follows:

$$a_t(s) = \frac{\exp(\text{score}(h_t, \overline{h}_s)}{\sum_{s'} \exp(\text{score}(h_t, \overline{h}_{s'})} \tag{1}$$

$$\text{score}(h_t, \overline{h}_s) = h_t^\top W_a \overline{h}_s \tag{2}$$

Here $h_t^\top$ is the current state of the decoder and matrix W_a is a parameter that is to be learned. The attention vector a_t can then be used to compute a weighted average of the source hidden states [8]. Similar to translation, emotions and their corresponding cues in the data may not be aligned. Thus the use of attention mechanisms could improve the performance of models for emotion recognition by allowing them to focus on the information that is relevant for recognizing emotion.

2.2 Proposed approach

The proposed model follows the works of Zhao et al. and Haifeng et al. [12, 3]. These works successfully use LSTMs for multi-modal affect recognition using features that are similar to the ones used in this work. The model used here is a stacked LSTM with dropout between the layers combined with a dense linear layer, where each model outputs both valence and arousal for each time step. This architecture is called the deep long short-term memory network (DLSTM). In total, the network consists of two LSTM layers followed by a dense linear layer. The first LSTM layer iterates over the input sequence and produces a hidden state for each time step in the input. These hidden states are the input for the second LSTM layer, which in turn produces a new sequence of hidden states. These final hidden states are the input to the linear layer that maps them into the two emotion dimensions. This architecture is expanded with attention-based layers on the embedding level (over time) and the fusion level (over the DLSTM networks). Figure 1 shows the overall network structure.

Embedding attention Attention over the embeddings closely follows the general dot-product (GDP) approach described by Luong et al. for language translation using encoder-decoder architectures [8]. The GDP attention mechanism computes scores for each source hidden state $\overline{h}_s$ in a sequence. A softmax function is applied to these to obtain a distribution which is subsequently used to

construct a weighted combination of the sequence. GPD attention computes the score in the following way: $h_t^\top W_a \overline{h}_s$. Where h_t is the current state of the decoder and $\overline{h}_s$ is a hidden state of the encoder. W_a is a weight matrix that maps the encoder hidden state and the target hidden state to a score and has to be optimized. To adapt this method for affect recognition, the target hidden state h_t is the hidden state of the DLSTM, C_v in figure 1, and the source states are replaced with the embedding vectors in a certain window of size n, represented by $V_{t-\frac{n}{2}} \cdots V_{t+\frac{n}{2}}$ in figure 1. The attention mechanism thus computes the score of an embedding vector depending on the current state of the DLSTM and the contents of the embeddings.

This attention mechanism is applied only to the video modality. As will be explained in section 3, the extracted audio features already contain temporal information and should therefore benefit less from attention.

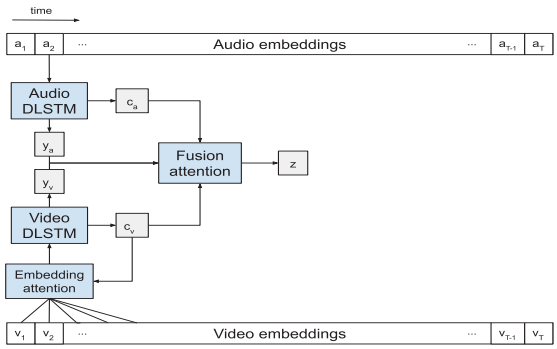


Fig. 1. Overview of the proposed model. a_t and v_t respectively, are the audio and video embeddings at time t. The variables y_m and c_m are the output and hidden state of the DLSTM responsible for modality m, and z represents the fused output.

Bimodal attention fusion The attention concept is also applied to decision fusion. The information that the DLSTMs receive may not allow for a very good prediction of emotion at every time step. For example, at some time t, audio may be more appropriate for emotion recognition, whereas the visual signal may not be carrying significant information. In that case, the output of the audio LSTM should have a higher weight in decision making. GDP attention mechanism outputs a linear combination of the inputs, resulting in equal weight being given to the valence and arousal output of a DLSTM. This is not necessarily the best solution, as at some time step t the audio DLSTM might be very "confident" about its output for arousal and unsure for valence, but the video DLSTM may be confident about valence. The GDP attention mechanism cannot assign different weights for each output dimension separately.

We experimented with an attention approach to decision fusion: Bimodal Attention Fusion. This method uses attention to combine the outputs of modality-specific DLSTM networks. Attention scores are computed by considering all DLSTM hidden states at once. Here $C \in \mathbb{R}^{cn}$, with c being the size of the hidden states and n the number of DLSTMs, is a vector containing the concatenation of the hidden states of all the underlying DLSTMs at time t. $Y_m \in \mathbb{R}^n$ is a vector containing the outputs of the DLSTMs for output modality m. This vector contains either valence or arousal outputs of all the DLSTM networks. The scores and outputs are computed separately for both arousal and valence, because different output dimensions may require different distribution of attention. The calculation of the scores can be reformulated as follows:

$$S_m = C^\top W_m \tag{3}$$

$$a_{m_i} = \frac{\exp(s_i)}{\sum_{j=1}^{n} \exp(s_j)} \tag{4}$$

$$Z = \{z_1, \cdots, z_o\}, \text{where } z_m = \sum_{i=1}^{n} a_{m_i} y_{m_i} \tag{5}$$

The attention vector A, whose elements are described in equation 4, is the matrix product of the DLSTM hidden states in C and the learned weights in W_m followed by an application of the softmax function. The weights $W_m \in \mathbb{R}^{cn \times n}$ map the hidden states to scores per DLSTM and are optimized separately per output modality m. This allows for different mappings from hidden states to scores for both valence and arousal. The attention weights in A_m are used to take a weighted combination of the DLSTM outputs.

Baseline fusion methods The effectiveness of the attention mechanism is assessed by comparing it to networks that fuse without attention. The first network realizes fusion as a static combination of the network outputs by a dense linear layer and is named output linear baseline (OLB). When $Y \in \mathbb{R}^{o \times n}$ is a matrix containing the LSTM outputs, with o being the number of output dimensions, and $W \in \mathbb{R}^{n \times 1}$ is a weight matrix, the fused output can be formulated as follows:

$$Z = YW \tag{6}$$

Matrix W is optimized during training and linearly fuses the outputs of the DLSTMs. The second network fuses by using the concatenation of the hidden states of the DLSTMs. This method, named hidden linear baseline (HLB), can be formulated as:

$$Z = C^\top W \tag{7}$$

Just as in equation 3, $C \in \mathbb{R}^{cn}$ is a matrix containing the concatenation of the hidden states of the DLSTMs, but here $W_m \in \mathbb{R}^{cn \times o}$ is a parameter that maps the hidden states directly to the output emotions.

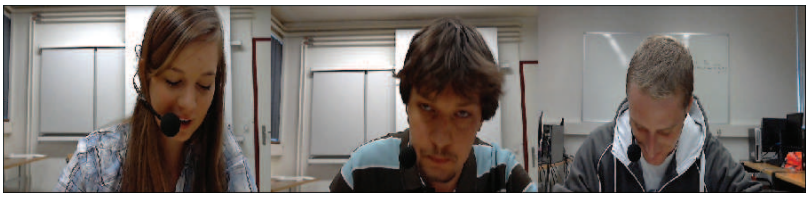


Fig. 2. Example frames from the RECOLA dataset

3 Results and discussion

This section details the experiments performed to determine the performance of the architecture described in section 2.2. A subset of the RECOLA dataset from the University of Fribourg was used in the 2015 and 2016 Audio/Visual Emotion Challenge (AVEC), and is used in this work to train the proposed architecture. The dataset consists of 18 five minute clips in predetermined train and validation partitions [10]. Examples of frames from this dataset are shown in figure 2. This subset of RECOLA contains several feature types, but only the raw video and audio data is used. Emotion is annotated continuously in the valence-arousal space for each video frame. Face extraction is performed on each frame of video and the raw data is transformed using embedding networks. Activations from the last convolutional layer of the VGGFace network is used to extract frame-level video features and VGGish is used to extract audio features from a 960ms window [1, 5].

The models mentioned below are trained by optimizing the mean squared error using the Adam optimizer with a learning rate of 0.01. The DLSTM architecture, described in section 2.2, is optimized using truncated backpropagation through time, following Zhao et al. and Haifeng et al. [12, 3]. The training and test partitions provided with the RECOLA data were used in order to make a fair comparison with the results from the state of the art from AVEC. Model performance is assessed with the Concordance Correlation Coefficient (CCC) measure, which is a common measure of performance in emotion recognition. The CCC is computed for each sequence in test partition of the dataset and averaged over the sequences to show the performance.

3.1 Attention

As explained in section 2.2, attention mechanisms are applied on two points in the proposed model. Embedding attention, which applies attention to the embeddings used as input to the video DLSTM, and fusion attention, which uses attention to fuse the outputs of the DLSTMs. These two methods are evaluated separately in the following experiments.

Embedding attention To assess the effectiveness of applying attention to the video embeddings, the performance of the DLSTM is compared with and

without attention. For embedding attention, a window size of 13 frames is used, allowing the mechanism to consider a segment of half a second. The baseline in this comparison is a DLSTM without any attention, and thus processes the embeddings sequentially, instead of being able to focus on a window.

Initial performance of the model with embedding attention was very poor and stagnated directly after start of training. This was in contrast with the behavior of the model without embedding attention and suggested that training was hindered by the addition of the attention mechanism. To counteract this, the embedding attention layer is bypassed for the first five training epochs. After this startup period, the embedding layer is included again, which significantly improved performance. A possible explanation for this phenomenon is the cyclic dependency between the embedding attention and the hidden layer of the DLSTM. The embedding attention layer uses the hidden state to compute the input to the DLSTM, which in turn affects the hidden state. A meaningless hidden layer could result in poorly attended input, which then maintains the form of the hidden state. To account for random initialization, training and testing is repeated 10 times. For arousal, this resulted in very similar results regardless of the use of attention, with CCCs of 0.15 ($\pm$0.05) and 0.14 ($\pm$0.05) for no attention and embedding attention respectively. A slight improvement was observed for valence with CCC results of 0.36 ($\pm$0.07) without attention and 0.39 ($\pm$0.06) with embedding attention. Even though promising, this difference is not statistically significant, with $p > 0.05$. A bidirectional variant of the DLSTM without attention, since the embedding attention allows the network to use frames ahead of the current time step, and a bigger attention window were evaluated, but these resulted in similar CCC scores.

Table 1. CCC results for the pre-trained uni-modal DLSTMs (left) and their fusion using bimodal attention fusion and baselines (right). Both Bimodal attention and Hidden linear baseline (HLB) successfully fuse the unimodal networks for valence prediction.

Uni-modal	Valence	Arousal
audio	0.42	0.60
video	0.24	0.10

Fusion	Valence	Arousal
Bimodal attention	0.48 ($\pm$0.04)	0.60 ($\pm$0.1)
OLB	0.32 ($\pm$0.03)	0.40 ($\pm$0.08)
HLB	0.48 ($\pm$0.01)	0.64 ($\pm$0.02)

Fusion attention Section 2.2 describes the attention mechanism that can be used to combine the outputs of the uni-modal DLSTM networks. To make a fair comparison with the detailed baselines, two DLSTM networks are pre-trained separately on audio and video, and subsequently frozen before training the fusion mechanisms. This procedure restricts the performance of the model as a whole, but allows for a clear comparison of the fusion methods. The training of the fusion mechanisms is repeated 10 times while using the same pre-trained DLSTMs, to account for random initialization.

The results can be seen in figure 3 and table 1. The CCC values for the pre-trained network are also detailed in table 1. Comparing the methods shows that the proposed attention fusion mechanism significantly improves performance when compared to fusion by linearly combining the DLSTM outputs (OLB). However, its performance is matched by the baseline method that regresses the hidden states of the DLSTMs directly (HLB). For valence, the HLB baseline and bimodal attention mechanism both showed better performance than the unimodal networks they fused. This suggests that the performance of the audio network is slightly increased by fusion with the video modality, but the difference is not large enough for any concrete conclusions. In short, the proposed method seems fuse the uni-modal networks successfully, however its performance does not improve beyond the HLB baseline, which does not use attention.

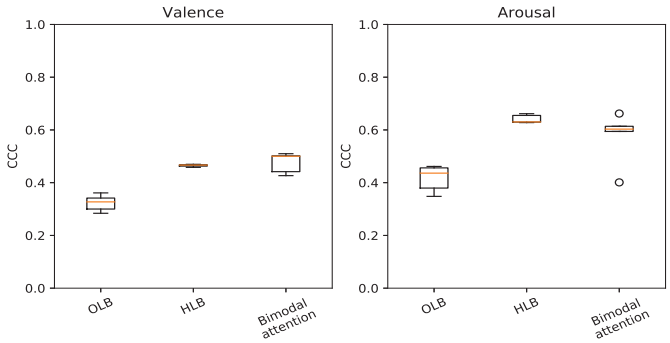


Fig. 3. CCC results for fusion of pre-trained uni-modal DLSTM models using bimodal attention fusion and baseline methods.

3.2 Comparison with the state of the art

The previous sections have each evaluated parts of the proposed architecture. In this section, a comparison with recent works from the literature is performed. For this, the network is trained in an end-to-end fashion using the bimodal attention fusion method and embedding attention on the video modalty. Hyperparameters are optimized empirically, resulting in hidden sizes of 32 and 128 for the audio and video DLSTMs respectively. Work by Haifeng et al. [3] shows that early fusion of features combined with decision level fusion provides improved results for emotion recognition. For this purpose, audio and video features are concatenated per time step to form an early fusion modality. The model described in section 2.2 is extended with a third DLSTM, with a hidden size of 128, that is used on this new modality. The outputs of this DLSTM are fused with the outputs from the audio and video DLSTMs to produce the final model output. The results are compared with a baseline provided by AVEC [10] and the best results on this dataset, achieved by Brady et al. [2]. This comparison is displayed in table 2. The CCC scores obtained by the proposed model for arousal are higher

than the baseline, but are below the ones by Brady et al. However, it should be highlighted here that these two works make use of a wider set of modalities (such as electrodiograms and electrodermal activity, beyond just video and audio), whereas, in the proposed method, only audio and video are considered. For valence, the performance is just below the baseline. The results obtained with the proposed model are comparable to the results of these methods, even though fewer modalities were used and embedding techniques from other domains were reused.

Table 2. Performance of the model proposed in section 2.2 compared to the AVEC baseline and state of the art for this dataset.

	Valence	Arousal
Baseline	0.683	0.639
State of the art (Brady et al.)	0.702	0.82
Proposed	0.62	0.72

4 Conclusions and future work

This work explored combining the information in audio and video data by using attention to fuse the output of networks that were trained on only one modality each. Furthermore, attention was used to spot important video embeddings in temporal windows using the hidden state of an LSTM network.

Fusion of the output modalities using attention shows a significant improvement when compared to a model that does not take the states of the input networks into account. However, it shows similar performance to the baseline that directly regresses the hidden states, suggesting that more improvements should be possible. Usage of embedding attention showed promising results, but this difference is not significant, with a p-value greater than 0.05. Applying attention on the embedding level produced new challenges, that were overcome by using a special training procedure. Future work could investigate the causes for this and explore other, more flexible, variants of this mechanism.

Combining embedding attention and fusion attention yields a model that shows promising performance. Results exhibited improved performance compared to the AVEC baseline for arousal and close performance for valence. The proximity to the baseline and state-of-the art results shows the potential of the proposed method, since the baseline and state-of-the-art methods use more modalities and fine-tuned pre-processing methods. This is in contrast to the proposed method, which uses fewer modalities and reuses feature embeddings from other domains. Other modalities can be easily included in the proposed method and it is expected that this will improve results.

In conclusion, the use of attention mechanisms for emotion recognition shows promising results and can successfully combine information from multiple modalities. Future research could expand on this architecture by experimenting with different forms of attention, extra modalities and different feature embeddings.

References

1. Barsoum, E., Zhang, C., Ferrer, C.C., Zhang, Z.: Training deep networks for facial expression recognition with crowd-sourced label distribution. In: Proceedings of the 18th ACM International Conference on Multimodal Interaction. p. 279–283 (2016). https://doi.org/10.1145/2993148.2993165
2. Brady, K., Gwon, Y., Khorrami, P., Godoy, E., Campbell, W., Dagli, C., Huang, T.S.: Multi-modal audio, video and physiological sensor learning for continuous emotion prediction. In: Proceedings of the 6th International Workshop on Audio/Visual Emotion Challenge. p. 97–104. AVEC '16, Association for Computing Machinery (2016). https://doi.org/10.1145/2988257.2988264
3. Chen, H., Deng, Y., Cheng, S., Wang, Y., Jiang, D., Sahli, H.: Efficient spatial temporal convolutional features for audiovisual continuous affect recognition. In: Proceedings of the 9th International on Audio/Visual Emotion Challenge and Workshop. p. 19–26. AVEC '19, Association for Computing Machinery (2019). https://doi.org/10.1145/3347320.3357690
4. Ghaleb, E., Popa, M., Asteriadis, S.: Metric learning based multimodal audio-visual emotion recognition. IEEE MultiMedia pp. 1–1 (2019). https://doi.org/10.1109/MMUL.2019.2960219
5. Hershey, S., Chaudhuri, S., Ellis, D.P.W., Gemmeke, J.F., Jansen, A., Moore, R.C., Plakal, M., Platt, D., Saurous, R.A., Seybold, B., Slaney, M., Weiss, R.J., Wilson, K.: CNN architectures for large-scale audio classification. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 131–135
6. Huang, J., Tao, J., Liu, B., Lian, Z., Niu, M.: Efficient modeling of long temporal contexts for continuous emotion recognition. In: 2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII). pp. 185–191 (9 2019). https://doi.org/10.1109/ACII.2019.8925452
7. Kossaifi, J., Walecki, R., Panagakis, Y., Shen, J., Schmitt, M., Ringeval, F., Han, J., Pandit, V., Toisoul, A., Schuller, B.W., Star, K., Hajiyev, E., Pantic, M.: SEWA DB: A rich database for audio-visual emotion and sentiment research in the wild. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–1 (2019)
8. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 1412–1421. Association for Computational Linguistics (Sep 2015). https://doi.org/10.18653/v1/D15-1166
9. Poria, S., Cambria, E., Bajpai, R., Hussain, A.: A review of affective computing: From unimodal analysis to multimodal fusion. Information Fusion **37** (02 2017). https://doi.org/10.1016/j.inffus.2017.02.003
10. Valstar, M., Gratch, J., Schuller, B., Ringeval, F., Lalanne, D., Torres Torres, M., Scherer, S., Stratou, G., Cowie, R., Pantic, M.: Avec 2016: Depression, mood, and emotion recognition workshop and challenge. In: Proceedings of the 6th International Workshop on Audio/Visual Emotion Challenge. p. 3–10. AVEC '16, Association for Computing Machinery (2016). https://doi.org/10.1145/2988257.2988258
11. Wu, Z., Zhang, X., Zhi-Xuan, T., Zaki, J., Ong, D.C.: Attending to emotional narratives. In: 2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII). pp. 648–654. IEEE Computer Society (sep 2019). https://doi.org/10.1109/ACII.2019.8925497
12. Zhao, J., Li, R., Chen, S., Jin, Q.: Multi-modal multi-cultural dimensional continues emotion recognition in dyadic interactions. In: Proceedings of the 2018 on Audio/Visual Emotion Challenge and Workshop. p. 65–72. AVEC'18, Association for Computing Machinery (2018). https://doi.org/10.1145/3266302.3266313

A Spiking Neuron Implementation of Genetic Algorithms for Optimization

Siegfried Ludwig[1], Joeri Hartjes[1], Bram Pol[1], Gabriela Rivas[1], and Johan Kwisthout[2][0000−0003−4383−7786]

[1] School for Artificial Intelligence, Radboud University
Montessorilaan 3 6525 HR Nijmegen, Netherlands
siegfried.m.ludwig@protonmail.ch
[2] Donders Center for Cognition, Radboud University
Montessorilaan 3 6525 HR Nijmegen, Netherlands
j.kwisthout@donders.ru.nl

Abstract. We designed freely scalable ensembles of spiking neurons to carry out the operations required to run a genetic algorithm, thereby opening up possibilities for making use of efficient neuromorphic hardware. Two types of implementation are explored that offer a complexity trade-off between computational space and time, with both designs having linear energy complexity. The designs were implemented in a simulator to successfully solve the one-max optimization problem, serving as a proof of concept for running genetic algorithms as spiking neural networks.

Keywords: neuromorphic computing · genetic algorithm · spiking neural networks

1 Introduction

Neuromorphic computing ranges back to the term being coined in 1990 [1], in which the first implementation consisted of very large scale integration (VLSI) with analog components mimicking the biological neural systems. Much research has been done since this time, and in the last few years the energy efficiency of such architectures have become an increasingly dominant research subject. Spiking neural networks (SNN) are known as a type of neuromorphic implementation which have exceptional energy saving properties, compared to other systems [2]. SNNs augment artificial neural networks with the spiking dynamics found in biological neurons [3]. Based on leaky integrate-and-fire (LIF) neurons [4], SNNs transmit information by means of timing and energy spikes, released when the potential difference inside a neuron reaches a certain threshold. This is because such hardware is modeled after the brain in that its activation is event-driven and asynchronous. On top of that, SNN's property of local information storage effectively avoids the von Neumann bottleneck arising from an idling processor while retrieving data from memory [5]. Researching the possibility of implementing various existing algorithms in such SNNs leads the way to a future in which

real life applications of such algorithms currently implemented on von Neumann architectures could be replaced.

Implementing algorithms as SNNs to run them on neuromorphic hardware has been done for sorting [6], constraint satisfaction [7], shortest path and neighborhood subgraph extraction problems [8]. The striking similarity between a genetic sequence and a neural spike train inspires the implementation of a genetic algorithm (GA) as a SNN, which could make use of recent hardware developments in neuromorphic computing.

The use of evolution-inspired algorithms has been proven a viable solution for tackling problems of optimization, bringing in advantages for optimisation over traditional methods. For instance, GA systems [9, 10] may provide the opportunity for difficult problem solving such as multi-objective optimisation [11] and have found applications in various practical settings (see [12] for a review). As in natural evolution, GAs work by modifying the characteristics of individuals in a population across several iterations. This is done by means of reproduction (*crossover*) and random gene mutation. With each run, individuals with an arrangement of genes with a higher fitness value are allowed to preferentially reproduce and carry over their genetic information into the next *generation*. In this study, each individual solution (*chromosome*) is represented as a binary bit sequence, in which each bit represents the value of a gene.

Our main aim was to investigate the feasibility of implementing a GA using spiking neurons with the potential for future implementation on neuromorphic hardware, such as Intel's Loihi [13] or IBM's TrueNorth [14]. Our design consists of binary genetic sequences, which are represented as neuronal spikes and are processed by LIF neurons with context-dependent parameters. The chosen optimization problem is the *one-max problem* due to its simplicity and wide use in the literature on genetic algorithms; the objective of which is to produce a fully active genetic sequence, in this case a fully active spike train. The neural network was implemented and tested using a spiking neuron simulator[3].

We considered two candidate possibilities for encoding the binary genetic sequence in neural ensembles. Firstly, the genetic sequence can be represented sequentially as a spike train, with a spike indicating a 1 and no spike indicating a 0. An ensemble in this design processes one bit at a time. The second way of representing a binary genetic sequence is parallel, using a separate neuron for each position of the genetic sequence. These two encodings are expected to offer a complexity trade-off between computational space and time.

In the following, the high-level architecture of the SNN is presented, followed by details on the sequential and parallel implementations. We then conduct a complexity analysis of both implementations with regard to space, time and energy, in order to assess the tractability of our design.

[3] https://gitlab.socsci.ru.nl/j.kwisthout/neuromorphic-genetic-algorithm

2 High-Level Architecture

The genetic algorithm consists of initializing and evaluating a starting population and then repeatedly performing selection, crossover, mutation, and evaluation on the population until termination. It is implemented as a single recurrent SNN, consisting of specialized neural ensembles for each operation (Figure 1). The topology of the network gives a fitness hierarchy, with the fittest chromosomes being at the top and conversely the least fit chromosomes being at the bottom. The network architecture is static during run-time and no learning of the weights is required.

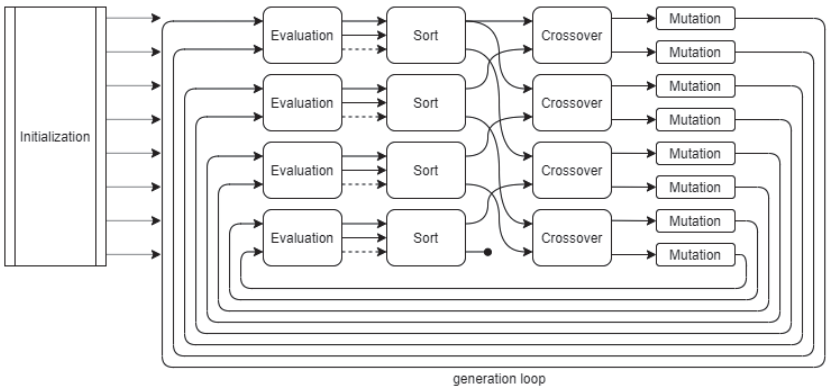


Fig. 1: High-level architecture of the genetic algorithm network, depicted with 8 chromosome lanes as solid arrows (each arrow can represent multiple neural connections in the parallel design). After *mutation*, all chromosomes are connected back to *evaluation* to close the generation loop, resulting in a single large neural network. The dashed arrows from *evaluation* to *sort* represent the evaluation result. To increase the number of chromosomes, the pattern of the second pair of the four lane pairs is repeated.

After initialization, the chromosomes enter the evaluation ensembles in pairs, where they are evaluated against each other and then potentially swapped to bring the chromosome with higher fitness to the top. This setup corresponds to a single pairwise bubble sort step and over time ranks chromosomes by their fitness, which is necessary for selection. The use of only a limited number of bubble sort steps in each generation will lead to incomplete sorting, but is more efficient and leads to some variety in the ranking of the chromosomes while still avoiding the removal of very promising individuals from the bottom. This design is related to the ranking selection mechanism [15].

Selection is implemented in the connections from the sorting ensembles to the crossover ensembles, by eliminating the bottom chromosome and connecting

the top chromosome twice. This results in better solutions propagating more successfully over time. In addition to potentially moving up one lane in the sorting ensemble itself, the winner of the pairwise evaluations moves up by another lane after sorting to ensure upwards mobility, as otherwise the same pairs would be compared each iteration. Conversely, the inferior chromosome moves down by a number of lanes after sorting.

Crossover is then performed on each pair of chromosomes. This reproduction is implemented with a stochastic crossover method, which splits two sequences at a random point and swaps all subsequent genes between the individuals.

After crossover, each chromosome is processed individually in a mutation ensemble. Mutation is carried out by assigning a probability for flipping the activity of each bit in a given sequence. In our designs, we use a probability of $p = \frac{1}{n}$, where n is the length of the chromosome, but other mutation rates are possible. We do not apply mutation on the top two chromosomes of each generation in order to allow for stable one-max solutions. Again, this choice is more up to the design of the genetic algorithm than the implementation as a spiking neural network.

To close the generation loop, the outputs of the mutation ensembles connect back into the evaluation ensembles, forming a recurrent neural network.

Scaling the network up for a larger population size or longer chromosomes is straight-forward beyond a small minimum size, by repeating whole ensembles and repeated elements within certain ensembles.

3 Neural Ensembles for Genetic Algorithms

3.1 Sequential Design

In our sequential design of the GA, the chromosome is processed one bit at a time, which more closely resembles genetic processing in nature. Sequential processing allows a small neural ensemble to process arbitrary lengths of chromosomes over time without itself growing in size. The implementation relies on a lead bit, which precedes every chromosome and is always active. This allows the signaling of the arrival of a new chromosome, ensuring correct processing. In the following, neurons will be ascribed different types based on their function in the ensemble. However, they are all based on the LIF neuron model.

Evaluation Ensemble The sequential one-max evaluation ensemble (Figure 2) makes use of 8 neurons and 11 internal connections. It takes as input two chromosomes and gives as output two chromosomes as well as a spike on a separate neuron serving as an indicator in case the bottom chromosome has a higher fitness than the top chromosome. The membrane potential of the accumulator neuron (ACC) is increased with each active bit in the bottom chromosome, and decreased with each active bit in the top chromosome. Note that the ACC neuron is like all other neuron types used here just a LIF neuron with specific parameters. The activation neuron (A), activated by the lead bit, then makes the ACC

neuron fire or not based on the final membrane potential of the ACC neuron. Only in the case of a membrane potential higher than zero will the indicator neuron fire, and will the chromosomes' ranking switch. A reset neuron (R) is responsible for spiking but suppressing the ACC as to prevent interference of previous chromosome comparisons with current iterations. Clearing the potential of the ACC neuron could alternatively be done using membrane leakage over time, but that would result in a less predictable design.

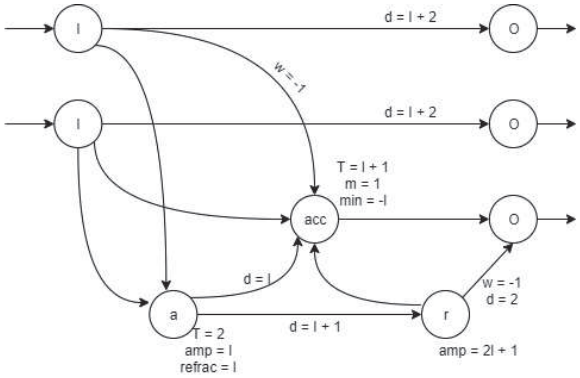


Fig. 2: Sequential one-max evaluation ensemble. (I) Input, (acc) Accumulator neuron, (a) Activation neuron, (r) Reset neuron, (o) Output.

Bubble Sort Ensemble The bubble sort ensemble (Figure 3) consists of 10 neurons and 15 internal connections. It takes two chromosomes plus a fitness indication as input and gives two chromosomes as output. It uses gate (G) neurons to open or close the identity and swap lanes connecting input and output and thereby controlling whether the incoming chromosomes are swapped or propagated as identity. This is achieved by giving the swap gate neurons a threshold of two, which means they can only fire if an input comes from the gate control (GC) neuron. The GC neuron is activated by the gate control activation (GCA) neuron, which takes the fitness indicator input coming from the evaluation ensemble. The GC neuron uses a recurrent connection to keep the swap gates open and the identity gates closed until the chromosomes passed through entirely, at which point it is deactivated by a delayed spike coming from the GCA neuron.

Crossover Ensemble The crossover ensemble (Figure 4) works similarly to the bubble sort ensemble, except that identity and swap gates are not open or closed for the whole chromosome, but switch activation at a random point. It uses 13 neurons and 27 internal connections. The ensemble could be simplified to only use one gate control (GC) neuron as in the bubble sort ensemble, but

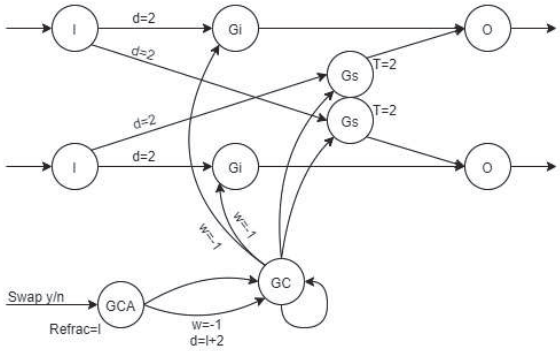


Fig. 3: Bubble sort ensemble in the sequential design. (I) input, (Gi) identity gate, (Gs) swap gate, (GC) gate control, (GCA) gate control activation, (O) output.

has been implemented with two in this project. The random crossover point is implemented via a stochastic (S) neuron and a stochasticity control (SC) neuron. The S neuron gets constant input from the SC neuron, while also generating a random membrane potential each time step. If this combined potential crosses the S neuron's threshold the identity GC neuron is deactivated and the swap GC neuron is activated. If S spikes, The S neuron also deactivates the SC neuron, since only one crossover point is desired.

Mutation Ensemble Finally, the mutation ensemble (Figure 5) stochastically turns a 0 into a 1 and conversely a 1 into a 0, independently for each bit excluding the lead bit. It uses 6 neurons and 12 internal connections. The first stochastic neuron (S1) gets a positive input from each spike in the input and adds a random membrane potential, which can cross the threshold and lead to a spike. A spike from S1 suppresses the ensemble output, thereby turning a 1 into a 0. The other stochastic neuron (S2) always gets input from the control (C) neuron and adds a random membrane potential, but it is suppressed by every spike in the input. If no spike comes from the input, it has a chance of firing and turning the ensemble output from a 0 to a 1. The control neuron is activated and finally deactivated by the control activation (CA) neuron.

Full Network Behavior Each chromosome is passed through the ensembles in its lane, as described in the high-level architecture (see Figure 1). In the sequential design, a chromosome can still be processed in one ensemble while already entering into the next (e.g. crossover to mutation), since here each bit can be handled independently. An exception to this is the evaluation ensemble, which needs to accumulate the full chromosome to make an evaluation. It therefore breaks the time-constant flow through the other ensembles and leads to a time

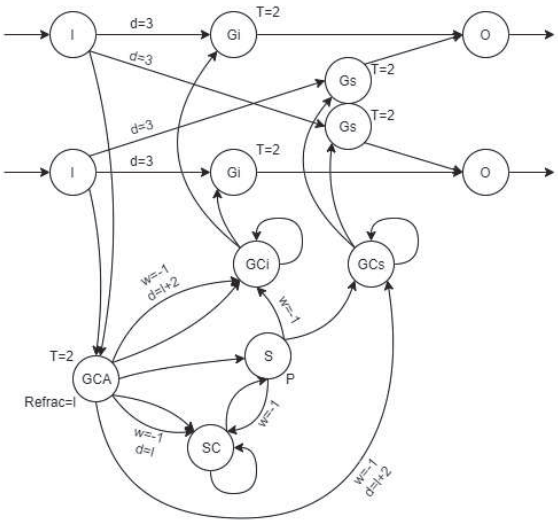


Fig. 4: Crossover ensemble in the sequential design. (I) input, (Gi) identity gate, (Gs) swap gate, (GCi) identity gate control, (GCs) swap gate control, (GCA) gate control activation, (S) stochastic, (SC) stochasticity control, (O) output.

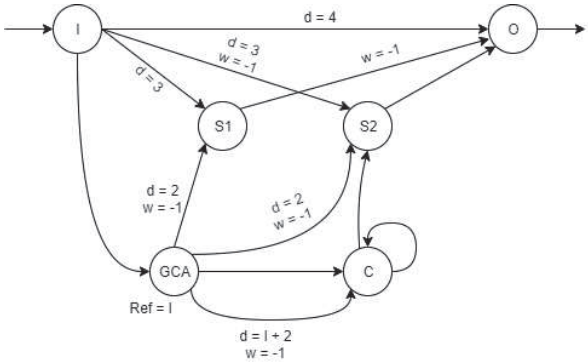


Fig. 5: Mutation ensemble in the sequential design. (I) input, (S1) stochastically flips 1 to 0, (S2) stochastically flips 0 to 1, (C) constant, (GCA) gate control activation, (O) output.

dependency on the chromosome length. On the upside this prevents chromosomes from being longer than the execution cycle, which could otherwise lead to the beginning of the next generation interfering with the end of the last for long chromosomes.

3.2 Parallel Design

In the parallel implementation, every gene of the chromosome gets processed at the same time. Instead of using a single spike train to represent the chromosome, multiple neurons are used that each represent one gene of the chromosome. A set of neurons can then represent the binary code of the chromosome by either spiking or not. Its advantage is that the entire binary code of the chromosomes can be conveyed in a single time step, but requires more neurons as chromosomes get longer. A generation of the entire algorithm in parallel design takes exactly eleven simulation time steps. Again, all neuron types presented here are simple LIF neurons with specific parameters. The different ensembles used in the algorithm will be explained below.

Evaluation & Bubble Sort Ensemble In the parallel design the evaluation step is combined with the bubble sort step. The goal of the evaluation is to have the chromosome with the highest fitness be transferred to the first n output neurons where n is the length of the chromosome. One lead bit is present for each chromosome pair which enables functionality in the other ensembles of the GA, however for this segment it is of no use and therefore linked directly to its corresponding output neuron. Also for this reason the decision was made to omit the lead bit altogether in Figure 6. By taking advantage of all information contained in the chromosome being available at once, the evaluation and sorting ensembles could be combined. This enables the comparison of the fitness through the use of one Accumulator neuron (ACC) to which all input genes are connected (excluding the lead bit). The sign of the connection weights leading to the ACC results in it becoming active only if the lower chromosome has a greater fitness than the top chromosome by at least one gene. Subsequently, the activation of the ACC will determine whether either the identity gates or the swap gates are activated. These are responsible for transferring the activity from the input to the output neuron of the same, or the 'adversarial chromosome', respectively.

Each of the input neurons are connected to both a dedicated identity gate and a dedicated swap gate, with these being connected to the identity neuron or the neuron on the other chromosome in the same position. The connection from the input to the gates is delayed by one time step, however, to allow for synchronous arrival of the spike and the spike coming from the ACC. The connections between the ACC and the gates are weighted such that by default the identity gates have a threshold low enough that a spike from the input neurons will be enough to spike the gate as well while the threshold of the swap gates is too high. As soon as the ACC is activated however this spike is no longer enough for the identity gates, while the extra activation coming from the ACC to the swap gates lowers

their threshold enough to let the spike pass from the input neuron to the correct output neuron on the side of the 'adversarial chromosome'.

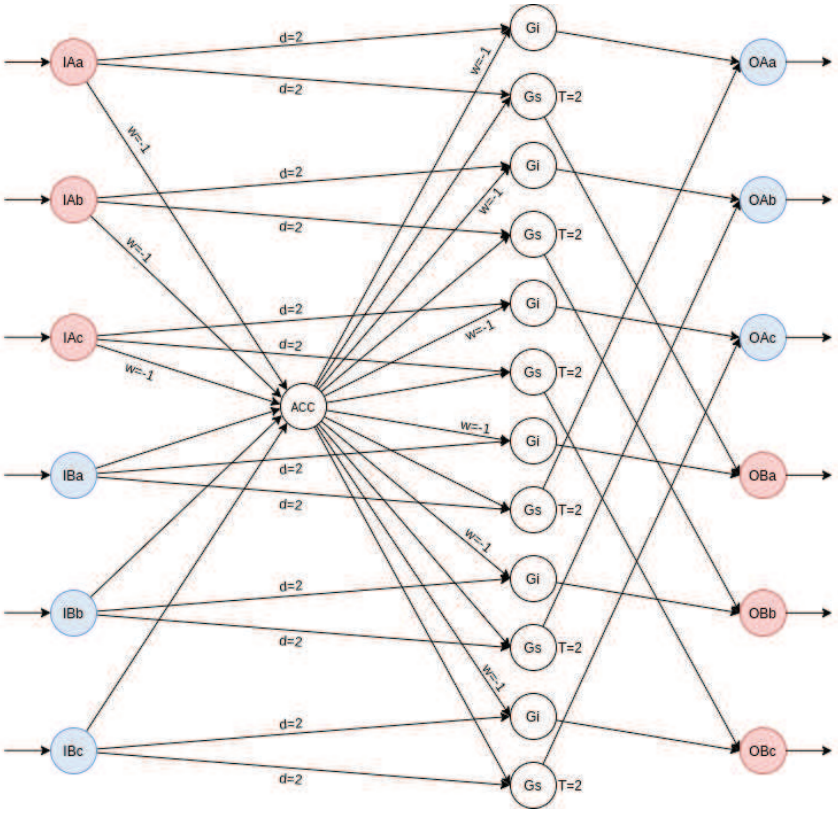


Fig. 6: Ensemble responsible for the evaluation and sorting in the parallel design, applied to a pair of chromosomes consisting of three genes each. Using an accumulator neuron (ACC), the ensemble determines which of the chromosomes has higher fitness and places the winner in the top lanes.

Crossover Ensemble The parallel crossover ensemble can be seen in Figure 7. The first gene of every chromosome always ends up in the same output chromosome. The last gene is always crossed over and ends up in the opposite chromosome. To decide where the genes in between go, a 'random point maker' has been designed (see Figure 8), which is activated by the lead bit. The input to the second layer of the random point maker spikes with a probability of $p = \frac{1}{n-2}$,

where n is the chromosome length. If activated, the node in this second layer transfers this spike to all nodes in the third layer on the same level or below, ensuring that once a gate opens the gates below also open.

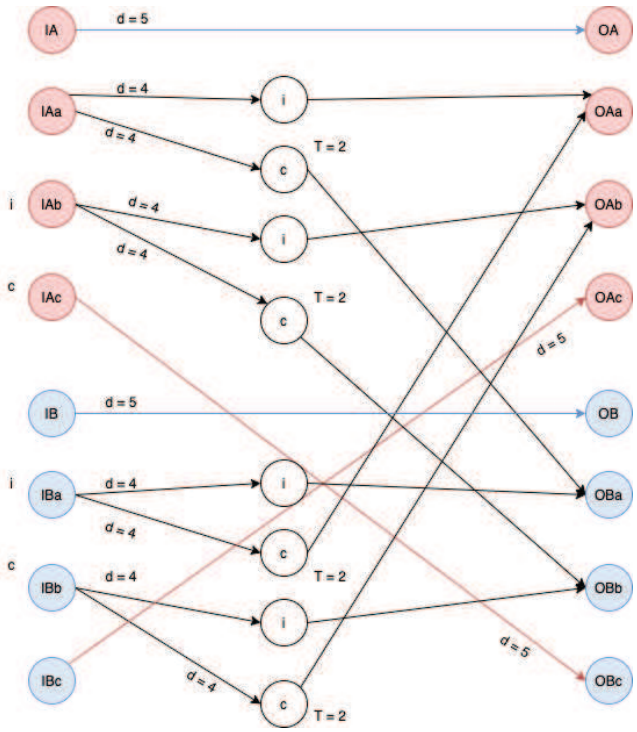


Fig. 7: The parallel crossover ensemble. The first gene of a chromosome is always sent to the same position and the last gene is always crossed over. For the genes in the middle, the random point maker determines whether they are crossed over or not.

The third layer of the random point maker (the gates) connect to the identity and crossover nodes in the crossover ensemble. When a gate neuron of the random point maker gets a spike, it closes the identity gate and opens the crossover gate of both chromosomes at that level. This way, initially the crossover ensemble will transfer genes to the same output chromosome, but at a random point will switch to crossing genes over to the other output chromosome. The crossover ensemble, together with the random point maker, takes five time steps to run for any chromosome length n. The number of neurons in the ensemble is $10n - 11$, meaning linear growth. The number of connections does not show linear growth,

because the connections between the second and third layer of the random point maker grow with $\frac{n^2+n}{2}$, which is quadratic growth. Because of this, the number of connections in the whole ensemble grows quadratically.

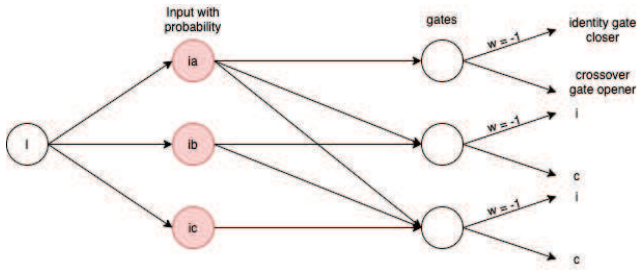


Fig. 8: The random point maker that connects to the gates of the crossover ensemble. This ensemble determines the point of the chromosomes where the identity ends and the swapping of genes with each other starts. It makes sure that there is an equal probability for every point in the chromosome to be the start of crossing over the remaining genes.

Mutation Ensemble The final ensemble in the parallel design is responsible for the stochastic mutation of the genes in the chromosomes, meaning turning a 1 into a 0 or vice versa (Figure 9). The way it is implemented is through assigning a probability P to each of the genes, and therefore neurons, to switch their activity. Except for the lead bit (Ia), every input-neuron (Ib, Ic) is connected to two neurons, and both of their thresholds are influenced by the switching-probability through $T = 2 - P$. A noise factor is present in both intermediate neurons, its function being to add randomness as to whether a neuron will mutate or not. In the diagram the first of the two intermediate neurons is responsible for potentially turning off the activation in case that the input neuron has spiked, and the other is responsible for the opposite. Each of the input neurons is connected directly to its corresponding output neuron, however this connection is delayed such that its spike is delivered synchronously to the potential spike of one of the two intermediate neurons. The role of the lead bit is essential to the mutation ensemble, as its guaranteed activity allows for the potential activation of the two intermediate neurons, which otherwise have no chance of reaching their threshold. Combining the stochastic nature of the intermediate neurons, together with the configuration of the intermediate neurons then has the desired effect of a random mutation of the gene together with the appropriate switching of its value.

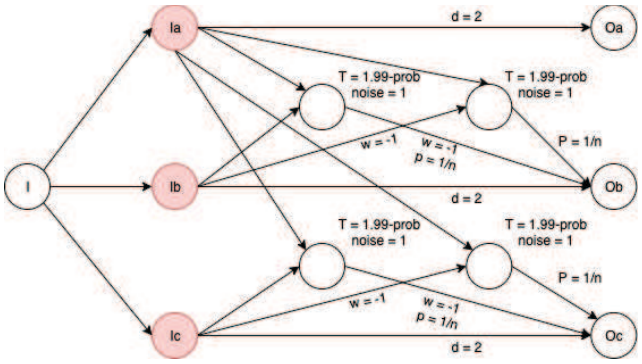


Fig. 9: The parallel mutation ensemble. This ensemble makes sure that every gene's bit has a chance to be swapped.

4 Analysis

A raster plot of the output neurons of the sequential bubble sort ensemble is given in Figure 10. It shows the improving solution quality over time, with the top chromosome reaching one-max, and also shows some resemblance of the fitness hierarchy, with better solutions being closer to the top (subject to imperfect sorting). Figure 11 confirms that the top chromosome in the hierarchy has a higher than average fitness, which specifically shows that even the single pairwise bubble sort step at each generation is enough to at least approximate a fitness ranking.

To assess the tractability of our two designs, a complexity analysis is performed. Computational complexity for neuromorphic computing is considered in terms of space, time, and energy, measured as the number of spikes. For this analysis, all three complexities have been considered with regard to the number of chromosomes and the chromosome length. Comparing the complexity of the sequential and parallel design shows a space-time trade-off between the two (Table 1), with the sequential design requiring less space but more time. Both designs have linear space complexity in the number of chromosomes, both in terms of the number of neurons and the number of connections. The sequential design has much lower space requirements however.

Regarding the chromosome length, the sequential design has constant space complexity, while the parallel design is linear in the number of neurons and quadratic in the number of connections. This is the least favorable of all measured behaviors. It is specifically caused by the current implementation of randomly determining a crossover point. Both designs have constant time complexity in the number of chromosomes, with time measured in simulation steps per generation.

While the parallel design also has constant time complexity in the chromosome length, the sequential design has linear time complexity. The sequential design inherently needs to have at least linear time complexity in the chromo-

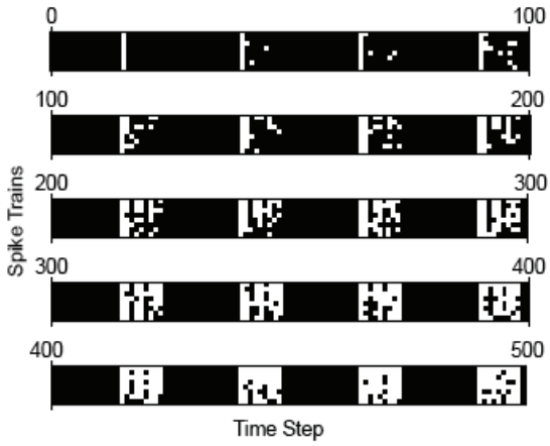


Fig. 10: Raster plot of bubble sort output neurons over time in the sequential design (8 chromosomes of length 8, for 500 steps). Each row of pixels depicts a neural spike train over 500 simulation steps. The solution quality is improving over time, with the top chromosome reaching one-max.

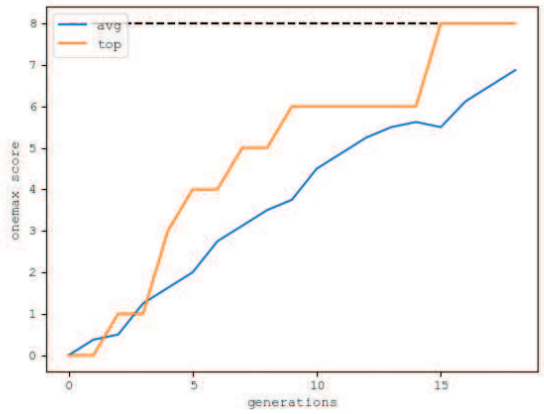


Fig. 11: Average and best solution quality over generations (8 chromosomes of length 8, for 500 steps). The fitness hierarchy results in the top chromosome having better fitness than the average. While this plot comes from the sequential design, the parallel design behaves similarly.

Table 1: Complexity analysis of space, time, and energy (number of spikes) for both the sequential and the parallel design. There is a trade-off between space and time comparing the two designs, with the sequential design requiring less space but more time.

		n_chromosomes		len_chromosomes	
		sequential	parallel	sequential	parallel
space	neurons	$O(n)$	$O(n)$	$O(1)$	$O(n)$
	connections	$O(n)$	$O(n)$	$O(1)$	$O(n^2)$
time		$O(1)$	$O(1)$	$O(n)$	$O(1)$
energy		$O(n)$	$O(n)$	$O(n)$	$O(n)$

some length, as the full chromosome needs to be accessed before an evaluation can be made. This is an advantage for the parallel design, as the full chromosome is available at once. Both designs have linear energy complexity in the number of chromosomes and in the chromosome length, when measuring energy as the average number of spikes required to process one generation.

5 Discussion

A fully functioning genetic algorithm has been successfully implemented as a spiking neural network with two different designs, representing chromosomes sequentially as a spike train over time or as parallel spikes at a single time step. Both implementations are freely scalable beyond a small minimum number of chromosomes, with arbitrary chromosome lengths. The complexity analysis of space, time and energy shows the tractability of this approach with the exception of the quadratically growing number of connections required for the parallel design when increasing chromosome length. The sequential design is at most linear in any of the analyzed complexities.

The design has not yet been implemented on neuromorphic hardware. Since fairly standard leaky integrate-and-fire neurons were used, however, and no learning is required, translating the design to an implementation in neuromorphic hardware should be relatively straight-forward.

For future work, the design of the crossover ensembles could be adapted to support gene lengths of more than one bit (a chromosome consists of a number of genes, which itself could consist of a number of bases/bits). Practically this just means that the random crossover point should only be allowed at transition points between genes, so at fixed intervals. This would allow for more complex behavior of the genetic algorithm.

More work needs to be done on the evaluation strategy, which under the current design requires a unique neural ensemble purpose-built for the optimization task at hand and thereby presents a hurdle for practical application. One possibility for a more general approach would be to train a spiking neural network to

perform approximate evaluations for the given task, instead of hand-engineering the neural ensemble for exact solutions as is performed in this paper.

References

[1] Carver Mead. "Neuromorphic electronic systems". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1629–1636.

[2] Catherine D Schuman et al. "A survey of neuromorphic computing and neural networks in hardware". In: *arXiv preprint arXiv:1705.06963* (2017).

[3] Samanwoy Ghosh-Dastidar and Hojjat Adeli. "Spiking neural networks". In: *International journal of neural systems* 19.04 (2009), pp. 295–308.

[4] Anthony N Burkitt. "A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input". In: *Biological cybernetics* 95.1 (2006), pp. 1–19.

[5] Aaron R Young et al. "A review of spiking neuromorphic hardware communication systems". In: *IEEE Access* 7 (2019), pp. 135606–135620.

[6] Samya Bagchi, Srikrishna S Bhat, and Atul Kumar. "O(1) time sorting algorithms using spiking neurons". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2016, pp. 1037–1043.

[7] Chris Yakopcic et al. "Solving constraint satisfaction problems using the loihi spiking neuromorphic processor". In: *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2020, pp. 1079–1084.

[8] Catherine D Schuman et al. "Shortest path and neighborhood subgraph extraction on a spiking memristive neuromorphic implementation". In: *Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop*. 2019, pp. 1–6.

[9] John H Holland. "Genetic algorithms". In: *Scientific american* 267.1 (1992), pp. 66–73.

[10] John R Koza. *Genetic programming: on the programming of computers by means of natural selection*. Vol. 1. MIT press, 1992.

[11] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. Vol. 16. John Wiley & Sons, 2001.

[12] Manoj Kumar et al. "Genetic algorithm: Review and application". In: *Available at SSRN 3529843* (2010).

[13] Chit-Kwan Lin et al. "Programming spiking neural networks on Intel's Loihi". In: *Computer* 51.3 (2018), pp. 52–61.

[14] Paul A Merolla et al. "A million spiking-neuron integrated circuit with a scalable communication network and interface". In: *Science* 345.6197 (2014), pp. 668–673.

[15] L Darrell Whitley et al. "The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best." In: *Icga*. Vol. 89. Fairfax, VA. 1989, pp. 116–123.

Reputation-driven Decision-making in Networks of Stochastic Agents

David Maoujoud[1] and Gavin Rens[2]

[1] Katholieke Universiteit Leuven, Belgium
david.maoujoud@student.kuleuven.be, david.maoujoud@hotmail.com
[2] Katholieke Universiteit Leuven, Belgium
gavin.rens@kuleuven.be

Abstract. This paper studies multi-agent systems that involve networks of self-interested agents. We propose a Markov Decision Process-derived framework, called RepNet-MDP, tailored to domains in which agent reputation is a key driver of the interactions between agents. The fundamentals are based on the principles of RepNet-POMDP, a framework developed by *Rens et al.* [11] in 2018, but addresses its mathematical inconsistencies and alleviates its intractability by only considering fully observable environments. We furthermore use an online learning algorithm for finding approximate solutions to RepNet-MDPs. In a series of experiments, RepNet agents are shown to be able to adapt their own behavior to the past behavior and reliability of the remaining agents of the network. Finally, our work identifies a limitation of the framework in its current formulation that prevents its agents from learning in circumstances in which they are not a primary actor.

Keywords: Uncertainty · Planning · Reputation · MDP · POMDP.

1 Introduction

Decision-making and learning in multi-agent settings is a multi-faceted area of research [4, 2, 3, 14, 7, 6, 1]. Frameworks used for fully cooperative networks of agents differ vastly from those used for networks of self-interested agents. A primary concern when dealing with self-centered agents is that it makes multi-agent learning inherently more complex than single-agent learning [6, 1]. In fact, each agent needs to take into account the behavior of the entire network of agents when learning its own behavior. Additionally, agent behavior tends to be ever-changing. This *non-stationarity* of agent behavior leads to the loss of policy *convergence* properties that can often be found in single-agent formalisms [1].

In 2018, *Rens et al.* [11] proposed a mathematical framework, called *RepNet-POMDP*, designed to handle partially observable environments in which an agent's reputation among other agents dictates its behavior. The framework was subject to several mathematical inconsistencies, had no working implementation, and had a highly intractable planning algorithm.

Nonetheless, the framework does present some ideas we believe are worth pursuing. Hence, in this paper, we provide an updated version of the framework, called RepNet-MDP. We address the mathematical inconsistencies of the original framework and alleviate its intractability by only considering fully observable environments. We furthermore make use of an online learning algorithm for finding approximate solutions to RepNet-MDPs. The viability of the framework is tested in a series of experiments designed to highlight its strengths and shortcomings.

Section 2 summarizes the relevant background required. Section 3 gives an overview of the work related to our framework. Section 4 provides an intuitive introduction to RepNet-MDPs. Section 5 covers the formal definition of the framework. Section 6 covers planning for RepNet-MDPs. The experimental setup and results are given in Section 7.

2 Background - Markov Decision Processes

A Markov Decision Process (MDP) describes a process for modeling decision-making in stochastic environments [13]. An agent is assumed to move about in an environment, described by a set of states $\mathcal{S}$, by applying actions in $\mathcal{A}$ to the environment. The transition rules of the environment are dictated by the transition model $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, that is, $\mathcal{T}(s, a, s')$ returns the probability of the agent transitioning to state s' upon performing action a in state s. Each action applied to the environment results in a reward for the agent, dictated by the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, that is, $\mathcal{R}(s, a)$ returns the reward received by the agent when performing action a in state s.

The objective of an MDP agent is to maximize its long-term cumulative reward, called *utility*. The utility U of a *finite* state-action sequence, sometimes called *episode*, $E = \langle s_0, a_0, s_1, a_1, ..., s_T, a_T \rangle$ is defined as [10]:

$$U(E) = \sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t),$$

where $\gamma \in [0, 1]$ is called the *discount factor*. An agent advances in the environment by following a *policy* $\pi : \mathcal{S} \times \mathbb{N} \to \mathcal{A}$ that maps each environment state and remaining time-steps to the action the agent should take.

The *expected utility*, or *value*, of being in any state s_t at time-step t, while following policy π, with d time-steps remaining, is defined as:

$$V^\pi(s_t, d) = \mathbb{E}[U(E_t) \,|\, s_t, \pi] = \mathbb{E}\big[\sum_{k=t}^{t+d} \gamma^{k-t} \mathcal{R}(s_k, a_k) \,\big|\, s_t, \pi\big],$$

where E_t is the sub-sequence of E starting at time-step t. An optimal policy $\pi^\star$ is a policy such that

$$\forall s \in \mathcal{S}, \forall d \in \mathbb{N}, \forall \pi : V^\star(s, d) \geq V^\pi(s, d),$$

where $V^\star : \mathcal{S} \times \mathbb{N} \to \mathbb{R}$ is the value function associated with optimal policy $\pi^\star$. This policy satisfies the *optimality equations*, also known as the *Bellman equations* ($\forall s \in \mathcal{S}$):

$$\begin{cases} V^\star(s,d) := \max\limits_{a \in \mathcal{A}} \Big\{ \mathcal{R}(s,a) + \gamma \sum\limits_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') V^\star(s', d-1) \Big\} & d > 1 \\ V^\star(s,1) := \max\limits_{a \in \mathcal{A}} \Big\{ \mathcal{R}(s,a) \Big\} \end{cases}$$

Partially Observable Markov Decision Processes are a common extension of classic MDPs that deal with the problem of partial observability of the environment [13]. To address the agent's inability to observe the exact state of the environment, the observation function $\mathcal{O} : \mathcal{A} \times \mathcal{S} \times \Omega \to [0,1]$, where Ω is the set of observations, is introduced. $\mathcal{O}(a, s', o)$ returns the probability of the agent making observation o after performing action a and the environment transitioning to state s'.

Instead of working with the actual states of the environment, the POMDP agents make use of the notion of belief state $b \in \Delta(\mathcal{S})$ [3], which is a probability distribution over the possible states of the environment. Belief states are updated using the *state estimation function* SE defined as follows:

$$b' := SE(b,a,o) := \Big\{ (s', p) \;\Big|\; s' \in \mathcal{S} \wedge p = \frac{\mathcal{O}(a,s',o) \sum_s \mathcal{T}(s,a,s') b(s)}{P(o|b,a)} \Big\},$$

where $P(o|b,a) = \sum_{s' \in \mathcal{S}} \mathcal{O}(a,s',o) \sum_{s \in \mathcal{S}} \mathcal{T}(s,a,s') b(s)$ is a normalizing constant. We refer to [13] for an extensive overview of POMDPs.

3 Related MDP-based frameworks

Early multi-agent frameworks, such as Multi-agent Markov Decision Processes (MMDPs) [4] and Decentralized Partially Observable MDPs (Dec-POMDPs) [2, 3], operate under the assumption that the agents are selfless and have a common goal. Consequently, planning can be centralized, that is, each agent's policy can be computed by central unit, before being distributed amid the agents for execution [14]. Dec-POMDPs furthermore differ from MMDPs in that states are no longer fully observable, meaning that each agent is in possession of its own set of *local observations*.

In 2005, *Gmytrasiewicz et al.* formalized an extension of POMDPs to multi-agent settings, called Interactive-POMDP (I-POMDP) [7]. I-POMDPs are designed for reasoning in networks of selfish agents. I-POMDP agents update their beliefs not only over physical states of the environment but also over models of the other agents in the network. The difficulty of solving I-POMDPs lies in the recursive nature of the models. Consider agent g's belief update in a network inhabited by another agent, say h. A model of agent h may consist of the belief function of said agent h over physical states and models of all other agents.

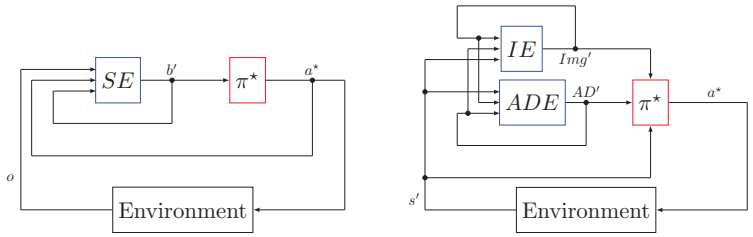
[3] $\Delta(\mathcal{E})$ is the set of probability distributions over the elements of set $\mathcal{E}$.

These models may, in turn, consist of belief functions of their own. This nesting of beliefs could theoretically be infinite, but is overcome by bounding the nesting depth by a finite number n, and solving the problem as a set of POMDPs.

The RepNet-MDP framework [11] simplifies the notion of model by focusing in on key concepts such as behavioral habits and reputation of other agents. While this reduces the insights RepNet agents can have into other agents' behavior, it makes the framework arguably more intuitive. The key, novel notion in the RepNet framework is that of subjective transitions, which have a dependence on the reputation of the agent performing the action.

4 Developing an intuition for RepNet-MDPs

To develop an intuition for the RepNet-MDP framework, parallels between the concepts found in classic POMDPs and RepNet-MDPs can be drawn. In a POMDP, a single agent, placed in a partially observable environment, applies an action $a^\star$ it deems optimal as per its current policy $\pi^\star$, and is sent back an observation o. The state estimation function SE can be thought of as a way of extracting information from said observation o, and storing it in a belief state b'. More specifically, o contains information about the actual state of the environment. The POMDP loop is depicted in Fig. 1a.



(a) The POMDP loop (b) The RepNet-MDP loop

Fig. 1: POMDP and RepNet-MDP loops.

Let us now consider a fully observable environment made up of 3 selfish agents, of which the behavior of the first is dictated by the RepNet-MDP framework. The willingness of the RepNet agent to engage with agents 2 or 3 is to be conditioned by their reputation and behavioral habits. The first agent once again applies action $a^\star$, as per its policy $\pi^\star$. The environment returns its new state s'. In an effort to make well-informed decisions, the RepNet agent should extract the other agents' behavior from s'.

Two functions, analogous to the state estimation function SE in POMDPs, are used to this end: The action distribution estimation function ADE extracts information regarding other agents' behavioral habits. The image estimation function IE informs the RepNet agent on the image all the agents have of each other. The RepNet-MDP loop is shown in Fig. 1b.

Closely tied to the concept of image is the notion of reputation. Specifically, the reputation of any agent in the framework can be seen as a summary of the

information encapsulated by the image. Unlike POMDPs, RepNet-MDPs feature two types of actions, and by extention two types of transition models:

- *Objective* actions, which, when performed, have a real effect on the environment. These actions can be seen as equivalent to actions as they exist in MDPs. The associated transition model is called the objective transition model OT and describes the rules of the environment as they apply to the RepNet agent.
- *Subjective* actions, which, unlike objective actions, are never actually applied to the environment. Instead, they are associated with another transition model called the subjective transition model ST: This transition model describes a RepNet agent's subjective perception of the rules of the environment. This perception is a function of said agent's reputation, and can be used by the agent to aid in its decision-making.

5 Formal definition of RepNet-MDPs

In this section, we will formalize the RepNet-MDP framework[4] introduced in Section 4. A RepNet-MDP $\mathcal{M}$ is defined as a pair of tuples $\mathcal{M} := \langle \Sigma, \Gamma \rangle$, where Σ is called the System tuple and incorporates aspects of the network that apply to all agents, and Γ is called the Agents tuple and contains each agent's subjective understanding of the environment it operates in.

Specifically, a System in a RepNet-MDP Σ is formally defined as a tuple

$$\Sigma := \langle \mathcal{G}, \mathcal{S}, \mathcal{A}, \mathcal{I}, \mathcal{U}, OT \rangle,$$

where:

- $\mathcal{G}$ is the set of agents that can interact with the environment.
- $\mathcal{S}$ is the set of possible states of the environment.
- $\mathcal{A}$ is the set of possible actions, both *objective* and *subjective*. Formally, $\mathcal{A} := \mathcal{A}^o \cup \mathcal{A}^s$, $\mathcal{A}^o \cap \mathcal{A}^s := \varnothing$. The concept of *subjective actions* will be discussed in Section 5.2.
- $\mathcal{I} : \mathcal{G} \times \mathcal{G} \times \mathcal{S} \times \mathcal{A} \to [-1, 1]$ is called the *impact function*. $\mathcal{I}(g, h, s, a)$ returns the impact on agent g that is due to agent h performing action a in state s. This function can be thought of as analogous to a Markov Decision Process's immediate reward function $\mathcal{R}$.
- $\mathcal{U} : [-1, 1] \times [-1, 1] \to [-1, 1]$ is called the *image update function*. Given a current value v, of the image an agent has of another agent, to be updated, and a new expected total impact i, of which the definition will be given shortly, $\mathcal{U}(v, i)$ returns an updated value of the image v'. Many instantiations of this function are possible, two of which are presented in [11]. We will use the following instantiation:

$$\mathcal{U}(v, i) := \begin{cases} v + (1 - v)i & \text{if } i \geq 0 \\ v + (1 + v)i & \text{if } i < 0 \end{cases}$$

[4] The implementation of the RepNet-MDP framework can be found at https://github.com/davidmaoujoud/RepNet-MDP

– $OT : \mathcal{G} \times \mathcal{S} \times \mathcal{A}^o \times \mathcal{S} \to [0,1]$ is called the *objective transition model*. $OT(h, s, a, s')$ returns the probability of the environment transitioning from state s to state s' when objective action a is taken by agent h.

In addition to the global information stored in Σ, each RepNet agent's subjective knowledge is stored in the Agents tuple Γ, formally defined as [5]

$$\Gamma := \big\langle \{ST_g\}, \{AD_g\}, \{Img_g\} \big\rangle,$$

where:

– $ST_g : \mathcal{G} \times \mathcal{S} \times \mathcal{A}^s \times \mathcal{S} \times [-1,1] \to [0,1]$ is called the *subjective transition model* of agent g. $ST_g(h, s, a, r_h, s')$ returns the probability, *as perceived by agent* g, of the environment transitioning from state s to state s' if agent h were to perform subjective action a, and has a reputation r_h according to agent g.
– $AD_g : \mathcal{G} \times \mathcal{S} \to \Delta(\mathcal{A})$ is called the *action distribution* according to agent g. $AD_g(h, s)$ returns a probability distribution over actions in $\mathcal{A}$ for agent h in state s, according to agent g.
– $Img_g : \mathcal{G} \times \mathcal{G} \to [-1,1]$ is called the *image function* according to agent g. $Img_g(h, i)$ returns the image agent i has of agent h according to agent g. Said differently, it returns what g thinks i thinks of h.

As introduced in Section 4, every agent bases its decision-making on the image it believes all agents to have of each other, as well as each agent's behavioral habits. Let g be an agent, whose image at time t is Img_g, and action distribution is AD_g. At time $t+1$, these constructs are updated via the image estimation function IE and action distribution estimation function ADE respectively, to produce Img'_g and AD'_g.

5.1 Image and Reputation

This subsection builds towards the formal definition of the image estimation function IE. To this end, we introduce the notion of *expected total impact*. Consider two agents h and i. In any given state, agent h can perform one of several actions which may or may not have an impact on agent i. Likewise, agent i can be expected to have an impact on agent h when performing an action. The *expected total impact* should be thought of as a way of assigning a numerical value to the *bidirectional* impact these two agents can be expected to have on each other. Additionally, one direction of the impact may be perceived as more important than the other and thus be weighed differently. According to an observing agent, say g, the total impact h is expected to have on, as well as perceive from, i when the environment is in state s, is defined as

$$\begin{aligned} ETI_g(h, i, s, AD_g) := \sum_{a \in \mathcal{A}} \big[& \delta AD_g(i, s)(a) \mathcal{I}(h, i, s, a) \\ & + (1-\delta) AD_g(h, s)(a) \mathcal{I}(i, h, s, a) \big], \end{aligned}$$

[5] $\{X_g\}$ is used as the shorthand notation for $\{X_g \mid g \in \mathcal{G}\}$

where $\delta \in [0,1]$ weighs the importance of impact due to agent h and impact perceived by h.

Agent g's image of other agents, as well as the image it believes all agents to have of each other changes as it observes the agents' behavior. Let Img_g be the current image function of agent g. Concretely, we wish to *update* the image any agent i has of any other agent h, according to the observing agent g ($= Img_g(h,i)$) on the basis of the impact h is expected to have on i ($= ETI_g(h,i,s,AD_g)$). The updated image function Img'_g is computed as follows:

$$\begin{aligned} Img'_g &:= IE(g, Img_g, \alpha, s, AD_g) \\ &:= \Big\{(h,i,t) \;\Big|\; h,i \in \mathcal{G} \wedge t = \mathcal{U}(Img_g(h,i), ETI_g(h,i,s,AD_g))\Big\}, \end{aligned}$$

where s is the current state of the environment, IE is called the *image estimation function*, and $\mathcal{U}$ is the *image update function*.

Finally, the notion of reputation as it is understood in this framework can be thought of as a way of summarizing the information encapsulated by the image.

Say agent g wishes to estimate the reputation of agent h in a network made up of several other agents. It can, to this end, use the image each agent i has of agent h ($= Img_g(h,i)$) as a guiding principle. A first idea might be to take agent h's reputation to be equal to its average image in the network. If, however, some agent i has a poor image of agent h ($Img_g(h,i) < 0$), but agent g has a poor image of agent i ($Img_g(i,g) < 0$), it may be unreasonable for agent g to assume that agent i's opinion of agent h is indicative of agent h's reputation being poor. These concerns are addressed by weighing the image each agent i has of h by the image g has of i. As such, if both images are negative, the resulting reputation of h will not be affected negatively ($Img_g(h,i) \times Img_g(i,g) > 0$). Formally, the reputation of an agent h, according to agent g, is defined as

$$REP_g(h, Img_g) := \frac{1}{|\mathcal{G}'|} \sum_{i \in \mathcal{G}'} Img_g(h,i) \times Img_g(i,g),$$

where $Img_g(i,i) = 1 \; \forall i \in \mathcal{G}$, and $\mathcal{G}' = \mathcal{G}$ if $h \neq g$ and $\mathcal{G}' = \mathcal{G} \setminus \{g\}$ if $h = g$. Recall that in the RepNet framework, reputation influences subjective transition probabilities, which, in turn, influence a RepNet agent's planning.

5.2 Subjective actions and the subjective transition model

In this section, we describe the use of subjective actions and subjective transition models in the RepNet framework. As introduced in Section 4, we make a distinction between the purpose of an *objective* transition model, which describes the actual rules of the environment as they apply to the RepNet agent, and that of a *subjective* transition model, which describes that agent's *subjective* perception of the rules of the environment, this perception being influenced by the reputation of the RepNet agent. To illustrate this further, we will make use of a simple trading example between two agents A and B. Agent A wishes to

trade with agent B, who can either accept or refuse the trade offer. The environment is made up of the set of states $\mathcal{S} = \{s_0, s_1, s_a, s_r\}$. s_0 is the initial state, prior to any trade, s_1 is the state in which agent B is made aware of agent A's trade offer, s_a is the accept state, and s_r is the refuse state. The set of *objective* actions at the disposal of both agents is given by

$$\mathcal{A}^o = \{\texttt{trade\_with\_A}, \texttt{trade\_with\_B}, \texttt{accept}, \texttt{refuse}, \texttt{wait}\}.$$

The transition model of the environment assumed to be deterministic, is given in Fig. 2. *In the eyes of* agent A, agent B's response to a trade offer, characterized

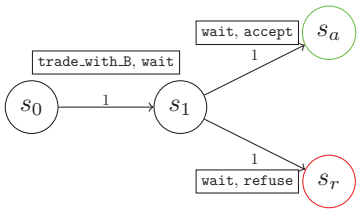


Fig. 2: Transition model of the environment (trading example). Each transition has two *objective* actions, the first action represents agent A's *objective* action, the second action represents agent B's *objective* action.

by transitions $s_1 \to s_a$ and $s_1 \to s_r$, depends on A's reputation. The action taken by agent A during these transitions is `wait`. To make use of the notion of subjective actions, the set of *subjective* actions $\mathcal{A}^s$ will contain the counterpart[6] of `wait` in its *subjective* form, that is, $\mathcal{A}^s = \{\texttt{wait\_s}\}$.

The way agent A makes use of actions in $\mathcal{A}^o$ and $\mathcal{A}^s$ can now be detailed. When *planning* to maximize its expected impact, agent A will make use of the *objective* transition model whenever the action currently investigated has no subjective counterpart in $\mathcal{A}^s$. For instance, the transition probability when investigating action `trade_with_B` is given by $OT(A, s_0, \texttt{trade\_with\_B}, s_1)$. When an action in $\mathcal{A}^o$ has a counterpart in $\mathcal{A}^s$, agent A will make use of the subjective transition model. For instance, the transition probability when investigating action `wait`/`wait_s` is given by $ST_A(A, s_1, \texttt{wait\_s}, s_a, r_A)$. As such, the reputation of agent A is accounted for when agent A *plans* to maximize its expected impact.

5.3 Action distribution

The next step in the formalization of RepNet-MDPs consists in redefining the updating scheme of the action distribution AD_g of each agent g. Say an environment hosting two agents g and h is currently in state s. Agent g has an *a priori* notion of the probability of agent h picking an action a in state s, $P_g(a|h, s, r_h)$. Following agent h performing action a in this state, the environment transitions

[6] We define *counterpart* as a partial mapping $\mathcal{C} : \mathcal{A}^o \to \mathcal{A}^s$. If $\mathcal{C}(a)$ is not defined, then a has no counterpart in $\mathcal{A}^s$.

from state s to state s'. The *a posteriori* probability of agent h performing that same action a in state s in the future is now computed using Bayes' rule:

$$P_g(a|h,s,r_h,s') = \frac{P_g(s'|h,s,r_h,a)P_g(a|h,s)}{\sum_{a'} P_g(s'|h,s,r_h,a')P_g(a'|h,s)}.$$

One can add a *smoothing* technique called *Laplace smoothing* to the present result in an effort to avoid undesirable side effects related to the use of deterministic transition models. The definition for the *action distribution estimation*, obtained after replacing the probabilities by the terms defined previously and applying the smoothing technique, is then given by:

$$\begin{aligned} AD'_g &:= ADE(g,s',AD_g,Img_g) \\ &:= \left\{ (h,s,a,p) \;\middle|\; h \in \mathcal{G} \wedge s \in \mathcal{S} \wedge a \in \mathcal{A} \wedge r_h = REP_g(h,Img_g) \right. \\ &\qquad \left. \wedge\, p = \frac{T_g(h,s,a,s',r_h)AD_g(h,s)(a)+\eta}{\sum_{a'}(T_g(h,s,a',s',r_h)AD_g(h,s)(a')+\eta)} \right\}, \end{aligned}$$

where ADE is called the *action distribution estimation function*, s' is the state the environment transitions to, and η is the *Laplace smoothing parameter*. We refer to the accompanying extensive version of this paper [9] for more details.

Note that to simplify the notation, we combined the *objective* and *subjective* transition models into a single model T_g, called the *global transition model* and formally defined as

$$T_g(h,s,a_h,s',r_h) := \begin{cases} ST_g(h,s,a_h,s',r_h) & \text{if } a_h \in \mathcal{A}^s \\ OT(h,s,a_h,s') & \text{if } a_h \in \mathcal{A}^o \end{cases} \tag{1}$$

6 Planning in the RepNet framework

We now describe optimal behavior in the context of RepNet-MDPs, for finite horizon look-ahead. To simplify the notation, we can define a construct called *epistemic state*. The epistemic state θ_g of agent g is formally defined as a tuple $\theta_g := \langle s, AD_g, Img_g \rangle$, where s is the current state of the environment, AD_g is the current action distribution of agent g, and Img_g is the current image function of agent g. $\theta_g \in \Theta_g$, and Θ_g is called the *epistemic state space*. This set contains every possible combination of physical states of the environment, action distributions, and image functions of agent g.

An agent should perform actions according to the *perceived immediate impact* they have on the agent itself. The perceived immediate impact on agent g resulting from performing action a in state s is defined as

$$PI_g(s,AD_g,a) := \frac{1}{|\mathcal{G}|}\left[\mathcal{I}(g,g,s,a) + \sum_{h \in \mathcal{G}\setminus\{g\}} \sum_{a' \in \mathcal{A}} \mathcal{I}(g,h,s,a')AD_g(h,s)(a')\right],$$

where AD_g is the current action distribution of agent g. The first term describes the immediate self-impact as a consequence of agent g performing action a, while the second term describes the expected immediate impact that the network (i.e., the remaining agents) has on agent g.

Analogously to regular MDPs, a RepNet-MDP agent g strives to maximize its expected discounted perceived impact $\mathbb{E}\big[\sum_{t=0}^{k}\gamma^t PI_{g,t}\big]$, where γ is the *discount factor* and $PI_{g,t}$ is agent g's perceived immediate impact at time-step t. This is accomplished by computing the optimal value function $V_g : \Theta_g \times \mathbb{N} \to \mathbb{R}$ (in a *finite-horizon* setting). It satisfies the *optimality equations*, which are defined as $(\forall \theta_g \in \Theta_g)$

$$\begin{cases} V_g(\theta_g, k) := \max\limits_{a\in\mathcal{A}} \Big\{ PI_g(s, AD_g, a) + \gamma \sum\limits_{s'\in\mathcal{S}} T_g(g, s, a, s', r_g) V_g(\theta'_g, k-1) \Big\} \\ V_g(\theta_g, 1) := \max\limits_{a\in\mathcal{A}} \Big\{ PI_g(s, AD_g, a) \Big\} \end{cases} \tag{2}$$

where $r_g = REP_g(g, Img_g)$, $\theta_g = \langle s, AD_g, Img_g \rangle$, and
$\theta'_g = \langle s', ADE(g, s', AD_g, Img_g), IE(g, Img_g, \alpha, s, AD_g) \rangle$.

In this work, we implement (approximate) online planning [12] instead of exact planning. The general principle of model-based online planning can be described as the interleaving of two phases, the *planning phase*, in which the (PO)MDP performs a look-ahead search of a given depth D, starting at the current environment state, the goal being to determine the most suitable action, and the *execution phase*, in which this action is applied to the environment [12]. In this paper, we make use of an implementation of this approximate technique for the RepNet-MDP framework. We refer to [9] for details on the implementation.

7 Experiments

The goal of the experiments is to showcase the strengths and shortcomings of the framework. To this end, the experimental setup consists of 2 trading scenarios, for which several experiments are conducted. All experiments were conducted with *look-ahead depth* $D = 3$, *Laplace smoothing parameter* $\eta = 0.1$, and *discount factor* $\gamma = 0.7$. Note that these experiments serve as a proof of concept for the RepNet framework and, as such, are not designed to reflect the framework's applicability to problems of realistic scale.

7.1 Experiment 1: Trading between two agents

Let A and B be two agents. Agent A plays the role of the buyer, agent B the role of the seller. Agent A can engage in a trade with agent B, and B can accept or refuse the trade offer. Furthermore, agent A can, prior to making a trade offer, do a good deed in an effort to improve its image in the eyes of agent B.

In this series of experiments, Agent A is managed by the RepNet algorithm. Agent B is run by a simple algorithm that accepts or rejects trade offers made by

agent A according to a set schedule. In particular, agent B is asked to reject trade offers for the 20 first time-steps, accept them for the 60 subsequent time-steps, and finally reject them for the last 20 time-steps.

Two series of experiments are conducted, the first one without making use of subjective actions, the second one by modeling the action of agent A awaiting agent B's response to a trade offer as a subjective action, meaning the outcome of agent A's planning will be influenced by its reputation. A well-designed subjective transition model, schematized in Fig. 3, that realistically reflects how the reputation of agent A may influence the willingness of agent B to accept A's trade offers is put to the test. The variables tracked are the action distribution, image, and by extension the reputation of both agents *in the eyes of agent* A, and frequency at which agent A makes trade offers.

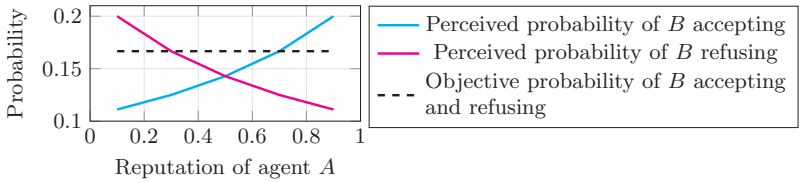


Fig. 3: Perceived probability of agent B accepting and refusing the trade offers, as a function of the self-reputation of agent A.

Fig. 4 shows the evolution of agent A's action distribution for target agent B. Fig. 5 shows the evolution of agent A's self-reputation during the experiment involving the subjective transition model. Note that A's self-reputation, and more generally A's image function, have no bearing on its decision-making if no subjective actions are used (see Equations 1 and 2, T_g makes use of the notion of reputation only for subjective actions). Finally, Fig. 6 shows the evolution of the frequency at which A makes trade offers.

In the first 20 time-steps, B refuses each trade offer. Regardless of the series of experiments, agent A is able to pick up on this via the action distribution. As a consequence, it quickly reduces the frequency at which it attempts to trade with B. In the 60 following time-steps, B is asked to change its behavior and accept each trade offer. Hesitant at first, A gradually increases the frequency at which it attempts to trade with B. Agent A is able to pick up on B reverting back to its old behavior during the final 20 steps.

Additionally making use of a well-designed subjective transition model noticeably improves the RepNet agent's performance. While the trajectories showcase the same key elements, the pace at which agent A is able to adapt improves greatly. The subjective transition model was designed such that agent A believes that its reputation must be good for B to be willing to trade with A (Fig. 3). As such, during the first 20 time-steps, A's relatively poor-in-comparison self-reputation has an immediate negative effect on the *value* it associates with the `trade_with_B` action during the look-ahead search. It quickly becomes more *valuable* to stop trading with B. Similarly, A's reputation needs to be high for it to start trading with B again, explaining the slow increase of the frequency of trade offers at the start of the second phase.

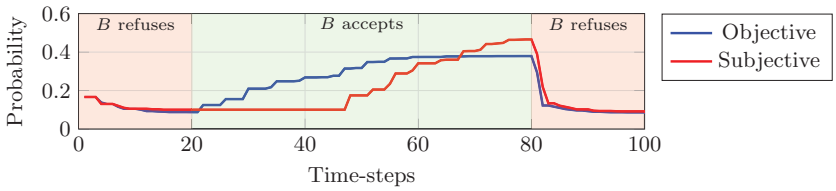


Fig. 4: Probability of B accepting A's trade offers, according to A

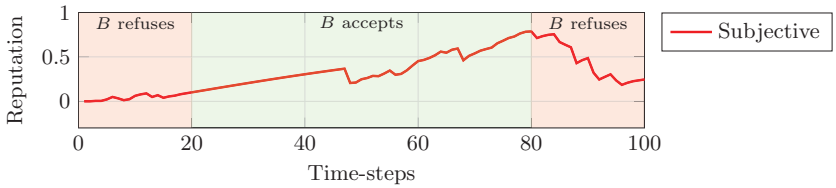


Fig. 5: Reputation of agent A, according to itself.

7.2 Experiment 2: Trading between three agents

Let A, B, and C be three agents. Each agent simultaneously plays the role of buyer and seller, and can thus engage in a trade with any other agent. Each agent can accept or refuse any trade offer made by any remaining agent.

The present scenario is used to verify the ability of a RepNet agent, say agent A, to manage its trades with the two remaining agents B and C, based not only on their behavior towards the agent of interest but also their behavior with each other.

In the first part, agent B is asked to refuse each trade offer made by agent A, while agent C is expected to accept each trade offer coming from A. This portion of the experiments assesses the ability of the RepNet agent (agent A) to accurately determine which agent it is more likely to successfully engage in trades with. In the second part, the roles are switched, and agent B accepts the trade offers, while agent C refuses them. This portion assesses the ability of the agent of interest to *unlearn* what it has learned and adapt its behavior accordingly. In the third and final part, the RepNet agent is asked to not trade with either B or C, that is, to only make use of the `wait` action. Said differently, the optimal action according to its planning, while tracked throughout the experiment, is not performed on the environment. All the while, agents B and C are asked to

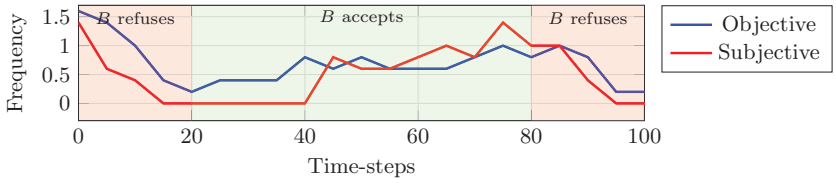


Fig. 6: Frequency of the trade offers made by A, measured in 5 time-step intervals

engage in trades with each other. Agent B is asked to reject all trade offers, while agent C is asked to accept all trade offers. The variables tracked are the action distribution and reputation of B and C *in the eyes of agent* A, as well as the evolution of whom agent A would rather trade with. This portion of the experiment aims at testing the ability of the RepNet agent to draw conclusions on how it should act based on interactions it is not directly affected by.

Fig. 7 shows the evolution of the reputations of agents B and C. Fig. 8 displays the evolution of the probabilities of agents B and C accepting trade offers from agent A. Finally, Fig. 9 shows the evolution of whom agent A would rather trade with.

Agent B is told to refuse, and agent C to accept, each trade offer during the first 33 time-steps. In accordance with the results obtained in Section 7.1, agent A is able to pick up on the other agents' behavioral habits it is affected by. As a result, the reputation of B and its probability of accepting trade offers decrease. Similarly, the reputation of C and its probability of accepting trade offers increase. All the while, agent A chooses to conduct the majority of its trades with C. The following 33 time-steps reverse B's and C's roles. Similarly, agent A is able to adapt its behavior accordingly and ends up trading mostly with B. The reputation of B has increased, while the reputation of C has decreased.

During the last 33 time-steps, agents B and C are tasked with trading with one another while A plays the role of observer, that is, only makes use of the `wait` action. B is asked to refuse all trade offers, while C is asked to accept all trade offers. Interestingly, Fig. 9 shows that, based on its planning, agent A would prefer to keep trading with B, even though the reputation of B decreases and the reputation of C increases in the eyes of A. Said differently, as long as B does not refuse A's offers directly, agent A will prefer to trade with B over C.

The explanation for this is twofold. Firstly, the subjective transition probability of a trade A might want to do with B is, in the eyes of A, conditioned only by A's *own* reputation. As such, B's falling or rising reputation has no bearing on A's decision-making. Secondly, the probability of B accepting (or refusing) A's trade offer, according to A, can only be updated through the direct experience it has with B. As such, the action distribution does not change and can thus not influence A decision-making either.

The simplest way of alleviating this shortcoming is to extend the subjective transition model. Adding the reputation of the agent at the receiving end of the trade offer (e.g., agent B) as a parameter to the subjective transition model would allow agent A to incorporate other agents' reputation in its decision-making process. As such, if the subjective transition probability of B accepting A's trade offer were given by $ST_A(A, \texttt{offer\_state}, \texttt{wait\_s}, \texttt{accept\_state}, r_A, r_B)$, where the newly introduced parameter r_B is B's reputation, agent A could make use of r_B to assist with its decision-making. This comes with the drawback of increasing the complexity of designing the subjective transition model.

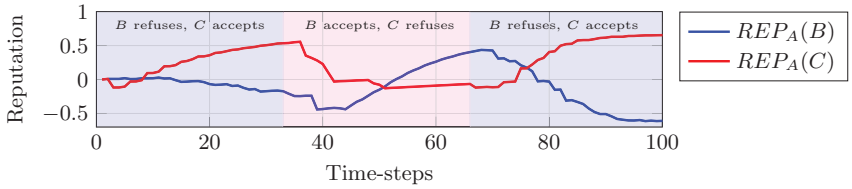


Fig. 7: Reputation of agents B and C, according to A

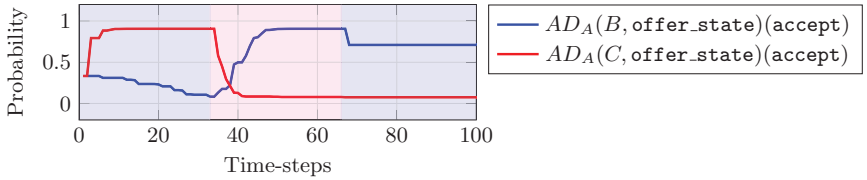


Fig. 8: Probability of agents B and C accepting trade offers from agent A, according to A

8 Summary and future work

In this paper, we revised the multi-agent framework called RepNet introduced by *Rens et al.* [11], addressed its mathematical inconsistencies and proposed a online learning algorithm for finding approximate solutions. The viability of the framework was then tested in a series of experiments.

The current definition of *objective* transitions could be extended to incorporate the reputation of agents other than the RepNet agent. The experimental results showed that the RepNet agent is incapable of adapting its behavior to situations that do not directly affect it. Including the reputation of the agent at the receiving end of a directed action in the *directed* transition model is likely to lead to better-informed decision-making.

We did not address partially observable environments. Many real-world problems do not benefit from full observability, bringing the updated RepNet framework back to a partially observable setting should be considered for future work.

The small-scale experiments conducted in Section 7 served as a proof of concept for the RepNet framework. While applying the framework to problems

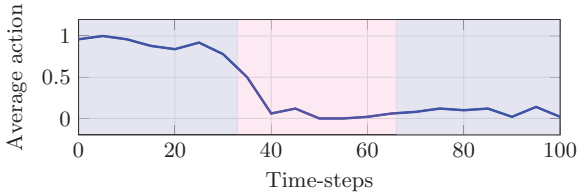


Fig. 9: Average action taken by agent A. Action $a = 0$ corresponds to trading with agent B, action $a = 1$ corresponds to trading with agent C.

of realistic size was beyond the scope of this paper, the absence of large-scale tests does raise questions as to the scalability of this approach. Real-world problems can easily become too complex for transition models to be designed by any one person without leveraging common state features [5]. A compact way to represent real-world state spaces can be achieved by introducing elements of relational logic [5]. From a *logic programming* point of view, a state space is hereby defined by a collection of relations, while a state is an *interpretation* of this collection [8]. Transition models and reward schemes are then represented by *probabilistic rules* [10].

References

1. Abbeel, P.: Learning for Robotics and Control - Value Iteration, CS294-40, University of California, Berkeley (2008), https://inst.eecs.berkeley.edu/ cs294-40/fa08/scribes/lecture2.pdf
2. Becker, R., Zilberstein, S., Lesser, V., Goldman, C.: Solving Transition Independent Decentralized Markov Decision Processes. J. Artif. Intell. Res. (JAIR) **22**, 423–455 (07 2004)
3. Bernstein, D.S., Zilberstein, S., Immerman, N.: The Complexity of Decentralized Control of Markov Decision Processes. CoRR **abs/1301.3836** (2013)
4. Boutilier, C.: Planning, Learning and Coordination in Multiagent Decision Processes. In: Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge. pp. 195–210. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1996)
5. Boutilier, C., Reiter, R., Price, B.: Symbolic Dynamic Programming for First-Order MDPs. pp. 690–700 (01 2001)
6. Busoniu, L., Babuska, R., De Schutter, B.: A Comprehensive Survey of Multiagent Reinforcement Learning. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on **38**, 156 – 172 (04 2008)
7. Doshi, P., Gmytrasiewicz, P.J.: A Framework for Sequential Planning in Multi-Agent Settings. CoRR **abs/1109.2135** (2011)
8. Joshi, S., Kersting, K., Khardon, R.: Self-taught decision theoretic planning with first order decision diagrams. pp. 89–96 (01 2010)
9. Maoujoud, D., Rens, G.: Reputation-driven Decision-making in Networks of Stochastic Agents (2020), https://arxiv.org/abs/2008.11791
10. Nitti, D., Belle, V., De Laet, T., De Raedt, L.: Planning in hybrid relational MDPs. Machine Learning **106**(12), 1905–1932 (Dec 2017)
11. Rens, G., Nayak, A., Meyer, T.: Maximizing Expected Impact in an Agent Reputation Network - Technical Report. CoRR **abs/1805.05230** (2018), http://arxiv.org/abs/1805.05230
12. Ross, S., Pineau, J., Paquet, S., Chaib-draa, B.: Online Planning Algorithms for POMDPs. CoRR **abs/1401.3436** (2014)
13. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edn. (2009)
14. Wiering, M., Otterlo, M.: Reinforcement Learning: State-Of-The-Art, vol. 12 (01 2012)

Learning to Classify Users in the Buyer Modalities Framework to Improve CTR*

Laurent Mertens[1,2], Peter Coopmans[3], and Joost Vennekens[1,2]

[1] KU Leuven, De Nayer Campus, Dept. of Computer Science
J.-P. De Nayerlaan 5, 2860 Sint-Katelijne-Waver, Belgium
firstname.lastname@kuleuven.be
[2] Leuven.AI - KU Leuven Institute for AI, B-3000 Leuven, Belgium
[3] Addrelevance
Grevensmolenweg 28, 3800 Sint-Truiden, Belgium
peter.coopmans@addrelevance.be

Abstract. The Buyer Modalities framework divides buyers into 4 profiles, where each profile has its own specifics as to how it makes its purchasing decisions. We built an online prediction system that categorizes website visitors based on this framework. According to this categorization, a specific banner ad variant tailored to that profile was shown to the visitor, rather than a default "neutral" variant, resulting in a significantly improved CTR.

Keywords: Data mining · Predictive modeling · Ensemble methods · Online advertising.

1 Introduction

The Buyer Modalities framework [3] is a model that defines four distinct psychological profiles of consumers according to how they make their purchasing decisions. These four types – competitive, spontaneous, methodical and humanistic – are illustrated in Fig. 1, and are based on two main axes: decision speed (impulsive vs. deliberate) and rationale (emotional vs logical). It states that each profile reacts to different types of information. If we consider, e.g., the purchase of a new car, people with a methodical profile will be more interested in a detailed list of features of the car as can be found in the brochure, whilst the humanistic profile will be more served with testimonials from people who already own the car.

The implication of this model for advertising is that in order to have an effective campaign, ideally each profile is targeted with an ad tailored toward its information needs. The issue with this of course is that one needs to know the profile of the user, which one typically does not. In order to remedy this issue, we propose a framework that uses historical user-website interaction data

* Supported by the Flanders Innovation & Entrepreneurship TETRA project "Start To Deep Learn".

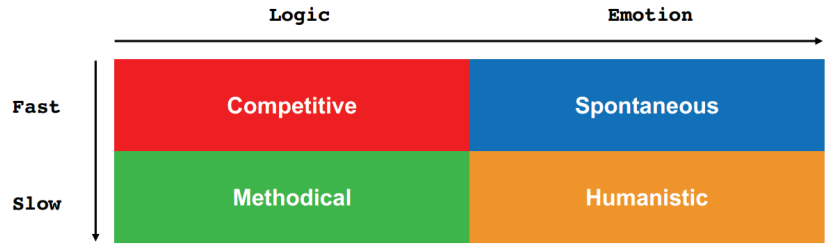


Fig. 1. A schematic overview of the four psychological profiles described by the Buyer Modality framework.

to predict the user profile when needed, and hence allows the ad server to display the appropriate variant for this particular user in a dynamic way. We show that this approach results in a significant increase in the click-through rate (CTR), i.e., the percentage of users viewing a web page who click on an ad displayed on that page.

Ads come in many forms, e.g., pop-ups or "sponsored content", each with its own specifics. Our work involves so-called "banner ads", banner-like graphical ads often displayed at the top or in the margins of a page. Ever since the advent of online advertising, the CTR has served as a key measure for the succes of an ad campaign. Given the multi-billion industry that is online advertising, the question of what makes an ad effective has been given quite some attention.

Specifically for banner ads, [8] look at the effect of the banner ad size, style and orientation on its success. In [1], the authors attempt to predict the CTR, rank according to CTR and categorize into "high" and "low" CTR a set of $\pm$10K banners by using a custom defined set of 43 different visual features. In all three tasks, they manage to consistently outperform the baseline. In contrast, [7] performed two eye tracking studies to investigate the relation between visual design and relatedness to the page content, and visual attention devoted to the ad. In a first study, they show a professionally designed graphical ad to one group of participants and a text-only banner (with the same text as the graphical ad) to another. Besides this, half of the ads (graphical and text-only combined) were related content-wise to the page, whilst the other half were not. They found that none of these parameters had a statistically significant effect on the dwell time. This prompted a second study, in which they showed that dwell time does increase significantly if an ad is relevant to a user's intent or task, rather than to a page's content. Somewhat closer ideologically to our work is [4], who studied the effect of demographically targeting banner ads on users' visual attention and brand evaluation. This kind of targeting focuses on demographic properties of the users such as gender, age and location, and follows the assumption that "similar people act in a similar way". Hence, by tailoring ads to these properties, it should be possible to increase user attention. They found that targeting ads

this way does indeed increase users' visual attention, but not necessarily their brand evaluation.

2 Conceptual setup

In this section we will first provide a conceptual description of our work, followed by a detailed description of the concrete implementation for our use case in the following section.

Conceptually, our setup is the following. Given a particular website that displays ads, we track certain user-website interaction data and use this historical data to extract features to be fed to a predictive model. Ideally, an expert should identify salient elements to be tracked, e.g., specific hyperlinks or other so-called Calls to Action (CTA), that are likely to appeal more to one profile rather than the others.

We distinguish two phases. During a first phase data is being collected to be used to train a predictive model. During this phase, different variations of the same ad targeted to each profile will be displayed at random. This means that a single user can get served different variants of the same ad. When a user clicks on one of the variants, we assign the associated profile to the user to obtain the training targets.

During the second phase, we continue to collect user-website interaction data, and use this data to query the predictive model we obtained in phase one in real time to obtain a user profile. This prediction then determines which ad variant will be shown to this user.

3 Case specifics

Our experiment was performed in collaboration with a commercial partner, Produpress [6], a company that owns amongst others a number of automotive magazines and corresponding websites. We worked with two websites, www.autogids.be (Autogids) and www.moniteurautomobile.be (Moniteur), which are essentially the Dutch and French language versions of the same content. Both sites target a Belgian audience. A large part of the content are extensive car reviews. There were some differences in data collection during training and deployment phases, which we will discuss in the following sections. These differences are due in large part to the fact that data collection was performed by a third party for the first phase, whilst being performed by ourselves for the second phase.

3.1 First phase

In this phase, four different variations of an ad banner for a specific car ad where designed, one for each buyer modality, which were shown on a random basis. The main difference between the variants was the CTA used (i.e., text), rather than

the graphics. If a user clicked on a banner, the profile targeted by the variant becomes the profile of the user. E.g., if a user clicked on the "competitive" variant, when training the model later on we take this user as being a target for the "competitive" profile.

The number of positive samples we collected per profile can be seen in Table 1. Note that some users clicked on more than one ad variant; there were 353 unique users who clicked on a variant for a total of 370 clicks. These users where treated as targets for all the variants they clicked on. The numbers used in this and following tables correspond to the different types as follows: 1 = competitive, 2 = spontaneous, 3 = methodical and 4 = humanistic.

Table 1. Number of collected positive samples per profile

	Autogids	Moniteur	All
1	89	79	168
2	46	30	76
3	38	26	64
4	34	28	62
Total	207	163	370

Besides target labels, also aggregated and custom features for each distinct user were collected. The 7 custom features basically correspond to (the URLs leading to) the main sections of the car reviews, here translated from Dutch: "Read our test report", "View the gallery", "Robotportrait and conclusion", "Tested version", "Users reviews", "Compare this car" and "Find a dealer". With these, the set of features for this specific experiment consists of, per user and over the data collection period (abbreviations correspond to Fig. 2):

- The number of pageviews. (PgV.)
- The average time spent per pageview. (AtP.)
- The number of sessions. (#Ses.)
- The average time spent per session. (ASD.)
- Per custom feature: the number of sessions the user saw this particular content type, i.e., clicked on the corresponding URL. (CT1–CT7)
- Per ad variant: the number of sessions the user was shown this particular ad variant. (AV1–AV4)

Note that at this stage, we did not have any other information besides these aggregated features. This means that we were unable to determine when exactly a user clicked on an ad, which in turn means that all aggregates were determined by also taking into account data from *after* when a user clicked an ad. Ideally, these statistics would have only been determined by using data prior to a click. Fig. 2 shows the average feature values between clicking and non-clicking users for each ad variant separately. This graph clearly illustrates that indeed there appears to be a behavioral difference between both groups of users, as indicated

by the fact that for "click" samples the average values are consistently higher than for "no click" samples.

To further analyze the data, we first checked whether or not we could distinguish between "click" and "no click" samples in general, regardless of profile type. For the remainder, all models were trained using Python's Scikit-learn package [5]. Table 2 contains the average accuracy over 200 Random Forest classifiers, each consisting of 200 trees with `max_depth = 3`. For each iteration, a random selection of negative samples was chosen to complement the positive ones, and 20% of the data was held out as test data. As the data shows, performance is far better than random, although also far from perfect, with Autogids and Moniteur performing very similarly, suggesting that it is indeed possible to predict what users are more inclined to click on an ad, regardless of profile type.

Table 2. Click vs. No Click classification performance over all ads by means of a Random Forest classifier. Average performance over 200 forests of depth 3.

Dataset	Train	Test
Autogids	0.743 ± 0.017	0.679 ± 0.054
Moniteur	0.738 ± 0.018	0.655 ± 0.062

In a next step, we checked to what extend it was possible to distinguish between each pair of profiles. The assumption is that if there is no correlation between user profiles and ad variants, users will randomly click an ad variant and hence it will not be possible to discriminate between ad pairs. To test this hypothesis, we again looked at the average accuracy over 200 Random Forest classifiers with 200 trees each and `max_depth = 3`, with an 80/20 train/test data split. The results are shown in Table 3, and except for the last pair (3 vs 4) show performance that is in line with the "click" vs "no click" scenario. This indicates that it is possible, up to a point, to discriminate users based on the ad variant they clicked. In other words: different people do have a different ad variant preference.

Table 3. Full dataset: Random Forest accuracy per ad pair.

Ad Pair	Train	Test
1 vs 2	0.767	0.650
1 vs 3	0.836	0.689
1 vs 4	0.852	0.685
2 vs 3	0.854	0.576
2 vs 4	0.857	0.612
3 vs 4	0.806	0.501

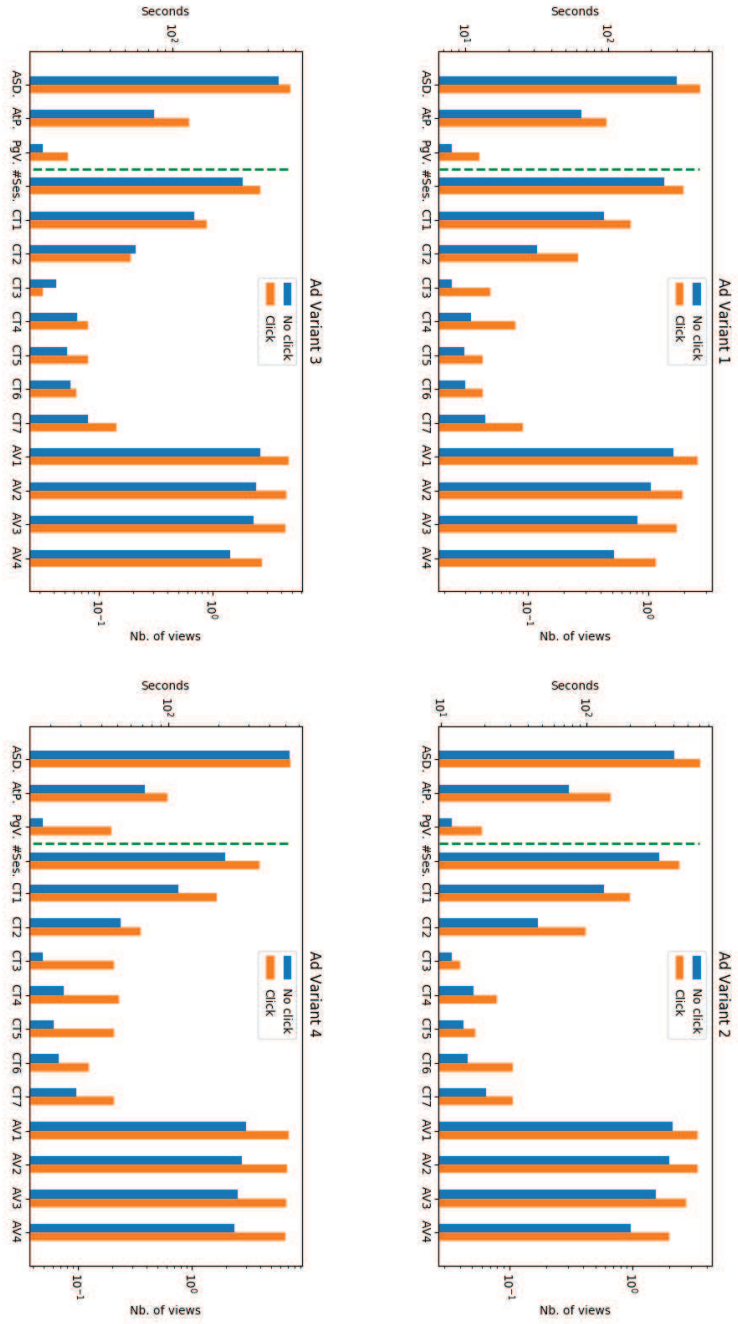


Fig. 2. Averages per feature between "click" and "no click" samples for each ad variant.

Since our assumption is that each user, regardless of whether they click an ad or not, can be described by one of the four buyer modalities, we wish to train a model that always predicts one of these profiles, and does not make a "neutral" prediction. As training data we used all positive samples of all four variants combined (i.e., no negatives), and again opted for a Random Forest model with 200 trees, albeit multiclass this time, to predict one of the four profiles for each user. Train and test sets were stratified so as to have equal ratios of samples per class. Table 4 shows results with `max_depth = 3` and `max_depth = 10` settings by again taking the average over 200 iterations, with an 80/20 data split at each iteration.

Table 4. Full dataset: multiclass Random Forest accuracy

`max_depth`	Train	Test
3	0.515 ± 0.013	0.470 ± 0.022
10	0.946 ± 0.011	0.405 ± 0.043

Performance is considerably lower than with previous experiments, even taking into account the fact that the baseline is 0.25 this time. As can be expected, the experiment with `max_depth = 10` results in overfitting, as apparent by the large discrepancy between train and test accuracies. Nevertheless, we chose to go with this model for phase 2, as our philosophy was that given the low number of samples at our disposal, we preferred the model to overfit on these so that they can serve as stringent prototypes, rather than making a more "diffuse" model.

3.2 Second phase

The second phase was ran in light of a specific advertisement campaign for a new car. The compaign ran for four weeks total; two on Autogids and two on Moniteur. Similar to phase one, five variants of the ad banner were made: one for each profile, plus a "neutral" variant in case the profile of the user could not adequately be predicted. A major difference with the first phase is that we collected the user-website interaction data ourselves. This allowed us to compute the features in an online way. Data was collected by means of a custom JavaScript script, that would send the data to a PHP service to be stored in a MySQL database. Information stored included, a.o., a unique user ID, page visits and clicks.

To allow the website to request a profile for a visiting user, we developed a Python API using CherryPy [2]. Whenever a profile was requested, the known data for this user would be retrieved from the SQL database, and the same features as used during phase one computed on the fly. A prediction was only made if the user had visited the site during at least 3 sessions (including the one at prediction time). If this was the case, we would then verify that the highest profile score returned by the model > 0.35. If so, the corresponding profile would

be returned. In all other cases (also if the user ID was unknown), a default "neutral" profile would be returned. Recall that we used the `max_depth = 10` model described at the end of §3.1, whose performance is shown in Table 4.

We are not allowed to report specific CTR numbers because of contractual obligations to our commercial partners. Hence, we can only report changes w.r.t. the baseline. For both Autogids and Moniteur, a baseline CTR was determined over 14909 and 12502 impressions of the "neutral" banner respectively. This means that the users that belonged to this control group did net get to see a banner based on our predictive system. The CTR for our system was determined over 63284 and 56751 impressions respectively. This does not mean that all users belonging to the test group saw a customized banner, simply that for these users, we attempted to make a prediction. The CTR on banners displayed using our system were 31% and 35% higher than the baseline CTR for Autogids and Moniteur respectively, for an average increase of 33%.

Given this result, it was decided to run a second campaign, for a different car by the same brand as the first campaign, using our system over a period of four weeks, but without further involvement from our part. CTR for this campaign were 129% and 94% higher that the baseline determined in the previous campaign. Unfortunately, a new baseline was not determined and hence these numbers are only reported by way of indication.

4 Conclusion

In this work, we described how the Buyer Modalities framework can be used to improve the CTR on online ads. We built a Random Forest model based on features extracted from aggregated web analytics, and used this model in a system that allows to predict the Buyer Modality profile of a website visitor. Using this predicted profile to dynamically adapt the ads shown to the user resulted in a 33% improvement in CTR compared to the reference user group.

We would like to point out that our method theoretically does not require a new training phase for each new ad, since although the ads change, the user modality profiles do not. This implies that, given careful design of the ad variants, once a model has been trained it should be applicable to any ad campaign. Consequently, by collecting data over several campaigns, the model can also continually be further improved by incrementally retraining the model.

Moreover, for this particular experiment the raw data consisted of aggregated features. We expect that having data available at a more granular level, as collected by ourselves in phase 2, should allow the development of more and better features to further improve the accuracy of the predictive model.

References

1. Azimi, J., Zhang, R., Zhou, Y., Navalpakkam, V., Mao, J., Fern, X.: Visual appearance of display ads and its effect on click through rate. In: Proceedings of

the 21st ACM International Conference on Information and Knowledge Management. p. 495504. CIKM '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2396761.2396826, https://doi.org/10.1145/2396761.2396826
2. CherryPy: https://cherrypy.org/
3. Eisenberg, B., Eisenberg, J.: Waiting for Your Cat to Bark. Thomas Nelson (2006)
4. Kaspar, K., Weber, S.L., Wilbers, A.K.: Personally relevant online advertisements: Effects of demographic targeting on visual attention and brand evaluation. PloS one **14**(2), e0212419–e0212419 (2019)
5. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
6. Produpress: https://www.produpress.be/
7. Resnick, M.L., Albert, W.: The influences of design esthetic, site relevancy and task relevancy on attention to banner advertising. Interacting with Computers **28**(5), 680–694 (2016)
8. Sigel, A., Braun, G., Sena, M.: The impact of banner ad styles on interaction and click-through rates. Issues in information systems **9**(2), 337–342 (2008)

Comparing Exploration Approaches in Deep Reinforcement Learning for Traffic Light Control

Yaniv Oren[1], Rolf A. N. Starre[1], and Frans A. Oliehoek[1]

Technical University Delft, Mekelweg 5 Delft, the Netherlands {y.oren@student., r.a.n.starre@, f.a.oliehoek@}tudelft.nl

Abstract. Identifying the most efficient exploration approach for deep reinforcement learning in traffic light control is not a trivial task, and can be a critical step in the development of reinforcement learning solutions that can effectively reduce traffic congestion. It is common to use baseline dithering methods such as ϵ-greedy. However, the value of more evolved exploration approaches in this setting has not yet been determined. This paper addresses this concern by comparing the performance of the popular deep Q-learning algorithm using one baseline and two state of the art exploration approaches, and their combination. Specifically, ϵ-greedy is used as a baseline, and compared to the exploration approaches Bootstrapped DQN, randomized prior functions, and their combination. This is done in three different traffic scenarios, capturing different traffic profiles. The results obtained suggest that the higher the complexity of the traffic scenario, and the larger the size of the observation space of the agent, the larger the gain from efficient exploration. This is illustrated by the improved performance observed in the agents using efficient exploration and enjoying a larger observation space in the complex traffic scenarios.

Keywords: Reinforcement learning · Traffic optimization · Exploration.

1 Introduction

Traffic congestion is a global predicament. For instance, in the EU alone its cost is estimated to be 1% of the EU's GDP [15]. One approach for reducing this cost is optimization of traffic flows by improving traffic light control policies. To find such policies, reinforcement learning (RL) has a strong appeal as a paradigm that is able to find high performance solutions to sophisticated problems. Research has been done into the application of RL to the problem of traffic light control optimization in the past [1],[4],[13],[14],[18], often specifically concerning application of deep RL algorithms [4],[13],[18].

A fundamental principle of RL is exploration, and the balance between exploration and exploitation. Namely, how much does the agent explore its environment, versus how much it opts for actions that it expects to return the most cumulative reward. Many different exploration approaches exist for reinforcement learning [3],[6],[10]–[12]. These approaches often perform differently in different settings, in addition to having different computational costs [11],[12]. It has been shown that for some RL settings, simple exploration approaches such as ϵ-greedy are insufficient for RL to be able to perform well, or at all [11],[12], which can be caused by the reward function used and the complexity of the specific problem tackled. This illustrates the importance of identifying effective exploration techniques for specific RL settings. To the best of our knowledge, there has not been an attempt to investigate the importance of efficient exploration in deep RL in the setting of traffic light control.

This paper investigates a comparison between different exploration approaches in deep RL for traffic light control. This, to facilitate better deep RL by identifying the value of evolved exploration approaches in this setting, such as higher sample efficiency, or higher final policy score. For that purpose, this paper compares the performance of the popular deep Q-learning algorithm (DQN) [10] using one baseline and two state of the art exploration approaches, and their combination. Specifically, ϵ-greedy is used as a baseline, and compared to the exploration approaches Bootstrapped DQN, randomized prior functions, and their combination. This is done in three different traffic scenarios, ranging from simplified to simulating real traffic, in order to investigate the effect of exploration in different traffic profiles.

This paper first introduces a theoretical background for deep RL and exploration. Second, a description of the exploration techniques compared, along with the modeling of traffic light control as an RL problem, are given. This is then followed by an explanation of the research methodology and the experimental setup, leading to a presentation of the results obtained and their analysis. Last, ethical and epistemic concerns are considered, implications of the work are discussed and conclusions are drawn.

Altogether, the results obtained suggest a link between the complexity of the traffic scenario, the amount of information accessible to the agent, and the gain from efficient exploration. This is illustrated by the improved performance observed in the agents using efficient exploration and enjoying a large observation space, in the complex traffic scenarios.

2 Background

This section introduces background information relevant to the work presented in this paper. First, an overview of reinforcement learning (RL), is given, laying the basis for an introduction to deep RL and a description of the deep Q-learning (DQN) algorithm [10] that follows. Last, the principle of exploration is explained, followed by an overview of dithering [12], deep and directed exploration.

2.1 Reinforcement learning

In RL, an agent operates in an environment. The environment provides information regarding the state the agent is in and what actions it can execute. The environment is usually described in the form of a Markov decision process (MDP), a stochastic control process often used to model decision making in partially stochastic domains. A Markov decision process is represented as a four-tuple, $M = (S, A, P, R)$, where S represents the state space, A the action space, P the transition function and R the reward function.

The agent interacts with the environment by observing a state $s_t \in S$, executing an action $a_t \in A$, and receiving a reward $r_t \in R$ for the action executed. The agent is attempting to learn a policy π, such that the expected reward over time is maximised.

In the Q-learning algorithm [17], the value of state-action pairs is estimated by the agent, using iterative Bellman updates: $Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha[y_t - Q_t(s_t, a_t)]$, where α is the learning rate, and the target $y_t = r_t + \gamma \max_a Q_t(s_{t+1}, a)$. s_{t+1} denotes the new state arrived at after choosing action a_t in state s_t, a any action available at state s_{t+1}, and $0 \leq \gamma \leq 1$, is a discount factor. In many scenarios however, the state-action pair space is too large for the computation or memorisation of each value $Q(s, a)$ to be tractable. To avoid this problem, function estimators such as neural networks can be used to estimate the Q value. This gives rise to the use of neural networks in reinforcement learning, and specifically the deep Q-Learning (or deep Q-networks) algorithm, commonly referred to as DQN [7], [10].

2.2 Deep reinforcement learning with DQN

The DQN algorithm uses deep neural networks to estimate a mapping from states to Q-values [10]. Instead of saving or computing each $Q(s, a)$ value separately, the algorithm learns a parameterized value function $Q(s, a; \theta_t)$. As a result, rather than learning the Q-values directly, the algorithm learns the parameter set θ of the Q-function. The previous Q-learning update then becomes:

$$\theta_{t+1} = \theta_t + \alpha(y_t - Q(s_t, a_t; \theta_t))\nabla_{\theta_t} Q(s_t, a_t; \theta_t)$$

Here, $y_t = r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a; \theta_t)$. To prevent instabilities, the DQN algorithm uses an additional network, termed the target network, θ^- [10]. The target network is the same as the regular, or online, network. However, it only updates every certain τ time-steps, by copying the parameters θ of the online network. This target network is used by the DQN algorithm in the target term, which becomes instead: $y_t = r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a; \theta_t^-)$.

Double DQN [16] is a commonly used modification suggested to the original DQN algorithm, which aims to reduce overoptimism caused by estimation errors, which DQN is prone to. This is done by decoupling the selection of an action from its evaluation [16]. In vanilla DQN, in the term $y_t = r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a; \theta_t^-)$, the agent uses the same network θ^- for both selecting and evaluating an action. The double-DQN algorithm proposes using the online network θ to choose the action, and the target network θ^- to evaluate the choice. The target term y_t used in the double DQN update then becomes:

$$y_t = r_{t+1} + \gamma Q_t(s_{t+1}, \arg\max_a Q_t(s_{t+1}, a; \theta_t); \theta_t^-)$$

2.3 Exploration in reinforcement and deep reinforcement learning

In order to find an optimal policy through experience alone, which is the general premise of RL, the agent must encounter the rewards that are part of an optimal policy at least once. This leads directly to a necessity to explore the environment - if the agent does not explore, how will it encounter valuable rewards that do not lie over its existing policy's path? However, the agent is also expected to efficiently converge into an optimal policy, and not only explore its environment. This leads to one of the fundamental principles of RL - exploration vs exploitation. Different approaches have been developed - ranging from dithering, random action choosing exploration [10], to more evolved notions such as deep and directed exploration [11], [12], and others. These approaches each attempt to achieve efficient exploration through different means - from simplicity of computation to effective analysis of the agent's knowledge and uncertainty.

Dithering exploration The common baseline exploration strategy used in DQN is a dithering exploration method, or ϵ-greedy. In ϵ-greedy, the agent takes a random action (i.e., explores) with probability ϵ, and with probability $1 - \epsilon$ the agent takes the best action according to its current Q value estimation. ϵ-greedy achieves state of the art performance against many popular benchmarks [12], [16]. However, as discussed in [12], in environments where rewards are scarce and distanced in the state-action space, and their values have a large spread, the dithering exploration of ϵ-greedy can take an exponentially long time to arrive at high-valued rewards. This raises the necessity for a more advanced type of exploration, that can be directed over over multiple time steps. These concepts have been coined 'directed exploration' and 'deep exploration' [12].

Deep & directed exploration Directed exploration attempts to improve the efficiency of the agent's exploration by directing it. For example, to previously unexplored or under-explored areas of the state-action space. To achieve a measure of directed exploration, an uncertainty measure can be used - the more uncertain the agent is about the value of some state or action, the more it can prioritize exploration. For directed exploration to be effective, it often doesn't suffice for it to be directed over one, or small number of time steps [12], but must be directed over multiple time steps. The term 'deep exploration' is used to describe such an exploration approach, that is directed over multiple time steps [12].

3 Exploration In Deep Reinforcement Learning for Traffic Light Control

This section will first motivate the choice to evaluate exploration techniques that focus on deep and directed exploration. Following that, the agent used in the experiments is outlined, and the exploration approaches it implements are described. Last, the modelling of traffic light control as an RL problem used in this paper is presented.

3.1 Motivation

This paper opts to specifically identify the value of exploration approaches that focus on achieving deep exploration, in the setting of traffic light control. These approaches have been chosen not only for being state of the art in this field, but also for their potential in this setting. In heavy traffic scenarios, suboptimal actions may carry a long term effect. They may immediately cause congestion, and once there, it may be difficult to return to less congested states [13]. Deep exploration may be able to improve the agent's ability to escape such scenarios, by directing its exploration along a specific path. While this path will not necessarily pay in the short run, it may allow the agent to recover from congestion in the longer run. Additionally, the ability of the agent to more efficiently explore areas of the state action space that lie beyond areas plagued by negative rewards, as a result of employing deep exploration, may allow the agent to learn optimal policies that will otherwise be unlikely for a dithering agent to ever achieve.

3.2 The agent

The agent used in the experiments presented in this paper is a DQN agent, using the double-DQN [16] modification. The agent implements the following exploration mechanisms: ϵ-greedy, Bootstrapped DQN (BDQN) [12] and randomized prior functions [11]. The implementation used in this paper, based on [8] and modified for the setting of traffic light control, allows the agent to use any combination of the three different mechanisms listed above. ϵ-greedy has been described in section 2.3, and has been chosen as it is the common baseline exploration used in DQN [10],[16]. BDQN and randomized prior functions have been chosen as state of the art exploration approaches that aim to achieve efficient deep exploration, and in addition, for their ability to elegantly combine for even better exploration. BDQN and randomized prior functions are described below.

Bootstrapped DQN BDQN has been developed in an attempt to achieve a measure of deep exploration, discussed in section 2.3. In order to achieve deep exploration, BDQN approximates a distribution over Q-values, using a bootstrap. Bootstrapping is a technique used to approximate a population distribution from a sample distribution, using random sampling with replacement [5].

The bootstrap can be implemented efficiently using a shared neural network with several heads. The shared network's role is to learn a feature representation, while each head is providing an independent Q-values estimation. A visualization can be found in figure 1. In learning, the algorithm randomly samples an estimator (a 'head') out of the bootstrap, and follows the policy which is optimal for that estimator for some number of steps greedily or ϵ-greedily. In the experiments done in this paper ϵ-greedy is used. The resulting experiences are gathered in a buffer, and are available for all estimators to learn from, under some probability that decides which experiences will be available to which estimator. Each estimator is trained against its own target network / target network head.

In evaluation, an ensemble voting policy is used to evaluate which action has been chosen by most heads. If there is no majority vote, an arbitrary choice is made between the actions chosen by the most heads. The action is then chosen and executed.

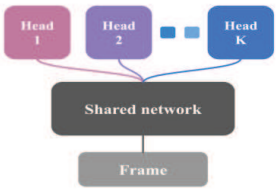


Fig. 1: The BDQN architecture proposed in [12].

BDQN attempts to achieve a measure of deep exploration by following the policy of one of the estimators for some number of steps. For this to be effective, the agent must guarantee that in areas of uncertainty (under-explored areas of the state action space), the different estimators will have different estimations. However, in BDQN this uncertainty, or variety in the Q-value estimations, is only based on the observed data [11]. This can be problematic, because in environments where rewards are very scarce, the agent may learn to believe that there is no reward, and lose all uncertainty, rather than direct its exploration to remote, unexplored areas of the state action space in the hope that they may contain rewards. Such 'prior' drive for exploration, that is independent from the data, is proposed in [11] in the form of randomized prior functions.

Randomized prior functions While usable with a regular DQN agent, the randomized prior functions algorithm is designed to be combined with the BDQN model. To achieve independent uncertainty, the randomized prior functions model consists of one additional neural network for each Q-value estimator, or one shared neural network with one head for each estimator. This additional network or head p is combined with the original estimator f to form the final output Q, through a scaling factor β: $Q = f + \beta p$ [11]. Q is then used in the learning process to minimize the training loss. This results in uncertainty that

is independent from the data: the Q value estimation always includes a neural network initialized with random parameters. No matter the uniformity of experiences the agent encounters (for example, only similar, negative rewards), it will still consider some additional prior 'assumption', in the form of the prior function, in regard to previously un-encountered states. As a result, each estimator will always approximate the Q value of as yet un-encountered states differently. This allows the agent to better direct its exploration, by guaranteeing diversity between the estimations of the different bootstrap heads for previously un-encountered states.

3.3 The model

The modeling of the traffic light control problem as an RL problem used in this paper is follows work done in [13]. The problem is modelled as an MDP $M = (S, A, P, R)$, where S is the state space, A is the action space, P the transition function and R the reward function. The open source traffic simulator SUMO [9] is used to generate the environment.

State space, action space & transition function The state provided to the agent is represented as a set of stacked frames of size $x \in \mathbb{Z}^+$. Each frame is a matrix containing current locations of vehicles in the agent's observation space, and the current traffic light configuration. The observation space of the agent is a square centered at the intersection controlled. Each location of a vehicle is marked with a 1, and empty locations with a 0. The traffic lights configuration is presented in the matrix as numbers between 0 and 1 chosen arbitrarily. Stacking several frames allows the agent to extrapolate vehicle speed from the state representation. $x = 4$ was used in the experiments, to achieve a balance between too much information (complicating the learning process), and too little information (hindering the capacity of the agent to learn effective policies).

The actions available to the agent at each time step are one of two traffic light configurations, representing which lanes receive a green light. The transition function is defined by SUMO.

Reward function The reward function used is a modified version of the reward function developed in [13]. At each time-step t, the agent receives a reward r_t, computed by iterating over all vehicles currently in the agent's observation space, and summing different penalties:

$$r_t = -1.5c - 0.2\sum_{i=1}^{N} e_i - 0.3\sum_{i=1}^{N} d_i - 0.3\sum_{i=1}^{N} w_i$$

This, where i represents the vehicle index. c is a penalty for switching the light configuration, to prevent flickering. e_i is a penalty for sharp decelerations, to penalize emergency stops. d_i is a penalty for the 'delay' of a vehicle, defined as $1 - \frac{vehicle\ speed}{allowed\ speed}$. Finally, w_i is a waiting penalty, defined as 0.5 for the first step of a car standing still, and 1 for any consecutive step.

The modification included removal of a term that punishes teleportation of vehicles (used in SUMO to mark traffic collisions) due to implementation challenges. Additionally, the coefficient of the term c was increased from 0.1 to 1.5, after observation that otherwise the penalty for light switching is barely noticeable with almost any number of cars.

4 Experimental Setup

This section first describes and motivates the methodology used to evaluate the exploration approaches in the setting of traffic light control. This is followed by a description of the three traffic scenarios used to evaluate the exploration approaches.

4.1 Comparison of different exploration methods

To evaluate the impact of exploration in the setting of traffic light control, this paper compares the performance of agents using ϵ-greedy, BDQN [12], randomized prior functions [11] and a combination of the above for exploration, in three different scenarios. The performance is evaluated and averaged over multiple repetitions.

The compared agents As all three exploration approaches investigated are designed to be combined, the performance of the following six agents is compared: a regular DQN agent ad regular DQN agent with a randomized prior function, both employing ϵ-greedy; Two BDQN agents with increasing bootstrap size: 4 & 10 bootstrap samples (or neural network 'heads'); Last, two similar BDQN agents, combining randomized prior functions in their bootstrap mechanism. The number of bootstrap samples has been chosen based on a relation between computational complexity (the larger the bootstrap, the larger the complexity), and gain from the bootstrap (the larger the bootstrap, the better the average performance). As shown empirically in [12], the relative gain from sizes larger than 10 becomes insignificant very quickly. The experimentation with different combinations of those techniques allows to evaluate, in essence, even more exploration techniques, and is the reason it is done in this paper.

The parameters of the agents investigated are not tuned for the specific setting of traffic light control. For the purpose of full reproducibility, a full list of the experiment parameters and agents' hyper-parameters used is available with the code base used in the experiments.

The evaluation The evaluation is done as follows: an experiment is done, for each agent in each scenario and each intersection considered. Each experiment consists of N learning episodes. Every $evaluation\_frequency$ learning episodes, an evaluation phase is ran. In order to reduce sensitivity to stochastic noise from the random nature of the traffic used in the experiments, in the evaluation phase the agent's policy is evaluated over $number\_evaluations$ evaluation episodes, with $\epsilon = 0$. The average episodic reward is then used for evaluation. The exact parameters $evaluation\_frequency$, $number\_evaluations$ used in each experiment are detailed in section 5. To further reduce the impact of stochasticity, the entire experiment is repeated X times for each agent and the results averaged. Finally, the performances of the different agents are plotted against each other, in the form of their averaged evaluations' rewards. The results are presented in section 5.

The different parameters mentioned above were chosen in the following way: N was chosen from experimentation, as the range within which the agents' learning starts to plateau, in order to present the differences between the evaluations of the agents' in the clearest way. The $evaluation\_frequency$ used for each experiment set is chosen to achieve balance between the number of episodes each experiment is ran for, and the number of total evaluation episodes in the experiment. The $number\_evaluations$ parameter has been chosen as a balance between the total number of evaluation phases and the total number of episodes

in each experiment. The larger the number of learning episodes, and lower the evaluation frequency, the larger the *number_evaluations* parameter. To balance computational costs and time constraints with reliability of results, each experiment was chosen to be averaged over $X = 25$ repetitions.

For the purpose of full reproducibility, all random generators used are fully seeded, and the seeds logged. Each experiment is initiated with a different random seed, to guarantee random initialization of the agents' neural networks' weights. The environment's traffic generator however is seeded with the same set of seeds for all experiments. This is done to guarantee that while all agents experimented with are different, they are tested against the same traffic simulations.

4.2 The traffic scenarios

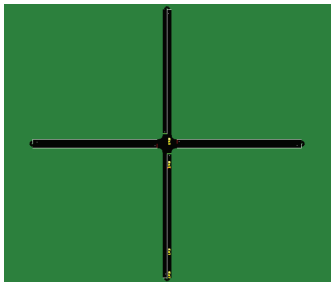
The impact of the exploration approaches investigated in this paper is evaluated using the traffic simulator SUMO [9], in three different traffic scenarios. In the first scenario, one set of experiments is done. In the second and third scenarios, two separate experiment sets are done, evaluating the agent against traffic of slower and faster average speed.

Scenario 1: The grid The first scenario is a basic grid-like road network, with one intersection in the center, and four roads going one in each direction from the intersection: north, east, south and west. A visualization of the grid scenario is presented in figure 2 a. The first scenario is meant to capture a simple, independent intersection profile, that does not consider or experience the behavior of other neighboring intersections. The traffic in this scenario is generated randomly, based on a set number of vehicles over a set spawning time.

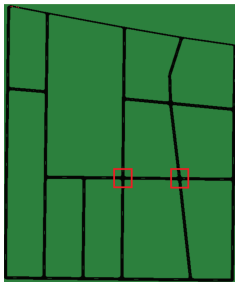
Scenario 2: Simulation of real traffic in Manhattan, New York The second scenario, visualized in figure 2 b, is based on a section of the road map of Manhattan, New York, and is a more complex network containing several interconnected intersections. The specific road map used in our experiments is a 700 m^2 section centered around the corner of Waring and Woodhull Avenues. The map has been imported using SUMO's web-wizard. This scenario means to evaluate the effect of efficient exploration in a more complex traffic profile, where the agent may observe and consider the behavior of neighboring intersections. Manhattan enjoys a grid road-map design, that for the purpose of this work serves as both realistic and practical to use.

The red squares marked in figure 2 point to the two intersections given to the agent to control, as two separate experiments sets in this scenario. These intersections were chosen due to their encapsulation of different traffic profiles. The top intersection, Waring-Woodhull, presents a gentler form of traffic with significantly lower vehicle speed average. The bottom intersection, Eastchester-Waring, experiences much higher average vehicle speed, and more strongly resembles a central road. The throughput of both intersections is rather similar, with a slightly heavier load going through Eastchester-Waring. In every experiment, all intersections in the network except the one controlled by the agent are controlled by SUMO. To generate traffic for the Manhattan scenario, a random routes generator is used, based on real population distributions in the area, for the time of day 08:30 AM to 09:00 AM. The traffic data is imported using SUMO's web-wizard as well.

Scenario 3: Simulation of custom traffic in Manhattan, New York A third scenario is introduced in order to evaluate exploration under heavier traffic settings. This scenario uses the same map, and experiments with the same intersections illustrated in figure 2 b. However, in this scenario a random traffic generator is used for traffic generation, in order to introduce much heavier traffic loads than the ones generated to simulate real traffic. Again, two different sets of experiments are done in this scenario, one on each of the two marked intersections in figure 2 b. The difference of the traffic profiles between the two intersections is similar to the one in the second scenario: the top intersection enjoys slower average speed, and the bottom faster.



(a) The basic grid map.



(b) The Manhattan map. The top red box marks the intersection Waring-Woodhull, and the bottom the intersection Eastchester-Waring.

Fig. 2: The road maps used in the traffic scenarios.

Observation space specification An important difference between the scenarios is the observation space provided to the agent in each one. The observation space provided in the experiments done with the grid scenario and the Eastchester-Waring intersection in either scenario, was $50\,\mathrm{m}^2$. This is done because in all three a larger observation space would be outside the bounds of the environment. Due to the structure of the scenario, it was possible to provide the agent with a larger observation space in the experiments done with the Waring-Woodhull intersection in either scenario. An observation space of $84\,\mathrm{m}^2$ was chosen, to balance complexity (the larger the observation, the more complex the learning) with observation distance. This is done to allow the agent access to more information, which (1) contains the adjacent intersections, enabling the agent to take their behavior into account, and (2) providing the agent with the ability to react to incoming traffic earlier, as a result of the larger observation space.

5 Results

This section presents and analyzes the results obtained in the experiments described in section 4, divided between the different scenarios investigated and intersections controlled.

For each set of experiments, the results presented are the episodic rewards attained in the evaluations, as described in section 4.1. This is presented alongside a plot of the 95% confidence interval of the mean, computed using the standard error of the mean (SEM) [2] of the different experiments for two sample agents. These agents are chosen independently for each experiment: the one that performed, on average, the best, and the one that performed the worst. This is done to illustrate how significant are the differences observed between the evaluations of the different agents in each experiment. Only two agents are presented in order to reduce clatter in the plot. A simple moving average (SMA) of window size 5 is applied to the data presented in order to smoothen the stochastic effect, to facilitate visual analysis of the results.

5.1 Scenario 1: The grid

The results of evaluating the performance of six different agents against the grid scenario can be found in figure 3. The agents' policies are evaluated every five learning episodes, and averaged over three evaluation episodes. Additionally, the 95% confidence intervals of the means of the two sample agents are presented.

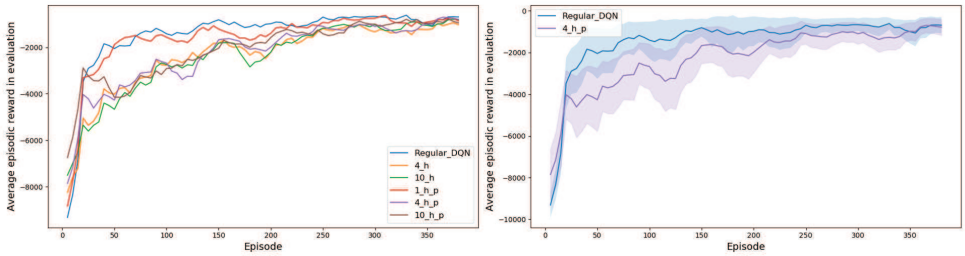


Fig. 3: Evaluating the agents against the grid scenario. The left figure presents the evaluations. The number describes the size of the bootstrap sample (number of heads), and the p whether a randomized prior has been incorporated. The right figure presents the same for two chosen agents, including the 95% confidence interval of the mean.

Figure 3 illustrates that while all agents learn, the agents that appear to have the sharpest learning rates are the regular DQN with and without prior function applied. However, as can be seen in the right plot in figure 3, there is some overlap in the confidence intervals of their means.

5.2 Scenario 2: Simulation of real traffic in Manhattan, New York

The results of evaluating the agents against the Manhattan scenario simulating real traffic can be found below, separately for each set of experiments, controlling each of the two intersections marked in figure 2 b. The agents' policies are evaluated every training episode, over two evaluation episodes, and averaged.

Traffic of low average speed Figure 4 presents the results of experimenting control of the intersection Waring-Woodhull, the top of the two intersections in figure 2 b, capturing a traffic profile of slower average speed.

As observable, the difference between the evaluation scores in relation to the confidence interval of the means, is mostly negligible, with the exception that the regular DQN achieves inferior scores prior to episode 20. However, these scores are still well within the confidence interval of the means of the other agent's, and thus cannot be considered significant. This behavior is attributed to the simplicity of the traffic profile, in relation to the amount of information accessible to the agent in this scenario. As mentioned in section 4.2, the observation space of the agents in this experiment is $84\,\mathrm{m}^2$. While the scenario can be viewed as complex (several interconnected intersections, whose behaviors directly influence each other's traffic), which can translate to the learning process being more difficulty, the volume of the traffic, including the average speed, is rather low, and thus the policy required is not complex.

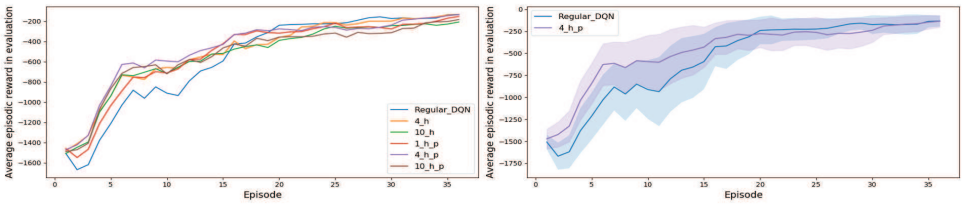


Fig. 4: Evaluating the different agents against the Waring-Woodhull intersection, simulating real traffic. The left figure presents the evaluation of the different agents. The number describes the size of the bootstrap sample (number of heads), and the p whether a randomized prior has been incorporated. The right figure presents the same results for a sample of the agents, including the 95% confidence interval of the mean.

Traffic of high average speed Figure 5 presents the results of experiments done controlling the second intersection, Eastchester-Waring. Eastchester-Waring enjoys both traffic of higher average speed, as well as higher traffic loads. No significant difference in the performance of the different agents is observed. This is further illustrated with the confidence interval of the means presented in figure 5 and their overlap.

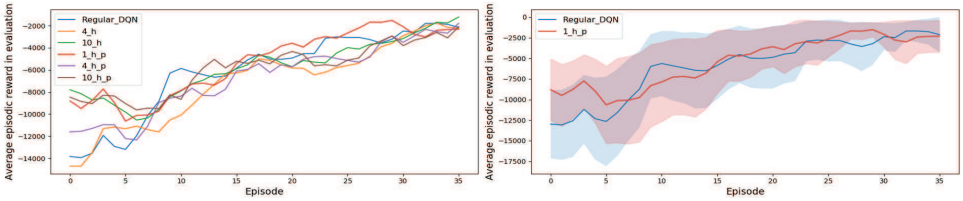


Fig. 5: Evaluating the different agents against the Eastchester-Waring intersection, simulating real traffic. The left figure presents the evaluation of the different agents. The number describes the size of the bootstrap sample (number of heads), and the p whether a randomized prior has been incorporated. The right figure presents the same results for a sample of the agents, including the 95% confidence interval of the mean.

5.3 Scenario 3: Simulation of custom traffic in Manhattan, New York

The results of evaluating the agents against the Manhattan scenario simulating random, heavy traffic can be found below. The agents' policies are evaluated every second training episodes, and averaged over two evaluation episodes.

Traffic of low average speed The results for the intersection Waring Woodhull are presented in figure 6. While the differences in the learning between most of the different agents appears negligible, all of them appear to outperform the regular DQN agent. However, as illustrated by the right plot in figure 6, the confidence intervals still have some overlap.

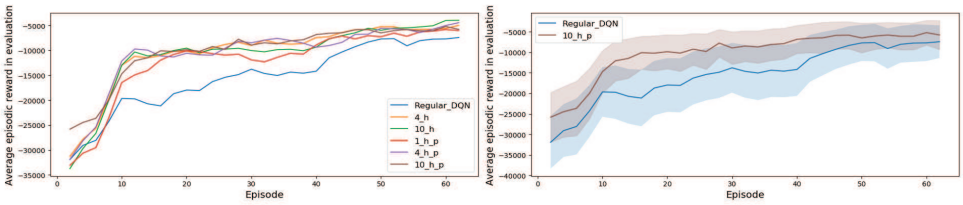


Fig. 6: Evaluating the different agents against the Waring-Woodhull intersection in the custom scenario. The left figure presents the evaluation of the different agents. The number describes the size of the bootstrap sample (number of heads), and the p whether a randomized prior has been incorporated. The right figure presents the same results for a sample of the agents, including the 95% confidence interval of the mean.

Traffic of high average speed The results of experimenting control over the intersection Eastchester Waring are presented in figure 7. No significant difference is observed between the evaluations of the different agents. This is illustrated strongly by the confidence intervals, and their overlap, presented in the right plot in figure 7.

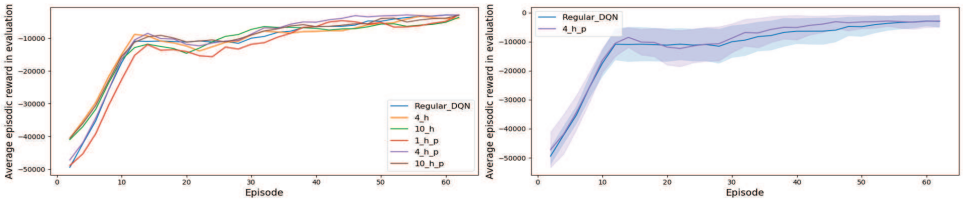


Fig. 7: Evaluating the different agents against the Eastchester-Waring intersection in the custom scenario. The left figure presents the evaluation of the different agents. The number describes the size of the bootstrap sample (number of heads), and the p whether a randomized prior has been incorporated. The right figure presents the same results for a sample of the agents, including the 95% confidence interval of the mean.

5.4 Analysis

Many different factors can influence the results obtained by the different agents, ranging from the reward function, the size of the observation space and its resolution, the state abstraction, the complexity of the traffic scenario and sheer stochasticity. The type and complexity of the traffic scenario, along with the observation space, are the main parameters that will be considered in this analysis.

Here, the complexity of the scenario, while not defined exactly, considers the volume and variety in the traffic and the number of surrounding interconnected intersections. In particular, the "grid" scenario can be viewed as having lower complexity, as it has a medium traffic volume, of low speeds and no neighbouring intersections. In the experiments done in the scenario simulating real traffic, both intersections could be viewed as having higher complexity: "Waring Woodhull", which is surrounded by a number of neighbouring intersections, with lower traffic volume and a slower speed; and Eastchester-Waring, which is surrounded by a similar number of neighbouring intersections, with somewhat higher traffic volume with significantly higher average speed. Last, the scenario simulating custom traffic could be viewed as having higher complexity as well, as it has similar parameters to the simulation of real traffic, with the exception of significantly increased traffic loads. In addition, it is worth mentioning that in the experiments done with the Waring Woodhull intersection the observation space was larger than in any of the other experiments, as specified in section 4.2.

It appears that when the complexity of the traffic is low along with a smaller observation space (the grid scenario), the simpler agents converge much faster, and more stably, to a highly evaluated policy. However, when the agent is given sufficient information of sufficient complexity (the Waring Woodhull intersection in both scenarios), deep exploration approaches are able to outperform the ϵ-greedy approach, by achieving faster learning. This may imply that advanced exploration approaches may play a significantly more critical role in much more sophisticated scenarios, such as an intersection balancing many lanes, experiencing many different traffic profiles, and provided a large observation space. The differences observed were generally small and within a 95% confidence interval of the means however, and as such, the strength of this implication is limited.

We note, though, that for the Eastchester-Waring intersection no improvements due to more sophisticated exploration were observed, even though the traffic patterns are more complex here than in the simple grid. The reason for this could be that as a result of the faster traffic and the smaller observation space, along with sufficient complexity in the road map and variety in the loads and directions of the traffic, the different costs and gains from the different exploration approaches balanced out.

6 Discussion

The results obtained suggest a link between the complexity of the scenario, the information accessible to the agent and the gain from deep exploration, as discussed in section 5.4. However, the advanced exploration approaches investigated are expected to have significant gain especially when utilized in environments with scarcity of significant positive rewards, and abundance of smaller negative rewards [11],[12]. As a consequence, the reward function used can almost directly dictate the gain from different exploration strategies. The reward function used was not designed for any specific exploration approach. It is therefore plausible that under a tailored reward function the gain from deep exploration would have been much

higher. This can be especially relevant in the case of reward functions that are known for promoting good policies, while hindering learning.

7 Conclusions and Future Work

This paper investigated the value of deep exploration in the setting of traffic light control, by comparing agents using different exploration approaches in three different traffic scenarios of rising complexity. Specifically, the state of the art approaches Bootstrapped DQN [12] and randomized prior functions [11] were compared to a baseline ϵ-greedy approach. This, to facilitate better deep RL in traffic light control, by identifying the value of evolved exploration approaches in this setting, such as higher sample efficiency or higher final policy score.

The results presented in this paper suggest a link between the complexity of the traffic scenario, the size of the observation space of the agent, and the gain from efficient exploration, achieved with Bootstrapped DQN and randomized prior functions, under a specific parameters configuration. Specifically, the more complex the scenario and the larger the observation space, the more significant the gain observed from efficient exploration.

The results presented leave the following open questions, however. What is the exact relation between the gain from efficient exploration and the complexity of the scenario? How sensitive is this relation to specific parameter configurations, and specifically under parameters optimized for the specific setting? Does this conclusion apply for other exploration approaches? These questions are left for future work.

Acknowledgements

We would like to thank the students in the research-project group, Pepijn Tersmette, Cian Jansen, Emanuel Kuhn & Chris van der Werf, for their ready assistance and availability for discussion, along with their contributions to the creation of the traffic scenarios and the implementation of the reward function used. Additionally, we would like to thank the TUDelft ILDM research group for their helpful comments and support. An additional thanks goes to the TUDelft for granting us access to the INSY cluster, which was a crucial computational resource used in this research.

This project had received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 758824 —INFLUENCE).

References

1. Bakker, B., Whiteson, S., Kester, L., Groen, F.C.: Traffic light control by multiagent reinforcement learning systems. In: Interactive Collaborative Information Systems, pp. 475–510. Springer (2010)
2. Barde, M.P., Barde, P.J.: What to use to express the variability of data: Standard deviation or standard error of mean? Perspectives in clinical research **3**(3), 113 (2012)

3. Ciosek, K., Vuong, Q., Loftin, R., Hofmann, K.: Better exploration with optimistic actor critic. In: Advances in Neural Information Processing Systems. pp. 1785–1796 (2019)
4. Coşkun, M., Baggag, A., Chawla, S.: Deep reinforcement learning for traffic light optimization. In: 2018 IEEE International Conference on Data Mining Workshops (ICDMW). pp. 564–571. IEEE (2018)
5. Efron, B., Tibshirani, R.J.: An introduction to the bootstrap. CRC press (1994)
6. Fortunato, M., Azar, M.G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., et al.: Noisy networks for exploration. arXiv preprint arXiv:1706.10295 (2017)
7. François-Lavet, V., Henderson, P., Islam, R., Bellemare, M.G., Pineau, J.: An introduction to deep reinforcement learning. arXiv preprint arXiv:1811.12560 (2018)
8. Hansen, J.: bootstrap_dqn. GitHub repository (2019)
9. Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., WieBner, E.: Microscopic traffic simulation using sumo. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC). pp. 2575–2582. IEEE (2018)
10. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
11. Osband, I., Aslanides, J., Cassirer, A.: Randomized prior functions for deep reinforcement learning. In: Advances in Neural Information Processing Systems. pp. 8617–8629 (2018)
12. Osband, I., Blundell, C., Pritzel, A., Van Roy, B.: Deep exploration via bootstrapped dqn. In: Advances in neural information processing systems. pp. 4026–4034 (2016)
13. Van der Pol, E., Oliehoek, F.A.: Coordinated deep reinforcement learners for traffic light control. Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016) (2016)
14. Rezzai, M., Dachry, W., Mouataouakkil, F., Medromi, H.: Reinforcement learning for traffic control system: Study of exploration methods using q-learning. International Research Journal of Engineering and Technology **4**(10), 1838–1848 (2017)
15. TO, H.R., Barker, M.M.: White paper european transport policy for 2010: time to decide (2001)
16. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Thirtieth AAAI conference on artificial intelligence (2016)
17. Watkins, C.J., Dayan, P.: Q-learning. Machine learning **8**(3-4), 279–292 (1992)
18. Wei, H., Zheng, G., Yao, H., Li, Z.: Intellilight: A reinforcement learning approach for intelligent traffic light control. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 2496–2505 (2018)