Data-Driven Query by Vocal Percussion

by

Alejandro Delgado Luezas

A thesis submitted for the degree of

Doctor of Philosophy

Department of Computer Science and Electronic Engineering
Queen Mary University of London
United Kingdom

April 2022

TO MY FAMILY AND CLOSEST FRIENDS

Abstract

The imitation of percussive sounds via the human voice is a natural and effective tool for communicating rhythmic ideas on the fly. Query by Vocal Percussion (QVP) is a subfield in Music Information Retrieval (MIR) that explores techniques to query percussive sounds using vocal imitations as input, usually plosive consonant sounds. In this way, fully automated QVP systems can help artists prototype drum patterns in a comfortable and quick way, smoothing the creative workflow as a result. This project explores the potential usefulness of recent data-driven neural network models in two of the most important tasks in QVP. Algorithms relative to Vocal Percussion Transcription (VPT) detect and classify vocal percussion sound events in a beatbox-like performance so to trigger individual drum samples. Algorithms relative to Drum Sample Retrieval by Vocalisation (DSRV) use input vocal imitations to pick appropriate drum samples from a sound library via timbral similarity. Our experiments with several kinds of data-driven deep neural networks suggest that these achieve better results in both VPT and DSRV compared to traditional data-informed approaches based on heuristic audio features. We also find that these networks, when paired with strong regularisation techniques, can still outperform data-informed approaches when data is scarce. Finally, we gather several insights relative to people's approach to vocal percussion and how user-based algorithms are essential to better model individual differences in vocalisation styles.

Acknowledgments

First of all, I would like to thank the most important person I have had around while working on this thesis: Prof. Mark Sandler. Mark, thanks a lot for your excellent supervision, wise pieces of advice, and constant dedication throughout these years. I feel your supervision has made me a far better scientist than I was when I joined the program and I can't be anything but grateful to you for this. I wish you all the best.

Secondly, I'd like to thank Dr. Charalampos Saitis for his outstanding supervisory role in this thesis. You received me with open arms as your second supervisee despite the topic being somewhat tangential to your field of expertise. I believe that truly honours you as an open-minded and devoted lecturer. My industry supervisors at Roli: Dr. Ning Xu, SKoT McDonald, Angus Hewlett, and Rhiannon McLaren. Thanks a lot for your help and for introducing me to the industry's vision of music research. Dr. Emmanouil Benetos, for his invaluable help during the project, Dr. Jason Hockman and Dr. Julian Hough for their careful and thorough VIVA examination, Prof. Simon Dixon and Álvaro Bort for their help with grant-related bureaucracy, my colleagues Vinod Subramanian, Emir Demirel, and Fred Bruford for their collaborative impact, and my colleagues at QMUL and Roli for being such wonderful people to hang out with.

Lastly, an enormous "thank you" to my family. Nothing of this would have been even remotely possible without you. You are my anchor, you are my reason, and I love you dearly. To the European Union for making this project possible and to the people in the MIP-Frontiers program, for helping and inspiring me during the thesis. To my closest friends, César, Diego, Martín, and Pablo for their support and fun times despite the distance, to Frank Grimes for teaching us that a man can triumph over adversity, to La Papa for being the guiding light through the uncertain paths of life, and to Celtic music for putting an awesome soundtrack to this project.

Thanks to each and every one of you. The ones I leave behind, I'll always remember you fondly. The ones I live with, can't wait for our next deed together.

Table of Contents

List of Figures

List of Tables

Acronyms

1D-CNN	1-Dimensional CNN.
AC-GAN	Auxiliary Classifier Generative Adversarial Network.
ADT	Automatic Drum Transcription.
AE	Autoencoder.
AIC	Akaike Information Criterion.
ANN	Artificial Neural Network.
AVP	Amateur Vocal Percussion.
BRNN	Bidirectional Recurrent Neural Networks.
BTX	beatboxset1.
CAE	Convolutional Autoencoder.
CD	Compact Disk.
CFS	Correlation-based Feature Selection.
CI	Confidence Interval.
CNN	Convolutional Neural Network.

CQT Constant-Q Transform.

CRNN Convolutional Recurrent Neural Network.

DAW Digital Audio Workstation.

DFT Discrete Fourier Transform.

DL Drum-type Labels.

DNN Deep Neural Network.

DSP Digital Signal Processing.

DSRV Drum Sample Retrieval by Vocalisation.

DT Decision Tree.

DWT Discrete Wavelet Transform.

ERB Equivalent Rectangular Bandwidth.

FC Fully-Connected.

FFT Fast Fourier Transform.

FSB Freesound Beatbox.

GCC-PHAT Generalized Cross-Correlation Phase Transform.

GRU Gated Recurrent Units.

HFC High-Frequency Content.

HMM Hidden Markov Model.

IPA International Phonetic Alphabet.

ISA Independent Subspace Analysis.

KL Kullback-Liebler.

KL Kullback-Liebler.

KNN K-Nearest Neighbours.

LDA Linear Discriminant Analysis.

LMER Linear Mixed-Effects Regression model.

LPC Linear Predictive Coefficients.

LR Logistic Regression.

LSTM Long Short-Term Memory.

LVT Live Vocalised Transcription.

MDV Mehrabi Drum Vocalisations.

MFCC Mel Frequency Cepstral Coefficients.

MIR Music Information Retrieval.

MKL Modified Kullback-Liebler.

MLLR Maximum Likelihood Linear Regression.

MLP Multi-Layer Perceptron.

MOD Musical Onset Detection.

MRR Mean Reciprocal Rank.

MSS Mantel Score Significance.

NMF Non-negative Matrix Factorization.

PCA Principal Component Analysis.

PCP Pitch Class Profile.

PSA Prior Subspace Analysis.

QE Query By Example.

QVI Query By Vocal Imitation.

QVP Query By Vocal Percussion.

ReLU Rectified Linear Unit.

RF Random Forest.

RMS Root Mean Square.

RNN Recurrent Neural Network.

SAE Stacked Auto-Encoder.

SDL Sound and Drum-type Labels.

SED Sound Event Detection.

SL Sound-type Labels.

SN Siamese Network.

SVM Support Vector Machine.

t-SNE t-distributed Stochastic Neighbor Embedding.

TN Triplet Network.

UA User-Agnostic.

UB User-Based.

VAE Variational Autoencoder.

VIS Vocal Imitation Set.

VIS-P Vocal Imitation Set Percussive.

VPT Vocal Percussion Transcription.

XGB Extreme Gradient Boost.

ZCR Zero Crossing Rate.

Chapter 1

Introduction

Music Information Retrieval (MIR) aims at extracting relevant information from music. It is a relatively new field in modern computer science and it is having an increasing impact on the music industry. Some of the main disciplines in MIR like chord recognition, music transcription, and source separation are receiving a growing amount of attention from independent musicians, as they constantly output applications that can make their creative workflow more pleasant. For example, automatic music transcription enables artists to learn and produce musical works more comfortably by transcribing instrumental lines in songs, while source separation is able to create karaoke tracks by extracting individual vocal tracks from songs.

Along these lines, many mechanical and sometimes laborious tasks can now be fully automated thanks to MIR, allowing artists to keep their focus on the creative aspects of their craft. Music producers and creators working on Digital Audio Workstations (DAW) sometimes have to deal with plugins with a high number of parameters, which usually interrupt their creative flow. To let artists avoid this and make their creative process more streamlined, plugins that act like shortcuts to achieve the desired result (e.g. Synthassist [1] for synthesising sounds based on an input vocal query) are available to those that may otherwise find it difficult to use some plugins in DAWs. The main motivation for this thesis is to investigate to what extent can vocal percussion be used as a shortcut to create drum patterns within DAWs.

Query By Example (QE) is a discipline that investigates how to quickly query a specific audio file from a sound library by providing another file that sounds similar to it, like a sketch recording. This search is usually fully audio content-based, not requiring any other additional kind of metadata for queries like labels or text descriptions. A special case of QE is that of Query By Vocal Imitation (QVI), in which the example audio file that is fed to the system contains a vocal imitation of the desired sound. The subject covered in this thesis, *Query By Vocal Percussion* (QVP), is a form of QVI in which percussive sounds are retrieved from percussive vocal imitations. These are vocal utterances that are usually articulated so as to communicate a rhythmic idea, usually by imitating the sound of percussive instruments like those featured in a drum set. QVP systems map these vocal percussion sounds to real drum samples from a sound library so as to create a realistic drum loop in seconds, making composers save time and effort prototyping rhythms even without actual music knowledge.

Despite the potential practicalities of these systems, they are seldom used today. This could be due to several reasons, including the limited commercial spread of the applications and the insufficient precision of their algorithms. The latter issue, which also influences the former, could be due to some important limitations of current *heuristic* algorithms when adapting to different application contexts. The simplicity of heuristic algorithms, based on engineered audio features and traditional machine learning systems, makes them ideal to approach problems with few samples and low data complexity. However, they can underperform when they are presented with several sources of variance like performances recorded with different microphones, on different locations, with different levels of background noise, and from different participants' voices with different ways of imitating drum sounds. In those situations, it is challenging to select a set of audio features that is supposed to work optimally for such a wide variety of input data. *Data-driven* algorithms like deep neural networks [2] could potentially offer better accuracies, as they are able to learn features directly from vocal percussion sounds' spectrograms from different contextual sources. For this reason among others, they are also consistently proven to be more effective than heuristic methods in music analysis [3] and audio event detection [4].

However, data-driven deep learning models have a significantly higher number of parameters than heuristic methods, which leads to a higher risk of data overfitting in limited-data scenarios. This means that neural networks are more likely to model training data so accurately that they lose extrapolation power and end up underperforming in the evaluation task. Fortunately, several strategies have been developed over the years to make data-driven algorithms less prone to overfitting in low-data user-based scenarios and they are now being used with more frequency in problems where training data is scarce [5].

Vocal percussion is divided into two main practices: beatbox and amateur vocal percussion. Vocal percussion sounds relative to *beatbox* are produced using several parts of the vocal tract, some of which are used in normal speech and some of which are not [6]. The articulation of these vocal percussion sounds is usually carried out in a relatively similar way, slightly modulating their loudness and timbre for expressive purposes. Also, it has its own universal set of basic techniques from which individual beatboxers base their own [7]. In contrast, *amateur vocal percussion* involves performers with little or no previous experience in beatbox, usually including most musicians and music producers. In this modality, the vast majority of vocal percussion sounds are speech-like and performers are much less consistent in articulation compared with beatboxing [6]. Also, as amateur performers do not follow vocal percussion techniques, they mostly decide to use their own particular set of vocal percussion sounds, which are based on their way of imitating drum sounds and usually differ from those of other amateur performers [8].

This thesis explores the relevance of data-driven algorithms and routines to the two main tasks in QVP. The first task, *Vocal Percussion Transcription* (VPT), aims at using vocal percussion sounds to trigger drum samples. The second task, *Drum Sample Retrieval by Vocalisation* (DSRV), aims at using vocal imitations of real drum samples to automatically retrieve these or similar-sounding samples from a sound library. These two tasks are complementary to one another in music production contexts: in order to produce or sketch a drum line using vocal imitations, producers would first select the drum samples they want to use (DSRV) and then vocalise a beatbox-like performance that will be transcribed to the final drum line (VPT). We take a closer look at these two tasks in the following sections.

1.1 Task 1: Vocal Percussion Transcription

Vocal Percussion Transcription (VPT) is a relatively old subfield in MIR that is concerned with the detection and classification of vocal percussion sound events so as to trigger drum samples, sitting just between monophonic music transcription and speech recognition. In this way, fully automated VPT systems help artists prototype drum patterns in a comfortable and quick way, smoothing the creative workflow as a result. The goal of VPT is to transcribe vocal percussion sound events into typical drum instrument classes, usually including kick drums, snare drums, and hi-hats.

VPT is a problem of correspondence between a sound and a label, i.e., a classification task. As such, the most relevant set of audio features to feed VPT algorithms would be the one that best separates vocal percussion sounds, independently of the timbral similarity between these sounds and the drum samples that they are triggering. The models and techniques used in VPT are shared across many other disciplines in MIR and sound event detection like musical instrument recognition, music transcription, query by vocal imitation, and anomalous sound detection. Likewise, vocal percussion datasets are also used in areas like music cognition and to study interpersonal differences in vocal imitation styles among others [9].

There are two main frameworks in VPT. *Match and adapt* tries to find the most similar spectrum to the query one in a database of rhythmic performances [10], while *onset-wise separation* divides the audio file in individual vocal percussion sounds and analyse them separately [11]. In this thesis, we focus exclusively on the latter (onset-wise separation), as it consistently outputs better performances [8, 12, 13]. This method is composed of two usually independent tasks: onset detection and classification. *Vocal percussion onset detection* deals with the prediction of the exact moments in the audio waveform where vocal percussion sounds start, whereas *vocal percussion classification* tries to assign each sound the correct associated drum instrument label (e.g. kick drums for p-like sounds). We will dedicate several experiments in this thesis to investigate the relevance of data-driven methods in both the onset detection and the classification tasks. This is done for several realistic VPT scenarios that often require distinct approaches to be effectively solved.

Vocal percussion onset detection is usually performed in an *user-agnostic* context; that is, disregarding information about individual participants when detecting vocal percussion sound onsets. This is done under the assumption that vocal percussion sounds, whether they belong to beatbox or amateur vocal percussion, have similar features in their onset region that allow algorithms to model them all together without losing prediction accuracy. Data-driven algorithms are the ones to benefit more from this method, as the more data with shared characteristics fed to the models the better their performance.

Beatbox sound classification is also typically approached in a user-agnostic fashion, as beatbox has a universal set of techniques that does not vary significantly among performers [7, 14]. In contrast, most recent works in amateur vocal percussion classification carry out the classification process in a *user-driven* fashion, as this is known to improve classification performance compared to user-agnostic approaches [15–17]. In a user-driven context, users show the classifier their particular way of vocalising drum instruments (training set) so that the algorithm can recognise those vocal percussion sounds in the future, usually within a beatbox-style improvisation (test set). Training sets are usually recorded either by reproducing a predictable beatbox-style phrase multiple times, which is called the *fixed phrase* strategy or by recording individual audio files containing same-class vocal percussion sounds, which is called the *isolated samples* strategy.

Independently of the recording methodology, user-driven training sets relative to amateur vocal percussion often contain less than one hundred sounds. This *data bottleneck* limits the amount of input audio features that classifiers can take for modelling so as to minimise their risk of overfitting. In consequence, amateur vocal percussion classifiers are in need of naturally informative input feature sets to guarantee robustness in prediction accuracy for all participants, irrespective of their stylistic idiosyncrasies and vocal percussion skills. As the informative power of a feature set depends largely on the task at hand, the exploration, and evaluation of feature sets would most likely have to pass through regularisation processes, potentially including data augmentation routines [18, 19], heuristic feature selection [20], and representation learning [21] among others.

VPT can also be carried out in two different regimes. In *offline* mode, the algorithm has access to the whole audio file containing the vocal percussion performance and can use all that information at once to detect the onsets of vocal percussion sound events and classify them. The system would output a transcription file afterward with the sound events and their timings. Conversely, in *online* mode, the algorithm only has access to a short analysis buffer that contains the most recent few milliseconds of the recorded audio stream. In this case, the system would have to detect, classify, and usually trigger the response very shortly after the sound event is recorded; for instance, it would trigger a snare drum sound almost at the same time as the performer vocalises the percussive sound event that is supposed to trigger it. This online procedure puts an important constraint on the system, forcing a trade-off between delay (length of the analysis buffer) and performance (detection and classification accuracy). In this sense, the longer the analysis buffer, the more information is available to the algorithm and the better the performance is expected; but also the more delay between the trigger and the response. This delay could be perceptually unpleasant if it exceeds a certain threshold that usually depends on the nature of the task at hand.

In this thesis, we wanted to see to what extent can data-driven algorithms improve the state of the art in VPT. We dedicated individual chapters to onset detection, offline classification, and online classification where we detailed experiments on data-driven VPT and how their results compare with earlier heuristic approaches. To the date of the beginning of the thesis, no data-driven VPT method was yet proposed in the literature.

1.2 Task 2: Drum Sample Retrieval by Vocalisation

Commercial drum sound libraries typically contain thousands of samples from different drum types and articulations. For this reason, they could be too large for an individual to explore and find, for example, a specific snare drum sample to use in a musical piece. To save search time, some pieces of software use perceptual attributes like “brightness” or “warmness” to filter the samples [22]. Drum Sample Retrieval by Vocalisation (DSRV) provides a complementary and potentially faster way of selecting desired drum samples [22, 23], allowing artists to carry on creating without having to deal with exhaustive manual searches.

In contrast with VPT, DSRV is a problem of correspondence between two sounds, i.e. similarity estimation, and the main objective is to find the set of audio features that best link them in relation with other distractor sounds. DSRV is firstly studied in [2], where authors implemented different convolutional autoencoder models to extract embeddings (learnt feature sets) that can predict listeners' drum-imitation similarity scores. DSRV was also implicitly studied in the past through QVI, as drum samples were often part of the evaluated datasets. Main insights from QVI studies suggest that humans are generally skilled when performing and recognising vocal imitations of generic sounds, with most results pointing towards the high retrieval accuracy, speed, and usability of QVI systems when compared to query-by-text ones. Recent implementations of QVI systems use *deep representation learning* techniques to estimate the similarity between source sounds and vocal imitations through metric learning, which significantly increases the retrieval accuracy of QVI systems [24].

DSRV can also be approached in user-agnostic and user-driven ways. The former, *user-agnostic*, would assume that the vocal imitations of drum sounds are similar enough among users that they can be analysed and modelled altogether. This assumption was proven to be correct to some extent [2], although other studies suggested that vocal imitation styles vary significantly among some users and that a *user-driven* approach could potentially improve accuracy in some scenarios [9, 25]. Apart from the user-agnostic and user-driven DSRV distinction, the problem can also be described in terms of the learning strategy that algorithms employ, more specifically the type of metric that they calculate sound similarity with. In this sense, algorithms using *heuristic metrics* first extract a set of features from drums and vocal imitations and compute the similarity between them using a fixed metric (e.g. Euclidean distance). By contrast, algorithms using *learnt metrics* aim at discovering the optimal metric that links drum sounds with their vocal imitations given a set of features. An interesting property that some end-to-end data-driven algorithms have is that they are capable of learning features and metrics at the same time (e.g. siamese neural networks).

In this thesis, as with VPT, we explore to what extent can data-driven DSRV algorithms improve the state of the art in the field. We dedicated a full chapter to data-driven DSRV detailing the methodology and results from three experiments, putting performances in context with earlier heuristic and data-driven approaches.

1.3 Outline of the thesis

Chapter 2 provides an extensive review of the thesis's background information. The chapter is divided into two sections. The first one outlines the *theoretical foundations* of the work, covering both heuristic and data-driven signal processing. The second section attempts to gather all relevant *past literature* on the topic of QVP and related fields.

Chapter 3 presents the datasets that are going to be used throughout the thesis for data-driven QVP, some of them already available in literature and others to be crafted.

Chapter 4 details the main routines that we followed to create and evaluate the performances of several heuristic and data-driven vocal percussion onset detection methods, generally directed to VPT. We describe these approaches and present the final results for both offline (non-real-time) and online (real-time) onset detection.

Chapter 5 discusses the methodology and results of experiments on vocal percussion sound classification in an online context. We first investigate which phoneme-to-instrument mappings were the most adequate to perform online amateur vocal percussion classification. Then, we evaluate the performance of several algorithms when classifying beatbox sounds and also the sounds derived from the previously-mentioned mappings for amateur vocal percussion.

Chapter 6 focuses on the experiments relative to vocal percussion classification in an offline context. We first introduce the different types of datasets available for offline vocal percussion classification and explore the performance of different heuristic and end-to-end data-driven algorithms. Then, we attempt to improve the efficiency, effectiveness, and generalisation abilities of user-based amateur vocal percussion via supervised embedding learning.

Chapter 7 details the experiments we carried out for DSRV. We first carry out a preliminary analysis of the main DSRV dataset to explore user differences in vocal imitation styles, discover audio features that these users imitate skilfully, and inform future data-driven DSRV systems on how to represent the audio data that their learning algorithms take as input. Then, we use the previous insights to explore the potential of user-agnosict data-driven techniques to learn useful features for DSRV.

Chapter 8 outlines the main conclusions of the thesis, summarising its main findings and discussing the challenges to overcome in order to advance the field of QVP.

References

[1] M. Cartwright and B. Pardo, "Synthassist: an audio synthesizer programmed with vocal imitation," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 741–742.

[2] A. Mehrabi, K. Choi, S. Dixon, and M. Sandler, "Similarity measures for vocal-based drum sample retrieval using deep convolutional auto-encoders," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* Ieee, 2018, pp. 356–360.

[3] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 927–939, 2016.

[4] I. McLoughlin, H. Zhang, Z. Xie, Y. Song, and W. Xiao, "Robust sound event classification using deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 540–552, 2015.

[5] G. Koch, R. Zemel, R. Salakhutdinov *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2. Lille, 2015.

[6] N. Patil, T. Greer, R. Blaylock, and S. S. Narayanan, "Comparison of basic beatboxing articulations between expert and novice artists using real-time magnetic resonance imag-

ing." in *Interspeech*, 2017, pp. 2277–2281.

[7] D. Stowell and M. D. Plumbley, "Characteristics of the beatboxing vocal style," *Dept. of Electronic Engineering, Queen Mary, University of London, Technical Report, Centre for Digital Music C4DMTR-08-01*, 2008.

[8] A. F. S. Ramires, "Automatic transcription of vocalized percussion," 2017.

[9] A. Mehrabi, S. Dixon, and M. Sandler, "Vocal imitation of percussion sounds: On the perceptual similarity between imitations and imitated sounds," *Plos one*, vol. 14, no. 7, p. e0219955, 2019.

[10] T. Nakano, J. Ogata, M. Goto, and Y. Hiraga, "A Drum Pattern Retrieval Method by Voice Percussion," in *5th International Conference on Music Information Retrieval (ISMIR), Barcelona, Spain*, 2004.

[11] A. Hazan, "Towards automatic transcription of expressive oral percussive performances," in *Proceedings of the 10th international conference on Intelligent user interfaces - IUI '05*. San Diego, California, USA: ACM Press, 2005.

[12] A. Kapur, M. Benning, and G. Tzanetakis, "Query-by-beat-boxing: Music retrieval for the dj," in *Proceedings of the International Conference on Music Information Retrieval*, 2004, pp. 170–177.

[13] E. Sinyor, C. M. Rebecca, D. Mcennis, and I. Fujinaga, "Beatbox classification using ace," in *Proceedings of the International Conference on Music Information Retrieval*. Citeseer, 2005.

[14] S. Evain, B. Lecouteux, D. Schwab, A. Contesse, A. Pinchaud, and N. H. Bernardoni, "Human beatbox sound recognition using an automatic speech recognition toolkit," *Biomedical Signal Processing and Control*, vol. 67, p. 102468, 2021.

[15] K. Hipke, M. Toomim, R. Fiebrink, and J. Fogarty, "Beatbox: End-user interactive definition and training of recognizers for percussive vocalizations," in *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, 2014, pp. 121–124.

[16] A. Ramires, R. Penha, and M. E. Davies, "User specific adaptation in automatic transcription of vocalised percussion," *arXiv preprint arXiv:1811.02406*, 2018.

[17] A. Delgado, S. McDonald, N. Xu, C. Saitis, and M. Sandler, "Learning models for query by vocal percussion: A comparative study," in *Proceedings of the International Computer*

Music Conference, 2021, accepted.

[18] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[19] L. Nanni, G. Maguolo, and M. Paci, "Data augmentation approaches for improving animal audio classification," *Ecological Informatics*, vol. 57, p. 101084, 2020.

[20] V. Kumar and S. Minz, "Feature selection: a literature review," *SmartCR*, vol. 4, no. 3, pp. 211–229, 2014.

[21] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[22] Y. Zhang, Y. Zhang, and Z. Duan, "Sound search by text description or vocal imitation?" *arXiv preprint arXiv:1907.08661*, 2019.

[23] G. Lemaitre and D. Rocchesso, "On the effectiveness of vocal imitations and verbal descriptions of sounds," *The Journal of the Acoustical Society of America*, vol. 135, no. 2, pp. 862–873, 2014.

[24] Y. Zhang, B. Pardo, and Z. Duan, "Siamese style convolutional neural networks for sound search by vocal imitation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 2, pp. 429–441, 2018.

[25] G. Lemaitre, O. Houix, F. Voisin, N. Misdariis, and P. Susini, "Vocal imitations of non-vocal sounds," *PloS one*, vol. 11, no. 12, p. e0168167, 2016.

Chapter 2

Background

In this second chapter, we provide an extensive review of the thesis's background information. The chapter is divided into two sections. The first one outlines the *theoretical foundations* of the work, covering both heuristic and data-driven signal processing. The second section attempts to gather all relevant *past literature* on the topic of QVP and related fields.

To write our theoretical summary, we have taken care of several attributes that we believe are of relevance and help to the reader. On the one hand, we tried to maximise *comprehensiveness* in such a way that even readers specialised in different but related fields of knowledge have the possibility of understanding it. On the other hand, while keeping an appropriate level of comprehensiveness, we also tried to cover important *low-level details* of the most important concepts (e.g. neural network dynamics). Lastly, as we acknowledge that we cannot cover all related algorithms in literature (audio features, machine learning algorithms, types of neural networks...), we tried to assign priority to those methodologies that are most relevant for our thesis, while sometimes briefly commenting on interesting approaches and referencing related review papers that cover these and other approaches in a more extensive way.

To write the literature review related to this thesis, we also took care of several desirable attributes regarding the reader. The first one is the conceptual *connectivity* to the theoretical summary, meaning that we ensured that most of the concepts that would arise when summarising the content of relevant papers have been already explained beforehand or are properly referenced with papers that explain them. We also tried to gather all literature in QVP to date and provide an *extensive review* of each of the most relevant papers, dedicating a full paragraph to each of them. Lastly, we provided a *distilled review* of several related fields (e.g. automatic drum transcription), covering the most important works in the fields and outlining the methodologies that are likely to inform this thesis's approaches.

2.1 Theory

In this section, we provide an extensive review of the theoretical foundations related to this thesis. As discussed in the chapter's introduction above, we divided this review into *heuristic signal processing* and *data-driven signal processing*. The former elaborates on topics like spectral processing, audio feature extraction, traditional machine learning algorithms, and similarity estimation. The latter, data-driven signal processing, includes a detailed overview of neural networks' building blocks, characteristics, architectures, and dynamics.

The main high-level purpose of this theoretical review is to allow the reader to familiarise himself/herself with the concepts that this thesis is built on. The reader also has the possibility of diving deeper into the details and functioning of these concepts through both the review's body text and the referenced literature.

2.1.1 Heuristic Digital Signal Processing

In order to dive into the details and techniques related to heuristic signal processing, we find it most appropriate to define the concept so as to get an idea of the types of algorithms that it covers.

A *signal* is defined as "a detectable physical quantity or impulse (such as a voltage, current, or magnetic field strength) by which messages or information can be transmitted" [1]. By this definition, images and audio recordings can also be referred to as "signals". The term *digital* is defined as "composed of data in the form of especially binary digits" [2]. In contrast with analogue signals, which are considered continuous (i.e., they have an assigned value for any arbitrary point in time), digital signals are discrete (i.e., they have an assigned value every t timesteps of fixed length). Hence, *Digital Signal Processing* (DSP) would be the field of knowledge dedicated to analysing, modifying, and synthesising digital signals.

The use of the word *heuristic* to describe certain traditional signal processing algorithms, although extensively used in literature, is still somewhat contentious regarding the precision of its meaning. The word "heuristic" means "of or relating to exploratory problem-solving techniques that utilize self-educating techniques (such as the evaluation of feedback) to improve performance" [3]. Adopting this, it could seem at first that practically every problem-solving strategy in computer science is heuristic by definition. Nevertheless, we believe there are two implicit distinctions within the term that help with concretising its meaning, one of them qualitative and the other quantitative.

The qualitative distinction concerns the *subject* that has agency over the "exploration of problem-solving techniques". In this way, we would describe an approach as heuristic if a *human* is in charge of the exploration of these problem-solving techniques and their evaluation. If a *machine* is in charge, the approach would not be described as heuristic. However, it could be argued that virtually all signal processing approaches comprise a mixture of human- and machine-driven decisions, so the concept would still be ill-defined in signal processing. However, we also see a relevant quantitative distinction within the term that helps to narrow down its meaning, and that is the *level of agency* of the subject. In this way, if the most important decisions when exploring problem-solving techniques are done by humans and/or the majority of algorithm developing time and evaluation is carried out by humans, the approach could be described as "more heuristic" than another approach where machines get to make the most crucial decisions that affect performance and/or take the most amount of working time (e.g. when training end-to-end neural networks).

Although here we use the term "heuristic" categorically, we believe that the ideal way of using it would be as a quantitative term. To illustrate this, below are a few examples of methodologies in descending order of heuristicness:

- The approach of a researcher who observes that the signals relative to harmonic sounds are more periodic than noisy sounds manually estimates signals' periodicity by measuring the length stability of the spaces between the zero-crossings of signal cycles, and manually sets a thresholding parameter relative to this periodicity in such a way that, if exceeded, the signal would be labelled as "harmonic" and, if not exceeded, the signal would be labelled as "noisy".
- The approach of the same researcher above but, instead of manually setting a manual threshold, feeds the value to a decision tree machine learning algorithm (see 2.1.1.3) and allows it to decide whether the signal is harmonic or noisy.
- The approach of the same researcher above but, instead of manually estimating the signal's periodicity and setting a manual threshold, calculates the CQT spectrogram (see 2.1.1.1) and feeds it to an end-to-end neural network (see 2.1.2.1) that decides whether the signal is harmonic or noisy.
- The approach of the same researcher above that feeds the raw waveform directly into an end-to-end neural network that decides whether the signal is harmonic or noisy.

In this work, we would consider the first two approaches as "heuristic" and the last two approaches as "non-heuristic", which we would also refer to as "data-driven" later on (see 2.1.2).

2.1.1.1 Audio Representations

An audio signal can be mathematically represented in several different ways, some more appropriate and interpretable than others depending on the application context. No matter which representation method we choose in DSP, they all come from digitally measuring a physical audio wave through an electronic device. A physical *audio wave* is created by the oscillatory movement of the air particles due to rapid changes of pressure in the environment. In essence, a digital *audio recording* is trying to approximate this continuous physical audio wave by measuring the air pressure in the environment at a very high rate, usually 44,100 pressure measurements per second. A scheme representing this is given in Figure 2.1.

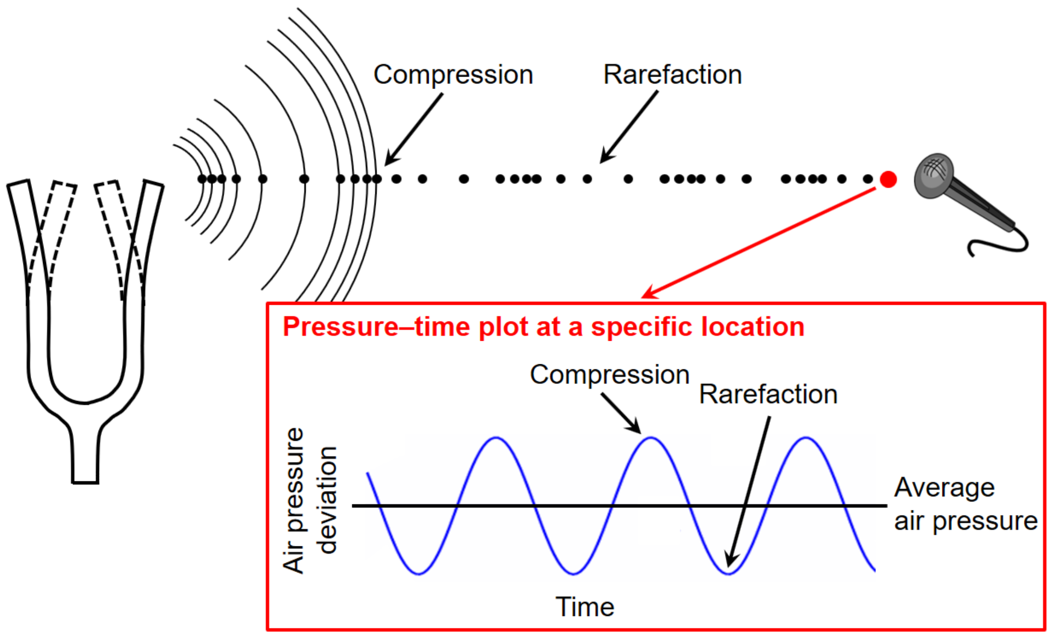


Figure 2.1: Schematic representation of the process of audio recording. Source: Muller, 2015 [4].

The digital representation of these measurements displayed in chronological order is called the *audio waveform*, which is illustrated in Figure 2.2. This waveform, if recorded digitally, is composed of a large number of amplitude measurements in a very short amount of time called *samples*. For instance, the audio content of a Compact Disk (CD) has 44,100 amplitude measurements per second, which is known as the *sample rate*. As we will see in later paragraphs, the higher the sample rate, the more exact the representation of the physical audio signal and, hence, the better the audio quality.

The raw waveform is the most basic representation of an audio signal, which can be later *transformed* to obtain other types of representation. There exist two main types of transforms for audio signals: the *invertible* or lossless transforms and the *non-invertible* or compressive transforms. By applying the invertible transforms, one can go back to the original representation by applying the inverse transform without any loss of information. With non-invertible transforms, in contrast, one cannot retrieve the original representation via an inverse transform, as the direct transform is already compressing the original signal.

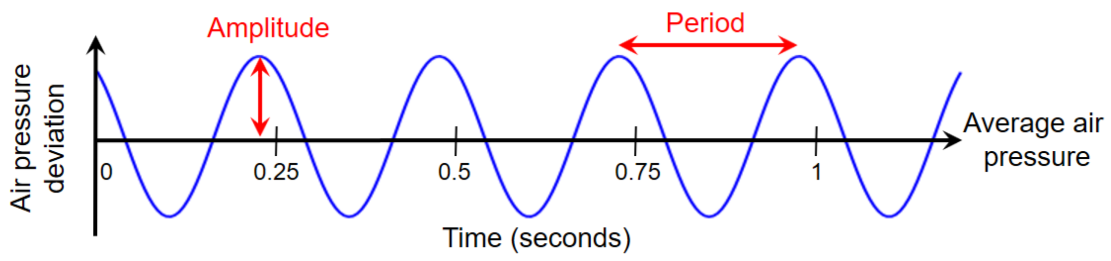


Figure 2.2: Diagram of a raw audio waveform. Source: Muller, 2015 [4].

Invertible Transforms

The most popular invertible transform of the raw audio waveform is the Discrete Fourier Transform (DFT), which expresses the audio waveform in terms of its frequency components. The output of this transform is known as the *frequency spectrum* of the audio signal. Mathematically, the DFT is defined as:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn} = \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(\frac{2\pi}{N}kn\right) - i \cdot \sin\left(\frac{2\pi}{N}kn\right)\right] \tag{2.1}$$

where $n \in [0, N-1]$ is the sample index in the audio waveform, x_n is the amplitude value of the audio waveform at sample n, $k \in [0, N-1]$ is the index of the frequency bin, and X_k is the Fourier coefficient relative to bin k. The last equivalence derives from Euler's formula, which is given by the formula:

$$e^{-\frac{i2\pi}{N}kn} = \cos\left(\frac{2\pi}{N}kn\right) - i \cdot \sin\left(\frac{2\pi}{N}kn\right) \tag{2.2}$$

Lastly, the inverse transform to obtain the samples of the audio waveform from the bins of the frequency spectrum is defined as:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{\frac{i2\pi}{N}kn} \tag{2.3}$$

Figure 2.3 provides an example of a DFT applied to an audio waveform. Among the relevant

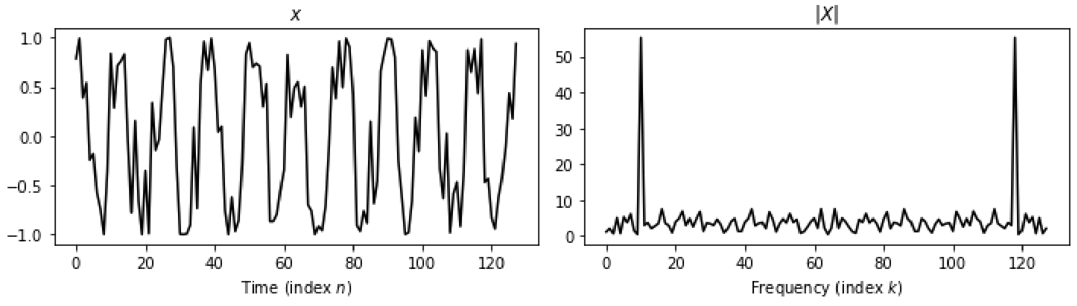


Figure 2.3: Illustration of an audio waveform and the magnitude of its DFT coefficients. Source: Muller, 2015 [4].

properties and characteristics of the DFT are the fact that it is horizontally symmetric with respect to the $k = \frac{N}{2}$ bin, it is time-invariant, and it encodes both the magnitude and phase information of all the sinusoidal signals that compose the original raw waveform. For audio analysis, the *magnitude spectrum*, which is given by the absolute value of the Fourier coefficients, is one of the most popular types of representations and also the basis to compute other non-invertible transforms. The *phase spectrum*, which is given by the angle of the complex sinusoids within the Fourier coefficients, is not as commonly used in heuristic DSP and it is usually reserved for concrete use cases like transient analysis and onset detection (see 2.2.3.1).

Another popular invertible transform related to the DFT is the Fast Fourier Transform [5] (FFT). This is a time-optimised convolution-based algorithm for calculating the DFT but orders of magnitude more efficient in terms of computation time than the original DFT algorithm. We use this transform extensively in this thesis to calculate spectral representations, especially for real-time processing, where optimising processing speed as much as possible is important so as to reduce response latency.

For a long audio file with millions of samples in its waveform, the usual approach to visualise the frequency content is to divide it into small frames of a few milliseconds in length, compute their associated spectrum, and lay these spectra vertically and in chronological order to see the variances of frequency content across time. This outputs an image type of representation called the *spectrogram* of an audio signal. This is illustrated in Figure 2.4.

Non-Invertible Transforms

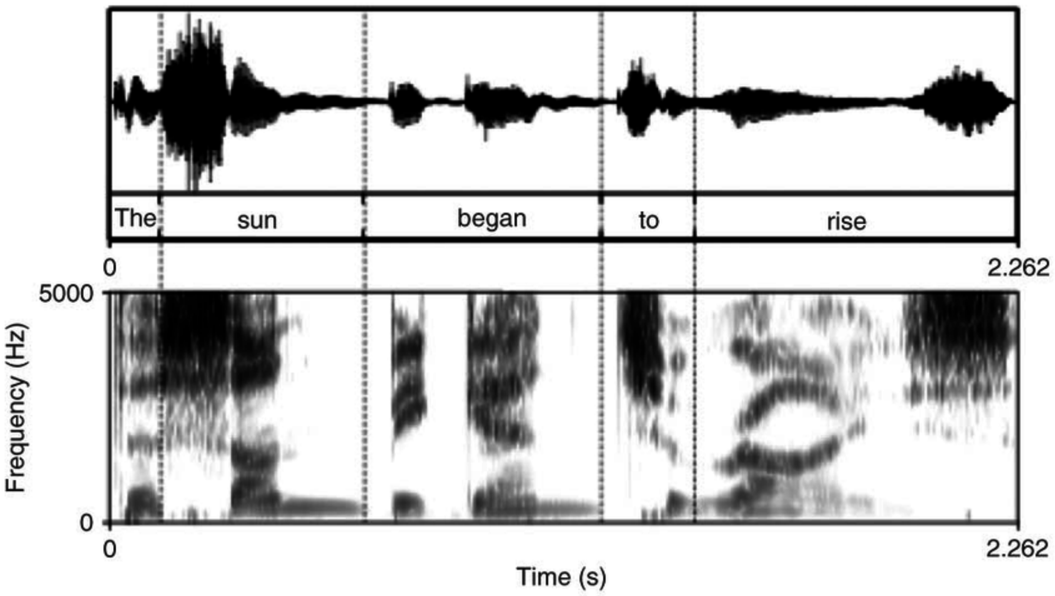


Figure 2.4: Waveform and spectrogram of the phrase "The sun began to rise". Darker regions correspond to more energy and the vertical dotted lines correspond to the estimated word boundaries. Source: Weber and Scharenborg, 2012 [6]

Popular non-invertible transforms are those based on the non-linear DFT (or FFT) compression using scaled frequency bands. In this way, one can choose an arbitrary number of bands and the DFT spectrogram would be compressed accordingly. For instance, if a total of 32 bands are selected, the information in the magnitude spectrum would be reduced to 32 bins whose value encodes the average spectral energy contained in each of these bands. This way of compressing the DFT is useful when the relatively high dimensionality of the DFT representation difficults analysis, which is common when feeding these representations to machine learning algorithms. The reason for this is that, the higher the dimensionality of input representations, the higher the amount of data required for the algorithm to generalise appropriately to unseen data points.

The most popular scales to compute frequency bands in audio signal processing literature are the Mel [7], Bark [8], and ERB [8] scales, whose frequency response is inspired by the way the human ear compresses auditory information to send to the brain. The frequency responses of each of these scales is compared against each other in Figure 2.5.

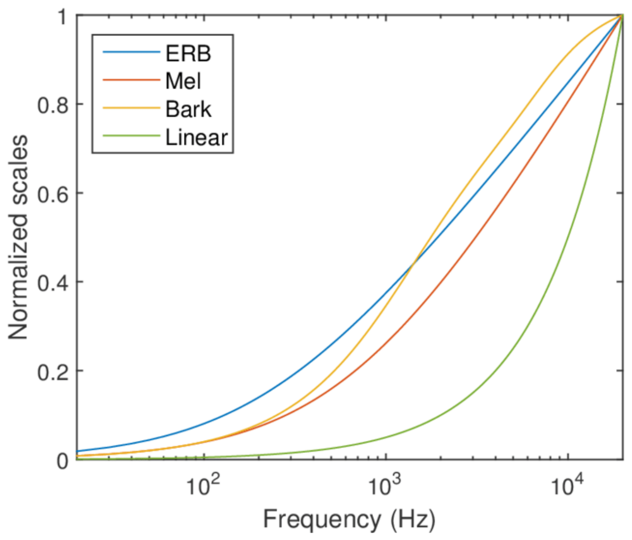


Figure 2.5: Normalised Linear, Mel, Bark, and ERB scales. Source: Rohr, 2015 [9].

Apart from the above, other types of non-invertible transforms that are widely used in MIR in particular include the Constant-Q Transform (CQT) and the Pitch Class Profile (PCP) series of algorithms, both of which are used to effectively encode the harmonic content in a compressed representation for analysis. As we are concerned with percussive signals, whose harmonic content is not as present as in other musical signals, we considered it best not to use these in the thesis.

2.1.1.2 Heuristic Audio Features

Given a certain task, DSP practitioners can further compress all the audio representations outlined above in a strategic way to solve the task in the most efficient way.

For instance, if the task is distinguishing between the audio signal of a bass guitar and the audio signal of a high whistle, the practitioner does not usually need the information from all the frequency bins in the DFT to base the decision upon. Instead, the practitioner can define and calculate a few informative heuristic audio features that are able to separate the classes by themselves. For instance, in our example, features that estimate the energy contained in the high-frequency bins appear to be relevant, as a bass guitar is expected to have a low amount of energy located there while the high whistle is likely to have almost all of its energy around that region in the spectrogram.

Heuristic features are usually composed by two elements: a base *audio descriptor* and an associated *statistical functional*. The former is what is used to compress the information inside the original audio representation for a single audio frame of a few milliseconds, while the latter is what is used to aggregate the values of the audio descriptors across frames (i.e. in time).

Below, we provide a list of some popular audio descriptors in DSP and MIR that we use in our work. A more complete overview of these and other descriptors is featured in [10].

- Spectral Energy: This is simply defined in DSP as the bin-wise squared value of the spectrum.
- Zero Crossing Rate (ZCR): This is defined as the number of times that the signal changes sign (i.e. crosses the zero line) divided by the length of the signal in samples.
- Mel Frequency Cepstral Coefficients (MFCC): These descriptors are widely-used in speech recognition and MIR to compress relevant frequency information in audio signals with the Mel scale as a basis. More information about the computation of these coefficients can be found in [11].
- Spectral Roll-Off: This is the frequency under which a certain percentage of the total spectral energy falls.
- Spectral Flux: This is defined as the difference in magnitude between the spectrum of the current frame and the one of the previous frame.
- Spectral Moments: These are derived by conceptualising the frequency spectrum as a statistical distribution whose values are the frequency bins and whose probabilities are the normalised amplitudes. The first, second, third, and fourth moments are known as the centroid, variance, skewness, and kurtosis respectively. The first moment, the spectral centroid, is frequently used in MIR like timbre analysis, whose approaches are generally relevant for this thesis.

- Fundamental Frequency: This is the frequency in a harmonic sound from which the overtones emerge, which is highly correlated with the psychoacoustical descriptor of *pitch*.

Some common statistical functionals that are applied to the values of these descriptors computed in a frame-wise manner are the *mean* value, the *standard deviation*, the *maximum* value, the *minimum* value among others. It is also a common practice in related fields like sound event detection to calculate these statistical descriptors not only from the original feature vectors but from the first and second derivative of these [12].

2.1.1.3 Machine Learning Models

Once an appropriate set of audio features is extracted from sounds, the *modelling* stage begins. Here, either the human researcher or an automated computer program (or both) takes charge of making sense of the information contained in the extracted feature set and uses it to solve the task at hand.

Modelling has been usually carried out via *machine learning* algorithms. These are automated computer programs that "learn" to solve a certain task usually by optimising a predefined objective function. The human researcher is normally in charge of designing the algorithm and setting its main learning hyperparameters according to external factors like the amount of data available for training.

Below, we outline some popular machine learning algorithms in the literature that we use in our work. A more complete overview of these and other machine learning algorithms can be found in [13].

- K-Nearest Neighbours (KNN): This is a simple algorithm that calculates the distance (usually Euclidean) between the feature vector of an unseen sample and those of training samples and classifies it using the majority vote of its k neighbours.

- Support Vector Machine (SVM): This algorithm works by estimating optimal decision boundaries in the feature space. The shape of these boundaries is given by SVM kernels, which could be of both linear and non-linear nature.

- Decision Tree (DT): This algorithm iteratively splits training data into ramifying subsets that land into one of the target classes. The splitting is done strategically, taking into account the most relevant features to separate classes.

- Random Forest (RF): This is a set of randomly-initialised trained DTs that classifies unseen samples by the majority vote of these DTs.

- Logistic Regression (LR): This algorithm is trained to predict the probability of an event, usually of binary nature, by fitting input data to a logit function via regression.

- Extreme Gradient Boost (XGB): This is an ensemble learning algorithm, i.e., a model made of several machine learning algorithms called base estimators, that usually makes more robust classification decisions when given a high amount of training data. It works under the assumption that the combination of enough weak and average classifiers can give rise to a single robust classifier.

2.1.2 Data-Driven Digital Signal Processing

As discussed earlier, heuristic DSP had two defining characteristics. On the one hand, the human researcher was the main subject carrying out the task. On the other hand, the human researcher had more agency on the task than the employed machines, usually making the most important decisions maybe also for a more prolonged time than machines.

Data-Driven DSP is a form of non-heuristic DSP that shifts the focus to the machine learning algorithms, being the main subject doing the task and making the most important decisions to achieve optimal results. Here the algorithms, instead of being informed by the human researcher's experience and intuitions on how to solve the task efficiently by observing data, are informed by the data itself. This means that the data-driven algorithms themselves are the ones to observe data and distil the relevant information that is necessary to solve the task.

The most popular and historically effective type of data-driven algorithms are Artificial Neural Networks (ANNs), in particular Deep Neural Networks (DNN). These algorithms are able to learn and extract meaningful compressed features from data that are generally better-performing

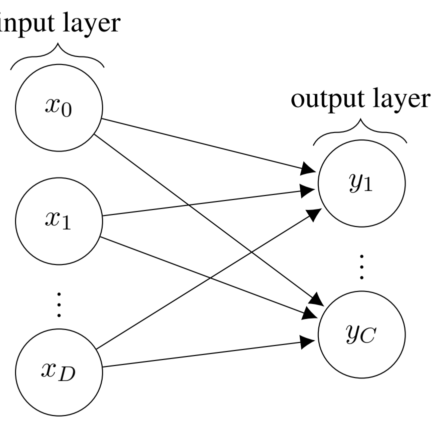


Figure 2.6: Network graph of a perceptron with D input neurons and C output neurons. Source: Stutz, 2017 [15]

than features extracted heuristically (see 2.1.1.2), and therefore perform better than heuristic approaches on a regular basis if certain prerequisites like the sufficient amount of training data are met.

Below, we provide a comprehensive summary of the building blocks of neural networks, their dynamics, and the types that have been most successful over the years and that we employ in our thesis.

2.1.2.1 Introduction to Deep Neural Networks

The historical predecessor of deep neural networks was the *perceptron* algorithm, originally invented in 1958 [14]. This network consisted of D input neurons, corresponding to the input features, and C output neurons, corresponding to the labels. The neurons belonging to the input layer are all connected to those belonging to the output layer while the neurons inside these individual layers are not connected to each other. This is known as a *Fully-Connected* (FC) scheme. As the algorithm only learns a mapping between input features and output labels without calculating any intermediate representations, the perceptron is still considered a heuristic, non-data-driven algorithm. An illustration of the algorithm's architecture is given in Figure 2.6.

Later advances in neural network research gave rise to the Multi-Layer Perceptron (MLP) [16], which was the first DNN that was capable of learning representations in a data-driven way.

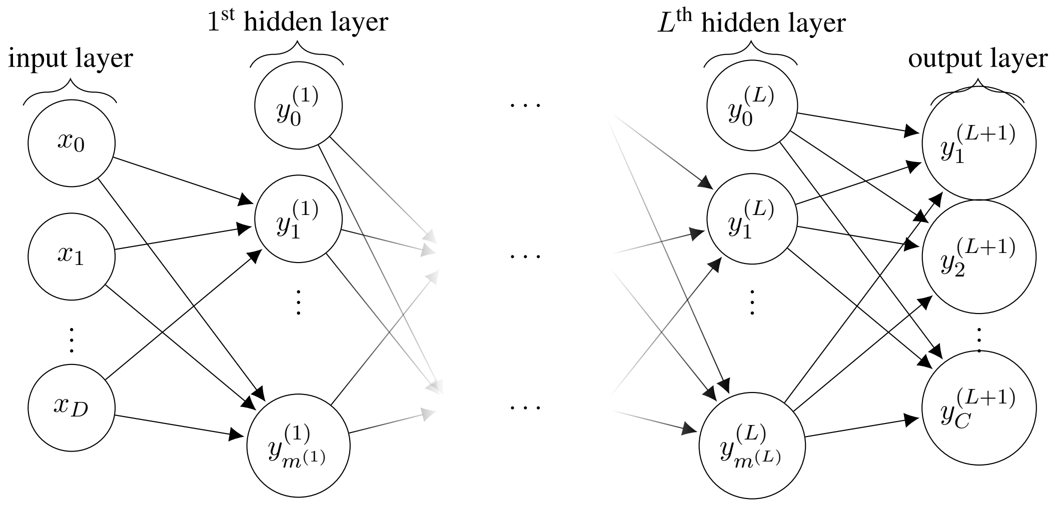


Figure 2.7: Network graph of an MLP with $(L + 1)$ layers. Source: Stutz, 2017 [15]

The MLP architecture and dynamics are exactly the same as the perceptron ones except that the MLP has a layer of neurons between the input and the output layers, called the *hidden layer*, that learns intermediate features and adds complexity to the resulting mapping function between inputs and outputs. Figure 2.7 schematises this type of network.

DNNs like MLPs are usually composed of the following basic elements:

- Input Layer: This is the first layer of the DNN, which stores the input features for the mapping process, each in one neuron.
- Output Layer: This is the last layer of the DNN, which stores the output layers for the mapping process, each in one neuron.
- Hidden Layers: These are the intermediate layers of the DNN, between the input and the output layers, and they store intermediate features that are learnt during the mapping process. The more hidden layers that a DNN has, the more complex the mapping will generally be.
- Weight Matrices: These are the "synapses" of the neural network, as they connect all the neurons in the layers sequentially via matrix multiplication. These matrices are multiplied by the values stored in the neurons of the previous layer so as to give the values that will

be stored in the next layer. The values in the weight matrices are updated in each learning iteration of the DNN so to optimise the mapping between input features and output labels.

- Biases: These are numerical constants that are added to each of the multiplications between layers and weight matrices, acting as an intercept point. They provide stability to the calculations, with their primary use being that of making sure that the calculated values for the next layer fall into the effective region of the activation function, which will be subsequently applied.
- Activation Function: These are generally non-linear functions (e.g. a sigmoid function) that are applied to the values calculated by multiplying the weight matrices by the previous network layer and adding the bias term. These functions add complexity to the mapping function that the networks calculate by introducing non-linearities.
- Objective Function: This is the function that the neural network is trying to optimise. In this way, the algorithm could, for example, learn a mapping between inputs and outputs that minimises a certain classification error metric (i.e. maximises classification accuracy).

These building blocks are common for all types of DNNs, whose structure and dynamics have been developed over the years giving rise to different types of DNN architectures and learning dynamics. In the next sections, we will give an overview of the learning strategies that DNNs can follow, we will explain their training dynamics with some common regularisation techniques, and detail the functioning and use cases of some of the most popular types of neural networks that we will be using in our thesis.

2.1.2.2 Learning Strategies

Given factors like the amount of data, the availability of labels, and the nature of the task at hand, different learning strategies for DNNs might be more appropriate. In this way, there are three main learning strategies that neural networks can follow: supervised learning, unsupervised learning, and reinforcement learning. These learning strategies also apply to other model types different than DNN like the machine learning algorithms in Section 2.1.1.3.

Supervised learning is the set of learning techniques that allow a certain machine learning algorithm to fit the objective function using labelled data. In this way, the labels (classes) that the data points have attached to them guide the algorithm in its goal of optimising the objective function and getting, for instance, optimal classification accuracies. The labels attached to data points could be of a categorical nature (e.g. kick drum vs. snare drum), where a *classification* algorithm is applied, or of quantitative nature (e.g. fundamental frequency), where a *regression* algorithm is used.

Conversely, the techniques that belong to *unsupervised learning* are the ones that do not have access to labelled data. Such strategies, like data clustering, make use of distance and/or similarity metrics to organise data points into subsets based on their resemblance to each other. These subsets can be seen as potential labels of some kind, although their relevance to the task at hand is not always guaranteed. For instance, an unsupervised clustering algorithm can end up segregating snare drum samples according to their brightness, duration, brand, recording setting... etc depending on the information contained in the feature vectors.

Lastly, *reinforcement learning* comprises a relatively new set of techniques that, although resemblant to supervised learning in a way, is fundamentally different to supervised and unsupervised learning. In reinforcement learning, algorithms learn to react to an environment with a single or few implicit goals on their own. For instance, a reinforcement learning algorithm can learn to play an arcade videogame in a trial-and-error way, recognising when the agent is failing and updating itself to avoid failure in future trials. This way, a major difference between reinforcement and supervised learning is that the former has implicit goals encoded in the environment it is reacting to while the latter's goals are made explicit via labelling.

2.1.2.3 Training Methodology

To describe the usual training methodology of neural networks, we will take the case of a supervised MLP classifier, although the process is practically identical to the cases of unsupervised and reinforcement learning.

Data Normalisation

It is good practice in deep learning to normalise input feature vectors before feeding them to the network. These routines ultimately help the algorithm with the process of learning, making it more smooth and less prone to unsuccessful learning.

Depending on the task, the type of network, and the type of weight initialisation routine (see below), one could apply different data normalisation techniques. The two most extended ones are normalisation between 0 and 1 and standardisation (or z-score), where the data would end up having an arithmetic mean of 0 and a standard deviation of 1. Normalisation between 0 and 1 is achieved by subtracting the minimum value from the feature vectors and dividing the result by the difference between the maximum and the minimum value, while standardisation is done by subtracting the mean value from the data and dividing the result by its standard deviation.

Hyperparameter Definition

In general, neural networks have associated with them a set of *hyperparameters* that dictate the dynamics of the learning process. These hyperparameters usually have to be set manually by the researcher based on experience and intuitions about the optimal way to solve the task.

Some of the most relevant hyperparameters for the MLP are the number of hidden layers, the number of neurons in each hidden layer, the number of iterations or epochs dedicated for training, batch size per iteration, and the learning rate, which regulates the speed at which the MLP will learn in each iteration. This hyperparameter is of special relevance as, depending on external factors like the amount of input data, the representativeness of this input data for all labels, and the complexity of the problem, one might set the learning rate accordingly so that the MLP is able to learn an appropriate mapping between input features and labels.

Weight Initialisation

Several initialisation routines for the weight matrices have been proposed over the years and, like data normalisation, their aim is to smooth the learning process ensuring that the algorithm optimises the objective function correctly.

Weight initialisation is carried out by generating random numbers under a specific probability distribution. Popular random weight initialisation techniques include the random normal initialisation (weights are initialised by sampling from a normal distribution), the random uniform initialisation, the Xavier initialisation [17], and the He initialisation [18]. The choice of one initialisation technique over another is highly impacted by the external setting like the nature of the distribution of input features and by algorithmic design elements like the chosen activation function [19].

Forward Propagation

The network dynamics start with a process known as forward propagation or forward pass. Here, the input neurons are multiplied by their respective weight matrices, added their respective bias term, and applied the activation function so as to give the values inside the neurons of the first hidden layer. Then, the neurons in this first hidden layer are multiplied by their respective weight matrices, added their respective bias term, and applied the activation function so as to give the values inside the neurons of the second hidden layer. And so on until we get the values in the output layer, which will be the predictions of the current network's iteration.

For instance, for a 1-hidden-layer neural network, the calculations relative to the forward pass can be expressed in the following manner:

$$\begin{gathered} Z_2 = X \cdot W_1 + B_1 \\ H_2 = \sigma\left(Z_2\right) \\ Z_3 = H_2 \cdot W_2 + B_2 \\ \hat{Y} = \sigma\left(Z_3\right) \end{gathered} \tag{2.4}$$

where X is the input feature vector, W_n are the weight matrices relative to the nth layer (1 = input layer, 2 = hidden layer, and 3 = output layer), Z_n are the non-activated neuron values, H_n are the activated neuron values (hidden states), σ is the activation function, and $\hat{Y}$ are the activated neuron values in the output layer (predictions).

Error calculation

After the forward pass, we obtain the prediction values for the current data point and iteration. In the first forward pass, these values generally will not be close to the target labels (e.g. 0 if the data point is relative to a snare drum and 1 if it is relative to a kick drum). Hence, in order to correct the values of the weight matrices so that the predictions is closer to the targets in the next forward pass, we first need to calculate the error between the target values Y and the predicted values $\hat{Y}$.

There are several error metrics available whose appropriateness depends largely on the task at hand. For instance, in our case (supervised classification), we could use the *cross-entropy* error function [20], which is given by the expression $(-Y \cdot log(\hat{Y}) - (1 - Y) \cdot log(1 - \hat{Y}))$.

Backward Propagation

Once the error is calculated, the backward pass is initialised. This is the most important process of the learning algorithms in DNNs, as it updates the values of the weight matrices through an algorithm called *backpropagation* [21]. Following this update, the network weights are tuned to maximise the possibility that the next forward pass would output prediction values that are closer to the target ones than the ones derived from the previous forward pass.

In our example above of a 1-hidden-layer neural network, the calculations relative to the backward pass through the backpropagation algorithm are expressed in the following way:

$$\begin{aligned} \partial W_1 &= \frac{\partial E}{\partial W_2} = \frac{\partial E}{\partial \hat{Y}} \cdot \frac{\partial \hat{Y}}{\partial Z_3} \cdot \frac{\partial Z_3}{\partial W_2} \\ \partial W_2 &= \frac{\partial E}{\partial W_1} = \frac{\partial E}{\partial \hat{Y}} \cdot \frac{\partial \hat{Y}}{\partial Z_3} \cdot \frac{\partial Z_3}{\partial H} \cdot \frac{\partial H}{\partial Z_2} \cdot \frac{\partial Z_2}{\partial W_1} \end{aligned} \tag{2.5}$$

where E is the error function, and ∂ denotes a partial derivative. If we calculate these partial derivatives using the expressions of the forward propagation phase (e.q. 2.4), we obtain the following simplified expressions:

$$
\begin{aligned}
\partial W_1 &= H^T \cdot \Big((\hat{Y} - Y) \cdot \sigma' \left(Z_3 \right) \Big) \\
\partial W_2 &= X^T \cdot \Big((\hat{Y} - Y) \cdot \sigma' \left(Z_3 \right) \cdot W_2^T \cdot \sigma' \left(Z_2 \right) \Big)
\end{aligned}
\tag{2.6}
$$

where the $'$ sign denotes the derivative of the function, the ^{T} sign means a transposing operation, and $\hat{Y}$ are the output predictions from the forward pass. These matrices can be then used to update the weight matrices in the following way:

$$
\begin{aligned}
W_1^{up} &= W_1 - \lambda \cdot \partial W_1 \\
W_2^{up} &= W_2 - \lambda \cdot \partial W_2
\end{aligned}
\tag{2.7}
$$

where λ is the learning rate hyperparameter and W_n^{up} are the updated weight matrices that would be used in the following forward pass.

In this way, all in all, the learning process of the neural network is a succession of forward propagations, error calculations, and backward propagations that ideally converge in an optimal solution for the task at hand.

Train, Validation, and Test Subsets

In machine learning, the dataset to be modelled is usually divided into the train and the test subsets. In this manner, the former would be used for training the algorithm and the latter would be used to evaluate its predictions in an “unseen” portion of the original dataset, from which the final results are derived.

In deep learning, however, it is best practice to include yet another subset for successfully training the algorithm: the validation (or development) set. The goal of this validation subset is to monitor the training performance and dynamics of the neural network using unseen data. In this way, if the network is performing well on the training set and poorly on the validation set, it might indicate issues with the generalisation abilities, usually due to *overfitting* on the training dataset. Some regularisation techniques like early stopping (below) take advantage of the predictions in the validation set to control the network's learning routines and stop training when the risk of overfitting is high enough.

Regularisation Techniques

The goal of regularisation in DNNs is to minimise the chances of overfitting during the training phase. Overfitting occurs when the mapping function is excessively complex and fits the training data excessively well so that the final algorithm is significantly less accurate when predicting unseen samples. In other words, if an algorithm overfits on its training dataset, its generalisation power decreases sharply. Although the chances of overfitting might have to do with the amount and quality of data as well, they mostly attend to factors related to the DNN design like, for instance, the number of trainable parameters (e.g. weights) that it has.

In this way, regularisation techniques are applied to minimise the chances of overfitting the training set. Some of the most popular regularisation strategies which we used in our thesis are:

- Data augmentation: This process consists in augmenting the size of the training subset by applying realistic transformations. For instance, a kick drum sample could be pitch-shifted within a range so that the transformed sample sounds realistic enough and incorporate it into the training set. Augmenting the size of the training set is vital in low-data regimes, as neural networks need very high amounts of data to generalise properly to unseen samples. Other common transformations for data augmentation in audio signal processing include time stretching and noise addition among others.

- Early stopping: This routine monitors the training and validation errors in the learning process and stops the training of the network if the validation error has not decreased for a certain number of training epochs. This simple heuristic is a very widely-adopted technique to avoid overfitting, specifically when the validation error starts rising.

- Dropout: This technique consists of the temporary turn-off of random network weights (by setting them to zero) during each of the forward propagation passes. In this way, the network weights are forced to be informative enough by themselves so that, if some of the remaining weights are set to zero, the network can still perform forward propagation passes without losing a significant amount of accuracy. When these weights are set to their usual value when evaluating the model, it is usually found that the models can achieve better accuracies and generalise better due to this restricted access to their own weights during training time.

- Batch normalisation: This technique consists in normalising the layers' inputs in the network via standardisation. This usually makes the networks converge faster to an optimal mapping and also makes the training process more stable (i.e. training and validation errors gradually decreasing at the same pace).

2.1.2.4 Other Types of Neural Networks

Apart from the MLP, which is arguably the simplest form of DNN, there exist several more families of DNNs that are widely used in deep learning literature, including the present thesis. In this last section of the theoretical review, we will introduce the main families of neural networks that we use in our thesis: Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Convolutional Recurrent Neural Networks (CRNNs), several types of Autoencoders (AEs), Siamese Networks (SNs), and Triplet Networks (TNs).

Convolutional Neural Network (CNN)

CNNs [22] are feed-forward DNNs composed of convolutional layers. The neurons in these layers compose a set of small local filter kernels to analyse the input, creating several feature maps as a result. An example of a two-dimensional convolutional operation is illustrated in Figure 2.8 A convolutional layer is usually followed by another convolutional layer, by a pooling layer, which subsamples these resulting feature maps following a certain set of rules, or by a fully-connected layer like the ones in the MLP. The main reason behind the use of CNNs for audio analysis is that they can effectively recognise spatial patterns in magnitude spectrograms that are informative to the classification task.

Recurrent Neural Network (RNN)

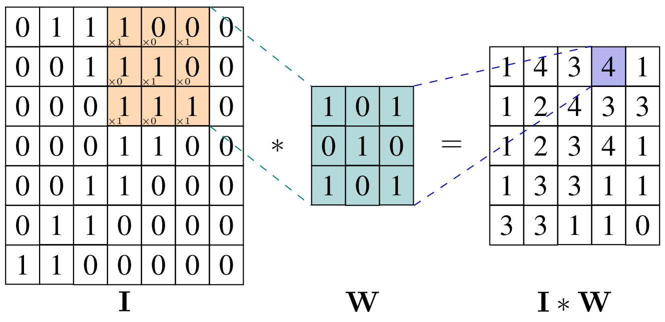


Figure 2.8: Illustration of a two-dimensional convolution operation on an input matrix I and a sliding kernel W with learnable weights.

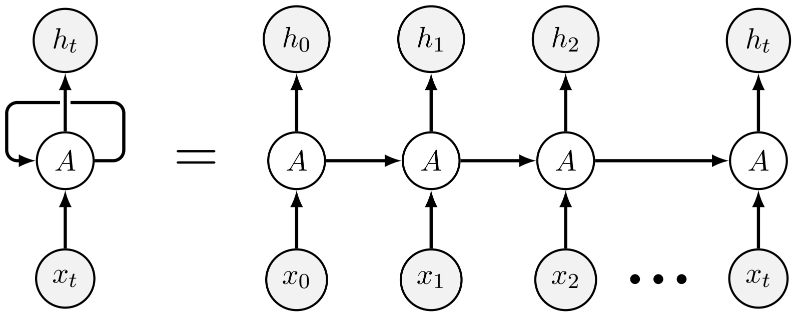


Figure 2.9: Diagram of a rolled (left) and unrolled (right) RNN, with x being the input feature vector, A being the recurrent cell, h being the hidden state, and t the time step.

The functioning of an RNN, in contrast with CNNs, is dependent on the information acquired from past inputs, allowing these networks to model complex data sequences in a non-linear way. An illustration of the recurrent structure in RNNs is given in Figure 2.9. When spectrograms are fed to RNNs, these networks can detect and model changes in time by contrasting observations from current frames with information from previous ones, which are especially useful in tasks like musical onset detection.

The training of RNNs is challenging due to inherent problems like the exploding/vanishing gradient, which states that big gradients that grow over training iterations can make the model unable to learn. In order to solve this problem and allow the networks to better handle dependencies in time, special memory cells for RNN have been proposed in recent years, one of them being the Long Short-Term Memory (LSTM) unit [23]. This memory cell introduced a cell state complementing the original hidden state of RNNs, which effectively stores information longer

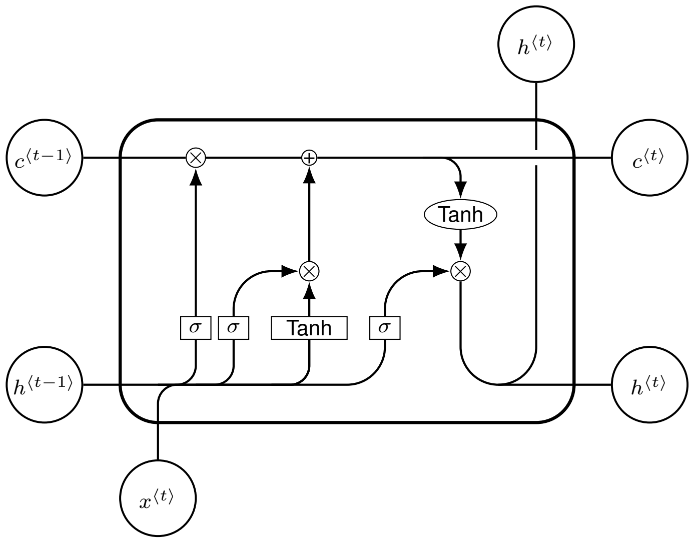


Figure 2.10: Diagram of an LSTM cell, with c being the cell state, h the hidden state, $Tanh$ the hyperbolic tangent activation function, σ the sigmoid activation function, and t the time step.

in time while keeping training gradients numerically stable. A diagram of an LSTM unit is provided in Figure 2.10.

Autoencoder (AE)

A deep autoencoder (AE) is a type of DNN that tries to learn embeddings by reconstructing (replicating) the values of an input feature vector while compressing its information by passing data through a bottleneck of lower dimensionality [24]. The bottleneck or latent space is also known as *embedding*, which we use extensively in this thesis and essentially means a compressed representation of an input one. AEs are, therefore, DNN-based dimensionality reduction algorithms, which we illustrate in Figure 2.11. Apart from the regular fully-connected AE, it has also been proposed variants of the model with convolutional and recurrent layers that are widely used for representation learning.

There have been several proposed variances of the original AE with fully-connected layers. Some of the most impactful ones include the Variational Autoencoder (VAE) [25], which is a generative model whose embeddings are sampled from a probability distribution (see Figure 2.12) and the Conditional Autoencoder [26], which takes label information to inform the model

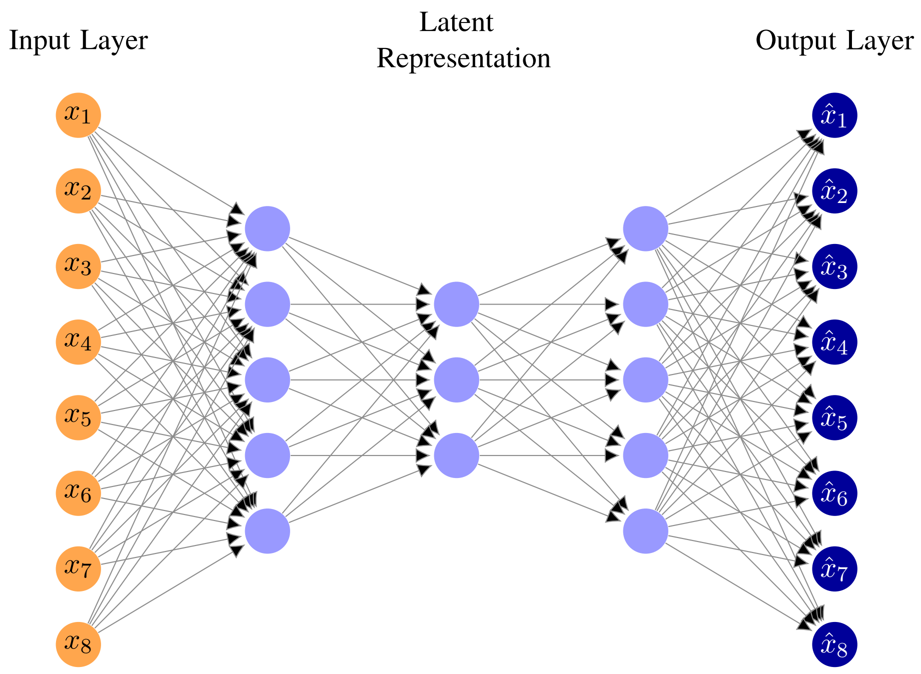


Figure 2.11: Illustration of an AE, with x_n being the input features relative to the nth index and $\hat{x}_n$ the predicted, reconstructed features relative to the nth index. The latent representation at the network's centre is the learnt, compressed embedding.

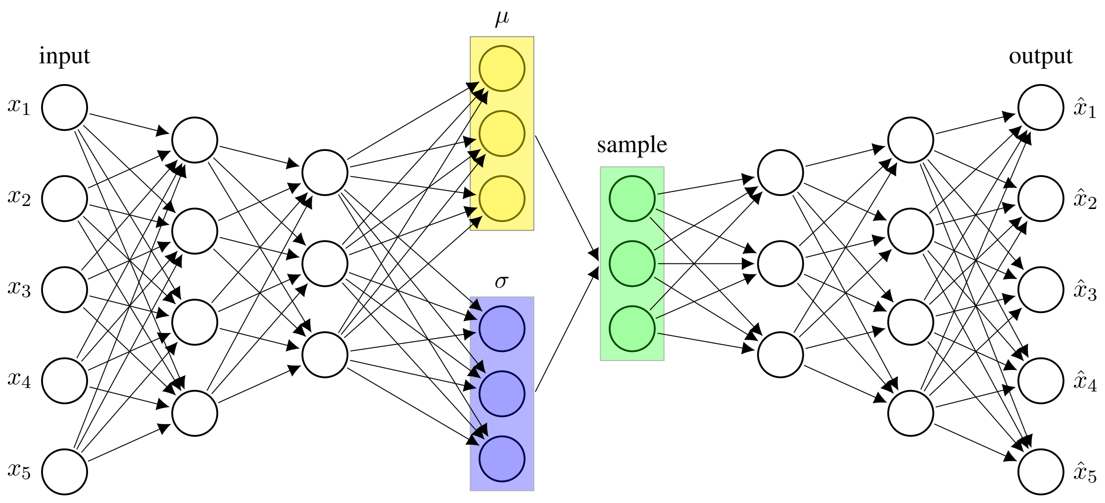


Figure 2.12: Illustration of a VAE, with x_n being the input features relative to the nth index and $\hat{x}_n$ the predicted, reconstructed features relative to the nth index, μ the vector containing the mean values of the probability distribution, and σ the vector containing its standard deviations.

while learning representations.

Siamese Network (SN)

SNs [27] are DNNs that learn features (embeddings) that maximise separability between samples from a different class and minimise separability between samples of the same class by optimising a special type of objective function called the *contrastive loss*. This is, therefore, a form of distance metric learning method and not a classifier per se. SNs are especially useful in one-shot and few-shot learning tasks, where there are one or just a few samples per class respectively [28].

Triplet Network (TN)

Similarly to SNs, TNs [29] are DNNs that use another special type of loss called the *triplet loss* to make learnt embeddings that maximise separability between classes via the *triplet loss*.

The triplet loss is a more sophisticated version of the contrastive loss and works by forming random data triplets with data samples: an anchor, a positive and a negative sample. The anchor and the positive samples are of the same class while the negative is not, and the goal of the algorithm is to minimise the distance (usually Euclidean) between the embeddings from the anchor and the positive samples while maximising the distance between the embeddings from the anchor and the negative samples.

2.2 Literature Review

In this second main section of the chapter, we provide a literature review of the most relevant past work for our thesis. We selected both the studies that we thought are most likely to inform our approaches in the thesis and also some of the most important works in the fields, even if they would end up having a limited impact on the development of our studies. We think that these studies, while not being used enough in our thesis, are still likely to have an impact on future directions of QVP, and thus are worth mentioning.

We dedicate a full section to the past literature related to vocal percussion transcription, another section to drum sample retrieval by vocalisation, and another one to related topics, which include musical onset detection, automatic drum transcription, query by vocal imitation, sound event detection, and phoneme recognition.

2.2.1 Vocal Percussion Transcription (VPT)

Vocal Percussion Transcription (VPT) is a subfield in MIR that aims at both the detection and the classification of vocal percussion sound events. These vocal percussion sounds generally occur in improvised musical performances (e.g. beatbox) and, given that the sounds are produced with the vocal apparatus, these performances are usually monophonic in nature, meaning that no two vocal percussion sound events can occur at the same time.

2.2.1.1 Methodology

As discussed in the introduction chapter (see 1.1), in this thesis we will pursue the onset-wise separation strategy when approaching VPT, as it is one of the most successful approaches not only in VPT, but also in similar fields like sound event detection [12]. This strategy consists of two tasks that are performed one after the other: sound onset detection and classification.

Sound *onset detection* algorithms usually work by (i) extracting input representations from the vocal percussion sound's waveform in a frame-wise manner, (ii) calculating an activation function (sometimes called novelty function in the literature) with the input representations as variables, (iii) postprocessing the resulting activation function (e.g. applying a low-pass filter to remove noise), and (iv) identifying the local maxima in the activation function through pick-picking functions. These input representations can be either engineered audio descriptors (heuristic onset detection) or a spectral frame like the FFT (data-driven onset detection). Similarly, the activation function could be calculated by combining the input audio descriptors' values (heuristic onset detection) or through the training of an end-to-end deep learning algorithm (data-driven onset detection). The most popular metric to evaluate onset detection tasks, and the one we will use in our thesis, is the F1-score [30].

After the onset of the vocal percussion sounds is predicted, the actual content of the sounds is assumed to be within the regions in between the onsets. These regions are extracted and analysed separately so that the classification process can take place. Sound *classification* algorithms usually work by (i) extracting input representations from the vocal percussion sound's waveform in a frame-wise manner from the onset time to an arbitrary point ahead, and (ii) training a machine learning classifier with such representations. These input representations could be descriptors aggregated over time using statistical functionals (heuristic classification) or a time-frequency spectral representation like the Mel spectrogram (data-driven classification). The most popular metric to evaluate classification tasks, and the one we will use in our thesis, is the accuracy [31].

2.2.1.2 Past Work

Until the starting date of the thesis, most (if not all) of the systems that used vocal percussion sounds to query drum sounds were mostly focused on beatboxing and they used heuristic feature extraction [32] and traditional machine learning methods [33] to train and evaluate classifiers. In the following paragraphs, we will detail the most relevant methodological aspects and conclusions from each of these studies, dedicating one paragraph to each of them.

Kapur et al. [34] recorded a dataset of 75 beatboxing sounds and studied the efficacy of different audio features and feature sets when classifying such sounds into a kick drum, snare drum, and hi-hat. The authors tried several time-domain features like the ZCR, the waveform energy via the Root Mean Square (RMS); some spectral features like MFCCs, centroid, roll-off, and flux; and some other features like Linear Predictive Coefficients (LPC) and wavelet coefficients. The zero-crossing rate achieved the best performance on its own (97.3% accuracy) as input to a neural network classifier. The authors also performed rhythm analysis via the Beat Histogram on drum loops using the Discrete Wavelet Transform (DTW) to measure tempo (main peak of DTW) and extract several heuristic features. Results suggested that the average tempo information is an important feature to distinguish between drum loops pertaining to different music genres. They output two systems from this work: the Bionic BeatBoxing Voice Processor and the Musescape tools, the former being a VPT system and the latter being a browsing device for drum and beatboxing loops.

Nakano et al. [35] recorded 200 utterances of vocal percussion patterns containing kick drum and snare drum vocal percussion sounds and built a retrieval system based on pattern matching. The system comprised several modules including an MFCC extractor, an acoustic model based on a phoneme-level Hidden Markov Model (HMM) that used an onomatopoeic pronunciation dictionary, and a Viterbi algorithm, and a cosine similarity calculator of Inter-Onset Interval vectors. The onomatopoeic dictionary consisted of several expressions that could be composed of consonants, vowels, choked sounds, syllabic nasals, and long vowels. They reported an accuracy of 91.0% for drum pattern retrieval using user-based acoustic models and the vocal percussion pronunciation dictionaries. This work was later used to develop a percussion instrument notation interface called Voice Drummer [36]. The system had three main modes: practice/adaptation, where the user records an imitation of one of eight possible practice drum patterns to also update; notation, where the user performs drum patterns vocally and are analysed and played by the system; and arrangement mode, where the user performs a vocal percussion accompaniment for a melodic backing track.

In a later study, Hazan [37] conducted experiments on VPT in a similar way as Kapur et al. using an energy-based onset detection algorithm and extracting several temporal and spectral features from vocal percussion sounds for classification. They explored the effectiveness of two kinds of classifiers: a KNN algorithm and a C4.5 decision tree. They also evaluated the latter by itself, with boosting, and with bagging. The author reports a cross-validation classification accuracy of 89.3% in his dataset (62 samples in the evaluation set) using C4.5 decision trees with bagging. They found especially helpful sound envelope descriptors like the attack duration and decay durations, time-domain descriptors like the ZCR, and spectral descriptors like the kurtosis (fourth spectral moment), and the second and fourth MFCC.

Sinyor et al. [38] were the first to explicitly include amateur vocal percussion sounds in the evaluation dataset, which were ensured to mimic beatboxing technique and sound qualitatively similar to the rest of the same-class sounds. When recording their dataset, they found that users generally imitated kick drums using the phoneme /p/, the closed hi-hat using /t/, and the open hi-hat using /s/, with the snare samples displaying more variance regarding their phonetic representation (/p/, /ps/, and /k/). The authors did not apply any onset detector algorithm like the study above but segmented the vocal percussion samples manually. They also applied feature selection routines using genetic algorithms so to reduce the dimensionality of the feature set, which showed a significant performance improvement for some classifiers. They reported an accuracy of 95.6% using an AdaBoost algorithm along with C4.5 decision trees in their dataset.

Stowell et al. [39] conducted a study on real-time beatbox sound separability between kick drum, snare drum, and hi-hat in a beatboxing context. They extracted several audio heuristic descriptors [32] to build feature vectors for each vocal percussion sound and used the Kullback-Liebler divergence as an inter-class distance measure. They observed that one could obtain higher classification accuracies by delaying the start of the analysis frame 23 milliseconds from the onset time, noticing that some of the separability sub-tasks (e.g. kick vs. snare) showed peak performance at significantly higher delays than others. They also tried feature stacking and feature selection with no significant improvement in separability scores. Lastly, they run a user study based on a MUSHRA type of test [40] where participants evaluated the unpleasantness of onset delays in drum performances in both isolated and musical contexts. They found that a delay of 12 to 35 ms was consistently described as having either excellent or good audio quality, while delays larger than 35 ms were usually described as having bad audio quality.

Hipke et al. [41] developed their own system called BeatBox that performed VPT in a user-based fashion, meaning that the algorithm was trained and evaluated on data coming from single users, usually of a few samples per drum class. They fed a total of 8 heuristic features (mean, standard deviation, minimum, and maximum of the spectral centroid and RMS) to a K-Nearest Neighbours (KNN) classifier with the Euclidean distance as the metric. The users of the BeatBox system are able to record the number of samples per instrument that they desire at any time and remove the badly-recorded ones so as to interactively update the algorithm. The users are also able to visualise the current classifier reliability by seeing to which class the algorithm assigns each of the recorded samples independently of the drum type that it was recorded for in the first place. They finally conducted an observational study where several participants trained and interacted with the model's graphical user interface. According to the authors, participants leaned on the classifier reliability feedback and the disabling of classification labels so as to simplify the problem if such labels were not going to be used in their prompts.

Picart et al. [42] attempted the classification of beatbox performances using a Hidden Markov Model (HMM). The authors recorded a dataset of 1,835 vocal percussion sounds from two beatboxers, which included kick drums, snare drums, rimshots, hi-hats, and cymbals. They used the spectral flux descriptor for onset detection and several audio features for classification like MFCCs (with first and second derivatives) and LPCs among others. They also used other performance metrics apart from the usual accuracy one including the number of substitutions, deletions, and insertions, which were later used to calculate the error rate. The authors reported an accuracy of 93% when classifying the five different classes of vocal percussion sounds.

Ramires [31] employed a similar strategy to Hipke et al. for amateur vocal percussion classification, as they also fed several heuristic features to a K-Nearest Neighbours (KNN) classifier operating in a user-based way. The author recorded a dataset of 841 vocal percussion sound from 20 different participants with different levels of skill using three different recording microphones (PC, AKG, and iPad) and three different types of imitated drum instruments (kick, snare, and hi-hat). This dataset is publicly available and we use it under the name of Live Vocalised Transcription (LVT) dataset (see 3.1.1), referencing the name of the system that was later derived from the study [43]. As his approach is user-based, they reported several transcription F1-scores (onset detection + classification) for different users. He also reported that the best onset detection algorithm for his particular dataset was one based on the High-Frequency Content (HFC) descriptor and that, for certain users, only one feature was sufficient when separating the classes to obtain a perfect classification score.

2.2.2 Drum Sample Retrieval by Vocalisation

Drum Sample Retrieval by Vocalisation (DSRV) is another subfield of MIR that is concerned with finding appropriate sets of audio features that best link drum samples with their vocal imitations. DSRV has been implicitly studied in the past through the field Query by Vocal Imitation (QVI), as drum samples were often part of the evaluation datasets. As seen in the introduction chapter (see 1.2), the primary application of DSRV in music production is to assist drum sample search engines.

2.2.2.1 Methodology

As in a standard similarity estimation task, DSRV works by (i) extracting input representations from the vocal percussion and drum sounds in a frame-wise manner from the onset time to an arbitrary point ahead, and (ii) estimating similarity between vocal percussion sounds and drum samples using a predefined similarity metric. This similarity metric could be of different nature (distance-based, correlation-based...) and it is generally fixed when estimating similarity, although recent methods are applying metric learning strategies that outperform fixed similarity metrics when enough drum-imitation pairs are provided for training. The most popular metric to evaluate classification tasks, and one of the metrics that we will use in our thesis, is the Mean Reciprocal Rank (MRR) [44].

2.2.2.2 Past Work

Although the field of QVI has extensive research on it, DSRV had only one published paper at the time of this thesis's beginning [45] and another one to be published in the following year [46]. Here we detail both of these papers and will leave the topic of QVI to Section 2.2.3.3 as a related field.

To the best of our knowledge, the first study that approached the topic of DSRV exclusively was that of Mehrabi et al. [45], where authors implemented a Convolutional Autoencoder (CAE) model to extract embeddings (learnt feature sets) that can predict human listeners' drum-imitation similarity scores. Therefore, instead of investigating the correspondence of variations in the acoustic space of drum sounds with those in the acoustic space of vocal imitations like in typical DSRV and QVI tasks, this study investigated the correspondence of variations in these two spaces and with those of several listeners' drum-imitation similarity ratings. This way, the evaluation metrics used throughout the thesis and some methodological aspects like the calculation of psychoacoustically-inspired input representations had a more marked perceptual nature than typical QVI tasks.

The authors used a dataset of thousands of vocal percussion sounds and generic vocal imitations as well as drum samples to train the convolutional autoencoder models. Then, they recorded a small dataset, which we refer to as the Mehrabi Drum Vocalisations (MDV) dataset, consisting of 30 drum sounds and their respective vocal imitations performed by 14 participants. They also performed a user study that gathered perceptual similarity ratings from 63 listeners between drum samples and imitations from the MDV dataset using a MUSHRA test. They used these similarity ratings as the ground truth for the experiments. They evaluated several network architectures using the MDV dataset and the perceptual similarity ratings as labels via two metrics: the Akaike Information Criterion (AIC) of a Linear Mixed-Effects Regression model (LMER) and the percentage of imitated sounds for which the fitted slope in a regular linear regression is significantly below 0 (under the 95% confidence interval), which they refer to as the "accuracy" metric.

Results from this study suggest that CAE models are better equipped than baselines like MFCCs to learn meaningful features from drum samples and vocal imitations even in a completely unsupervised way. The best scores were obtained using a CAE model with a wide kernel on the first and the last convolutional layers.

The same authors extended this work in a follow-up journal article elaborating on observations on the task and the main results from their previous experiment [46]. Their goal with this paper extension, in their own words, was to "establish whether musicians could effectively imitate percussion sounds such that listeners would consider the imitations more similar to the imitated sound than to other same–category sounds." They derived the following list of main observations, some of which we take into account for our own thesis decisions in DSRV:

- It was difficult for some listeners to reproduce their own similarity scores for certain drum-imitation pairs.
- Imitators tended to focus on the sounds' envelope when imitating same-instrument sounds.
- Participants found both the imitation and listening tasks significantly difficult, probably due to the fact that some of the same–category sounds were very similar.

- There was considerable confusion between sounds when the imitators adopted similar vocal techniques to imitate different sounds.
- 12 out of 63 listeners could not reproduce their own results to a reasonable enough degree so to include them for evaluation.
- There was a moderate-to-strong agreement amongst the ratings for the listeners that could reproduce their results well enough.
- 16 out of 30 drum sounds were overall considered as most similar to their original imitations.
- Imitators used different techniques to imitate the same sounds and, thus, DSRV systems may benefit from user-based approaches.

2.2.3 Related Fields

In this last section, we provide a review of the fields that we consider relevant for VPT and DSRV. We have chosen these disciplines attending to the simliarities with QVP mostly in terms of the type of data that they work with, but we also in terms of tooling and training/evaluation methodology. With this criteria in mind, we chose fields of Musical Onset Detection (MOD), Automatic Drum Transcription (ADT), Query by Vocal Imitation (QVI), Sound Event Detection (SED), and Phoneme Recognition.

2.2.3.1 Musical Onset Detection

Musical Onset Detection (MOD) algorithms attempt at identifying the onset of musical sounds in a certain audio recording and their methodologies can be of a heuristic and data-driven nature. The usual concepts and methodological aspects to carry out the task of MOD are explained in Section 2.2.1.1.

The first methods to approach MOD were based on the extraction heuristic audio descriptors and thresholding routines to predict onsets in case the value of the threshold is exceeded for a certain descriptor. Below, we list some of the most relevant audio descriptors and features that have been traditionally used for heuristic onset detection. A more complete review of these is given in [47].

- Spectral Energy: the amount of spectral energy that is contained on the whole spectrum or in specific energy bands. [48]
- High-Frequency Content (HFC): the summation of the bin magnitudes of the spectrum multiplied by their own bin indices. [49]
- Spectral Flux: is a frame-wise measure of how quickly the magnitude in each frequency bin of the spectrum changes over time. The sharper the change in a particular region, the more probable a percussive onset is occurring there. [47]
- Kullback-Liebler divergence (KL): is a probabilistic measure of spectral distance between the current and past frame [50]
- Modified Kullback-Liebler divergence (MKL): is a correction of the original KL divergence measure that accounts exclusively for positive amplitude changes in frequency bins and removes the original weighting that was done using the current frame's norm. [51]
- Spectral Phase: instead of using the energy of frequency bins in the magnitude spectrum, the *phase* [52] spectrum can also be used to detect onsets, in particular by measuring the amount of phase instability in the signal. In this way, high phase instability would correlate with the transient part of the sound while low phase instability would correlate with its steady-state part.
- Complex Domain: combines the above phase information with the one contained in the magnitude spectrum by detecting onsets directly in the complex domain. [53]

An important contribution to heuristic MOD is the *SuperFlux* method introduced by Böck et al. [54]. This is an enhanced version of the spectral flux MOD algorithm that reduces the number of false positives due to the vibrato (frequency modulation) and tremolo (amplitude modulation) of musical instruments, which can sometimes confuse onset detectors. They achieved this applying a trajectory-tracking algorithm with a maximum-filtered version of the input spectrogram. Although the superflux holds state-of-the-art in heuristic onset detection, it is not as clear if it would replicate this for vocal percussion sounds, as these usually do not display any vibratto nor tremolo.

Later on, new data-driven MOD methods based on end-to-end artificial neural networks were being proposed instead. The use of Bidirectional Recurrent Neural Networks (BRNN) [55] with Long-Short Term Memory (LSTM) units [56] for MOD was first proposed by Eyben et al. in [57]. The activations that these networks produced were computed using the log Mel spectrogram of frames as input calculated with a hop size of 10 ms. They fed two Mel spectrograms to the BRNN, one of them computed with a frame size 23.2 ms and another one with a frame size of 46.4 ms, allowing the network to have access to two different frequency and time resolutions that help with the onset estimation of multiple musical instruments. BRNNs consistently achieved better results than heuristic onset detectors in several datasets including pitched percussive instruments, pitched non-percussive instruments, non-pitched percussive instruments, and complex music mixes.

The idea of Eyben et al. above was later adapted to real-time processing by Böck et al. in [58], some of which were authors of the previous paper. They changed its architecture to a Recurrent Neural Network (RNN) [59] without LSTM memory units that took three types of Bark-filtered spectrogram frames: one of them with frame size 23.2 ms, another one with frame size of 46.4 ms, and an additional one with frame size of 92.8 ms. They also included the derivatives of these Bark bins stacked in the final feature vector. As it was directed to real-time regimes, which restricts the model to causal inference (no access to information after the current frame), their RNN did not achieve state-of-the-art results in MOD when compared with [57], although it came close to this result and also outperformed all the heuristic MOD methods evaluated.

In a later study, some of the same authors of the previous work compared the performance under an online (real-time) regime of their real-time RNN MOD system with those from several MOD algorithms based on heuristic descriptors [30]. The RNN in [58], apart from proving to be more accurate than the rest of the heuristic methods, it also demonstrated its invariance to loudness level differences of input data. In this way, while heuristc MOD methods lost accuracy as the loudness level of input data dropped, the RNN's performance remained stable, highlighting this practical property of data-driven onset detectors.

Convolutional Neural Networks (CNN) [60] would be the ones to later inspire a CNN MOD algorithm [61] that further improved the state of the art in musical onset detection. CNNs were an especially popular algorithm in computer vision that achieved high accuracies when detecting edges in images. The main idea behind their use for MOD is that CNNs can effectively detect edges in the magnitude spectrograms as well, which usually correspond to audio onsets. They fed as input representations three Mel spectrograms of 80 bands by 15 time steps computed with a hop size of 10 ms and with frame sizes of 23.2, 46.4 ms, and 92.8 ms respectively. The authors also implemented dropout-based regularisation, annotation fuzziness, and ReLU activation functions in the CNN, which all demonstrated to improve its performance. It was also discovered that the CNN, like spectral flux-based MOD techniques, computes spectral differences over time and uses them to estimate the likelihood of an onset happening in a certain time region. CNNs constitute the current state of the art in MOD.

Concerning MOD applied to vocal percussion sounds specifically, Hazan [37] used an energy-based onset detection algorithm, Kim et al. [62] used the RNN onset detection method in [58], and Ramires et al. [31] used an onset detection algorithm based on the High-Frequency Content (HFC) in their respective studies. The latter HFC algorithm detected 96.7% of all vocal percussion events in the LVT dataset without any false positives. Also, Picart et al. [42] studied the performance of several heuristic onset detection methods on beatbox performance segmentation, with the spectral flux method achieving the highest F1-score (0.922).

2.2.3.2 Automatic Drum Transcription

Automatic Drum Transcription (ADT) is a relatively old discipline in MIR that attempts to both predict the onsets of drum sounds in a given audio recording and classify them into the correct instrument class. Therefore, given the methodological resemblance with VPT, we believe that the findings that have been gathered in the past in ADT could inform the design of VPT algorithms to a certain extent. Below, we provide a selection of the studies that we considered most relevant for our work. A detailed review of ADT can be found in [63].

A pioneering study in heuristic ADT was included in the thesis of Schloss [64], where he studied audio features relative to the transients of percussive sounds and developed a methodology for pitch estimation on these types of samples. Gouyon et al. [65] proposed the Zero-Crossing Rate (ZCR) audio descriptor to classify kick and snare drums. The authors found the decay part of the sounds especially discriminative between these two types of drum sounds. Fitzgerald et al. conducted two subsequent studies [66] [67] in which they applied Independent Subspace Analysis (ISA) and Prior Subspace Analysis (PSA) to drum sound classification respectively. They found that prior knowledge about audio sources significantly improved results when compared to the ISA approach.

Later on, Van Steelant et al. [68] applied Support Vector Machines (SVM) to the classification of kick and snare drums. They used spectral and temporal descriptors to compute input feature vectors, including the energy of three frequency bands via the Root Mean Square (RMS), the ZCR, and spectral moments. They also found out that linear SVM kernels performed similarly to Gaussian kernels, saving a significant amount of computation time. Paulus and Virtanen [69] proposed an approach based on Non-negative Matrix Factorization (NMF) that achieved better classification results than those relative to the PSA and the SVM approaches.

Paulus and Klapuri [70] tried an approach based on Hidden Markov Models (HMMs) to transcribe drum sounds in polyphonic music. They preprocessed the signals via sinusoidal modelling [71] to filter out harmonic components, extracted features relative to the signals' MFCCs, ran Linear Discriminant Analysis (LDA) to reduce the dimensionality of the feature vectors, and fed these vectors to the HMM. They also post-processed data via Maximum Likelihood Linear Regression (MLLR) and the Viterbi algorithm. Then, Kaliakatsos-Papakostas et al. [72] used amplifiers and band-pass filters to transcribe drums in a real-time regime. In this way, if the amplitude of the filtered signals crossed a certain heuristically-determined threshold, the relative drum sound is predicted.

One of the first studies proposing neural networks as a viable data-driven approach to ADT was that of Gajhede et al. [73], who applied CNNs to the classification of a dataset of around 3,000 isolated samples pertaining to kick drum, snare drum, and hi-hat instruments. They used Mel spectrograms as input and got the best results when using batch normalisation for regularisation. Then, Southall et al. [74] proposed a Bidirectional RNN (BRNN) for the transcription of the kick drum, snare drum, and hi-hat, achieving state-of-the-art results in the IDMT-SMT-Drums and ENST minus one datasets, widely used in ADT. They also created an open-source web framework for ADT called ADTweb and a toolbox called ADTLib [75].

Vogl et al. [76] explored the use of several RNN models to predict kick drum, snare drum, and hi-hat as well. They obtained the best results when using a delay of 25 ms when training, similarly to the 23 ms delay in [77] for real-time beatbox transcription. In a later study [78], the same authors evaluated their RNN model with delayed prediction against past approaches and demonstrated its better performance in polyphonic ADT. In a subsequent study, Vogl et al. [79] approached the task of multi-instrumental ADT, which included classes relative to other percussive instruments usually featured in drum sets (cymbals, tambourines, cowbells...) and also ramified some of the drum labels used in the past (e.g. opened and closed hi-hat as two separate classes). They evaluated several models against the earlier one in [76] and demonstrated that their proposed Convolutional RNN (CRNN) model obtained the best results. This CRNN approach is able to take the surrounding context of the current frame into account to inform predictions.

Choi and Cho [80] developed a novel unsupervised strategy for ADT based on a CRNN model that takes advantage of unlabelled data for training, which is considered an important bottleneck in ADT [63]. Their work was followed-up by Wang et al. [81], who implemented a semi-supervised few-shot learning method that is also able to generalise predictions to unseen drum instrument types. Lastly, the work of Ishizuka et al. [82] further advanced the state of the art of supervised ADT by proposing an encoder-decoder architecture with a self-attention mechanism.

2.2.3.3 Query by Vocal Imitation

Research in Query by Vocal Imitation (QVI) has been very active over the last decade and it was mainly characterised by investigations on (i) the nature of vocal imitations and the acoustic relationship with their respective source sounds [46, 83, 84], (ii) the ability of humans to imitate sounds and link imitations to their source sound [83, 85, 86], (iii) the usefulness of vocal imitations for sound design [87, 88], (iv) their suitability as inputs for audio retrieval engines [44, 89–92], and (v) their relative importance when compared to text information [92, 93].

All in all, insights from these studies in QVI suggest that humans are generally skilled when performing and recognising vocal imitations of generic sounds, with most results pointing towards the high retrieval accuracy, speed, and usability of QVI systems when compared to query-by-text ones. Recent implementations of QVI systems use *deep representation learning* techniques to estimate the similarity between source sounds and vocal imitations through metric learning, which significantly increases the retrieval accuracy of QVI systems [44, 45, 92, 94].

Regarding particular relevant studies, Blancas et al. [91] created a supervised QVI model using heuristic features and a Support Vector Machine (SVM) classifier. They compared the SVM's performance to that of a Naïve Bayes classifier and found that the SVM significantly improved results: They also compared the performances of the SVM with 472 input features and the same model with 31 selected features via a Correlation-based Feature Selection (CFS) algorithm and found that selecting features this way meant a drastic improvement in accuracy scores.

Zhang and Duan [95] trained a data-driven model based on a supervised Stacked Auto-Encoder (SAE) architecture and an SVM that classified vocal imitations. They evaluated the algorithm using the accuracy and the Mean Reciprocal Rank (MRR) metrics in various sound categories including acoustic instruments, commercial synthesisers, everyday sounds, and single synthesiser sounds. They compared the performance of features learnt using the SAE with several extracted MFCCs and the former ouperformed the latter in every metric for all types of sounds.

The same authors of the earlier study, Zhang and Duan, proposed yet another deep learning-based algorithm called IMISOUND [94] that operated in an unsupervised manner, using the SAE above for feature extraction. Instead of using an SVM, the authors employed both the Kullback-Leibler divergence and the Dynamic Time Warping distance to measure similarity between embeddings. Results with these learnt features still excelled those calculated with MFCCs and were similar to those from their earlier study in [95]. They also found that this unsupervised system performed significantly worse than the earlier one in [95] when it came to everyday sounds, which reached around half of the original accuracy.

Still the same authors, Zhang and Duan, later presented an end-to-end Siamese Networks (SN) system of CNNs [96] that leveraged transfer and metric learning to measure similarity between samples. They experimented with three types of SN models: a tied one, where the two subnetworks of the SN shared exactly the same weights and biases in all layers; an untied model, where the two towers did not share weights nor biases while sharing architecture; and the partially tied model, where the weights and biases in the two towers are not shared for the first half of convolutional layers, but are shared for the last half. The algorithm, called IMINET, displayed a significant improvement over the earlier one, IMISOUND, when the probabilities of the three types of models (tied, untied, and partially tied) were multiplied to make the final decision.

A more advanced model that the system above was presented as TL-IMINET in [44]. While the two convolutional subnetworks in the original IMINET model both took 72 CQT bins and 129 time frames with a hop size of 26.25 ms, the new TL-IMINET model used input representations of 39 Mel bands by 482 time frames as input for the subnetwork that analysed vocal imitations and of 128 mel bands by 128 time frames for the subnetwork that analysed the sound recordings to imitate. Therefore, domain knowledge of the optimal frequency and time resolution for sound recordings and their vocal imitations was applied to maximise performance. The TL-IMINET model was shown to have slightly better results for most sound categories when its probabilities were multiplied with the ones derived from the IMISOUND model.

This TL-IMINET model was further explored and interpreted in [97]. There, authors visualised the input patterns that maximise the activation of different neurons in each convolutional subnetwork of the TL-IMINET model and visualised the imitation-sound input pairs that maximised the activation of different neurons in the layers relative to the fully-connected network that joins the convolutional subnetworks for metric learning. The findings revealed how the model learnt to conceptualise sounds and imitations and proved that performing transfer learning on the TL-IMINET model helped to improve its performance.

Lastly, an interesting study on the appropriateness of vocal imitations for sound retrieval was carried out by Kim et al. in [98]. They demonstrated that vocal imitation feedback in query-by-example systems improves the retrieval performance by a statistically significant margin, while exploring methods of how to combine multiple kinds of similarity estimation methods to achieve this feedback.

2.2.3.4 Sound Event Detection

Sound Event Detection (SED) is concerned with both the detection and classification of generic sound events of usually environmental nature. By this definition, SED could be considered the parent field of both VPT and ADT, with its related successful algorithms being potentially useful for VPT and vice versa. SED-related algorithms have also been traditionally divided into heuristic and data-driven depending on the approach to solve the task. See [12] for a more in-detail review of most of these algorithms and others.

Heuristic SED methods first extract audio features from audio and then feed these features to a machine learning algorithm that detects and classifies sound events. Some of the most popular features to extract from audio signals for heuristic SED are the MFCCs with their first and second derivatives [99], the FFT spectrogram [100], and the Mel spectrogram [101]. Regarding algorithms, GMM-HMM models coupled with Viterbi algorithm for sequence decoding were extensively used throughout the literature [99], as well as NMF-based approaches with regression algorithms [101].

Data-driven SED methods usually take the energies in the Log Mel spectrogram as input features [102] and sometimes other complementary features like pitch and harmonic content [103].

Architectures of neural networks include MLPs [104], BRNNs [105], CNNs [106], and Capsule Neural Networks (CapsNets) [102], which are a type of biologically-inspired CNNs whose layers are arranged so as to better model hierarchical relationships. CRNNs have also been frequently used in SED [107, 108] More sophisticated hybrid network methodologies also emerged in SED as a way to overcome limitations like the large amounts of strongly-labelled training data. Some of the successful approaches in this front included three-dimensional CNNs that modelled both Log Mel energies and information related to the Generalized Cross-Correlation Phase Transform (GCC-PHAT) [109] and a CRNN-based Auxiliary Classifier Generative Adversarial Network (AC-GAN) [110].

The results that these approaches output significantly favour data-driven approaches with respect to heuristic ones. As an example of which, the evaluation of the heuristic algorithm described in [101] using the development subset of the TUT-SED dataset gave a total error rate of 69.5%, while the algorithm described in [102] gave a total error rate of 36.0% using the same evaluation set.

2.2.3.5 Phoneme Recognition

Phonemes are the smallest sound units that can be perceptually distinguished by humans. As there are significantly fewer phonemes than syllables and words in a certain language, some early speech recognition systems attempted to retrieve phonetic information using acoustic features, which was also known as acoustic-phonetic analysis [111].

This field gathered some insights about which acoustic features were useful for phoneme recognition. Among others, studies mention several widely-used features in phoneme recognition like the auto-correlation coefficients, fundamental frequency, energy, zero crossing rate, spectral centroid, spectral bands and their derivatives, silences, and most prominent peak frequency [112]. Also, apart from acoustic features, psychoacoustic features are also commonly used in the field, which include information about pitch, duration, loudness, and timbre, also known as voice quality in speech recognition [113]. These psychoacoustical features overlap with prosodic (also known as suprasegmental) features of speech, which usually include accent, stress, rhythm, tone, pitch, and intonation [114].

Traditional and modern machine learning techniques include Hidden Markov Models (HMMs) [115], naive Bayes classifiers [116], KNNs [117], Support Vector Machines (SVMs) [118], Decision Trees (DTs) [119], Random Forests (RFs) [120], and DNNs which, at the same time, include connectionist networks [121], MLPs [122], RNN-LSTMs [123], and CNNs [124] among other more complex and hybrid models like DNN-HMMs [125].

2.3 Summary

In this chapter, we provided both a theoretical review, necessary to understand basic concepts in the thesis, and a literature review, necessary to give an appropriate sense of context to the thesis.

The first part of the theoretical review was dedicated to heuristic digital signal processing, covering the basic concepts related to DSP, heuristic feature extraction, and traditional machine learning. The second part was dedicated to data-driven digital signal processing, which covered the essential components of deep neural networks, their training dynamics, and their most popular and effective types in literature.

The literature review was structured so that the first two sections provided extensive reviews of VPT and DSRV respectively while the third section provided a synthesised review of five related fields: musical onset detection, automatic drum transcription, query by vocal imitation, sound event detection, and phoneme recognition.

References

[1] Merriam-Webster, “signal,” in *Merriam-Webster.com dictionary*. [Online]. Available: https://www.merriam-webster.com/dictionary/signal

[2] ——, “digital,” in *Merriam-Webster.com dictionary*. [Online]. Available: https://www.merriam-webster.com/dictionary/digital

[3] ——, “heuristic,” in *Merriam-Webster.com dictionary*. [Online]. Available: https://www.merriam-webster.com/dictionary/heuristic

[4] M. Müller, *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015.

[5] H. J. Nussbaumer, "The fast fourier transform," in *Fast Fourier Transform and Convolution Algorithms*. Springer, 1981, pp. 80–111.

[6] A. Weber and O. Scharenborg, "Models of spoken-word recognition," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 3, no. 3, pp. 387–401, 2012.

[7] P. Pedersen, "The mel scale," *Journal of Music Theory*, vol. 9, no. 2, pp. 295–308, 1965.

[8] J. O. Smith and J. S. Abel, "Bark and erb bilinear transforms," *IEEE Transactions on speech and Audio Processing*, vol. 7, no. 6, pp. 697–708, 1999.

[9] L. Rohr, "Evaluation of audio source separation in the context of 3d audio," EPFL, Tech. Rep., 2015.

[10] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the cuidado project," *CUIDADO Ist Project Report*, vol. 54, no. 0, pp. 1–25, 2004.

[11] M. Sahidullah and G. Saha, "Design, analysis and experimental evaluation of block based transformation in mfcc computation for speaker recognition," *Speech communication*, vol. 54, no. 4, pp. 543–565, 2012.

[12] T. K. Chan and C. S. Chin, "A comprehensive review of polyphonic sound event detection," *IEEE Access*, vol. 8, pp. 103 339–103 373, 2020.

[13] Z.-H. Zhou, *Machine learning*. Springer Nature, 2021.

[14] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[15] D. Stutz, "Learning shape completion from bounding boxes with cad shape priors," 2017.

[16] L. Noriega, "Multilayer perceptron tutorial," *School of Computing. Staffordshire University*, 2005.

[17] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-

level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[19] S. K. Kumar, "On weight initialization in deep neural networks," *arXiv preprint arXiv:1704.08863*, 2017.

[20] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] W. H. L. Pinaya, S. Vieira, R. Garcia-Dias, and A. Mechelli, "Autoencoders," in *Machine learning*. Elsevier, 2020, pp. 193–208.

[25] D. P. Kingma, M. Welling *et al.*, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[26] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in neural information processing systems*, 2015, pp. 3483–3491.

[27] G. Koch, R. Zemel, R. Salakhutdinov *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2. Lille, 2015.

[28] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[29] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015, pp. 84–92.

[30] S. Böck, F. Krebs, and M. Schedl, "Evaluating the online capabilities of onset detection methods." in *Ismir*. Citeseer, 2012, pp. 49–54.

[31] A. F. S. Ramires, "Automatic transcription of vocalized percussion," 2017.

[32] G. Peeters, B. L. Giordano, P. Susini, N. Misdariis, and S. McAdams, "The timbre tool-

box: Extracting audio descriptors from musical signals," *The Journal of the Acoustical Society of America*, vol. 130, no. 5, pp. 2902–2916, 2011.

[33] C. Sammut and G. I. Webb, *Encyclopedia of machine learning.* Springer Science & Business Media, 2011.

[34] A. Kapur, M. Benning, and G. Tzanetakis, "Query-by-beat-boxing: Music retrieval for the dj," in *Proceedings of the International Conference on Music Information Retrieval*, 2004, pp. 170–177.

[35] T. Nakano, J. Ogata, M. Goto, and Y. Hiraga, "A drum pattern retrieval method by voice percussion," *Database (Musical Instrument Sound)*, vol. 3, p. 1, 2004.

[36] T. Nakano, M. Goto, J. Ogata, and Y. Hiraga, "Voice drummer: A music notation interface of drum sounds using voice percussion input," *Proc. of UIST 2005 (Demos)*, pp. 49–50, 2005.

[37] A. Hazan, "Towards automatic transcription of expressive oral percussive performances," in *Proceedings of the 10th international conference on Intelligent user interfaces*, 2005, pp. 296–298.

[38] E. Sinyor, C. M. Rebecca, D. Mcennis, and I. Fujinaga, "Beatbox classification using ace," in *Proceedings of the International Conference on Music Information Retrieval.* Citeseer, 2005.

[39] D. Stowell and M. D. Plumbley, "Delayed decision-making in real-time beatbox percussion classification," *Journal of New Music Research*, vol. 39, no. 3, pp. 203–213, 2010.

[40] B. Series, "Method for the subjective assessment of intermediate quality level of audio systems," *International Telecommunication Union Radiocommunication Assembly*, 2014.

[41] K. Hipke, M. Toomim, R. Fiebrink, and J. Fogarty, "Beatbox: End-user interactive definition and training of recognizers for percussive vocalizations," in *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, 2014, pp. 121–124.

[42] B. Picart, S. Brognaux, and S. Dupont, "Analysis and automatic recognition of human beatbox sounds: A comparative study," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP).* Ieee, 2015, pp. 4255–4259.

[43] A. Ramires, R. Penha, and M. E. Davies, "User specific adaptation in automatic transcription of vocalised percussion," *arXiv preprint arXiv:1811.02406*, 2018.

[44] Y. Zhang, B. Pardo, and Z. Duan, “Siamese style convolutional neural networks for sound search by vocal imitation,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 2, pp. 429–441, 2018.

[45] A. Mehrabi, K. Choi, S. Dixon, and M. Sandler, “Similarity measures for vocal-based drum sample retrieval using deep convolutional auto-encoders,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Ieee, 2018, pp. 356–360.

[46] A. Mehrabi, S. Dixon, and M. Sandler, “Vocal imitation of percussion sounds: On the perceptual similarity between imitations and imitated sounds,” *Plos one*, vol. 14, no. 7, p. e0219955, 2019.

[47] S. Dixon, “Onset detection revisited,” in *Proceedings of the 9th International Conference on Digital Audio Effects*, vol. 120. Citeseer, 2006, pp. 133–137.

[48] M. Goto and Y. Muraoka, “Beat tracking based on multiple-agent architecture-a real-time beat tracking system for audio signals,” in *Proceedings of the Second International Conference on Multiagent Systems*, 1996, pp. 103–110.

[49] P. Masri, “Computer modelling of sound for transformation and synthesis of musical signals.” Ph.D. dissertation, University of Bristol, 1996.

[50] S. Hainsworth, M. D. Macleod *et al.*, “Onset detection in musical audio signals,” in *Icmc*. Citeseer, 2003.

[51] P. M. Brossier, “Automatic annotation of musical audio for interactive applications,” Ph.D. dissertation, 2006.

[52] J. P. Bello and M. Sandler, “Phase-based note onset detection for music signals,” in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP’03).*, vol. 5. Ieee, 2003, pp. V–441.

[53] C. Duxbury, J. P. Bello, M. Davies, M. Sandler *et al.*, “Complex domain onset detection for musical signals,” in *Proc. Digital Audio Effects Workshop (DAFx)*, vol. 1. Citeseer, 2003, pp. 6–9.

[54] S. Böck and G. Widmer, “Maximum filter vibrato suppression for onset detection,” in *Proc. of the 16th Int. Conf. on Digital Audio Effects (DAFx). Maynooth, Ireland (Sept 2013)*, vol. 7, 2013.

[55] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[56] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[57] F. Eyben, S. Böck, B. Schuller, and A. Graves, "Universal onset detection with bidirectional long-short term memory neural networks," in *Proc. 11th Intern. Soc. for Music Information Retrieval Conference, ISMIR, Utrecht, The Netherlands*, 2010, pp. 589–594.

[58] S. Böck, A. Arzt, F. Krebs, and M. Schedl, "Online real-time onset detection with recurrent neural networks," in *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx-12), York, UK*. sn, 2012.

[59] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[60] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[61] J. Schlüter and S. Böck, "Improved musical onset detection with convolutional neural networks," in *2014 ieee international conference on acoustics, speech and signal processing (icassp)*. Ieee, 2014, pp. 6979–6983.

[62] W. Kim, T. Kwon, and M. Choi, "Beatvoice: Drum pattern generation from vocal imitation."

[63] C.-W. Wu, C. Dittmar, C. Southall, R. Vogl, G. Widmer, J. Hockman, M. Müller, and A. Lerch, "A Review of Automatic Drum Transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 9, pp. 1457–1483, Sep. 2018, conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing.

[64] W. A. Schloss, *On the Automatic Transcription of Percussive Music–From Acoustic Signal to High-level Analysis*. Stanford University, 1985.

[65] F. Gouyon, F. Pachet, O. Delerue *et al.*, "On the use of zero-crossing rate for an application of classification of percussive sounds," in *Proceedings of the COST G-6 conference on Digital Audio Effects (DAFX-00), Verona, Italy*, vol. 5. Citeseer, 2000, p. 16.

[66] D. FitzGerald, R. Lawlor, and E. Coyle, "Sub-band independent subspace analysis for

drum transcription," 2002.

[67] ——, "Prior subspace analysis for drum transcription," in *Audio Engineering Society Convention 114*. Audio Engineering Society, 2003.

[68] D. Van Steelant, K. Tanghe, S. Degroeve, B. De Baets, M. Leman, J.-P. Martens, and T. De Mulder, "Classification of percussive sounds using support vector machines," in *Proceedings of the annual machine learning conference of Belgium and The Netherlands, Brussels, Belgium*. Citeseer, 2004.

[69] J. Paulus and T. Virtanen, "Drum transcription with non-negative spectrogram factorisation," in *2005 13th European Signal Processing Conference*. IEEE, 2005, pp. 1–4.

[70] J. Paulus and A. Klapuri, "Drum sound detection in polyphonic music with hidden markov models," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2009, pp. 1–9, 2009.

[71] X. Serra *et al.*, "Musical sound modeling with sinusoids plus noise," *Musical signal processing*, pp. 91–122, 1997.

[72] A. Maximos, A. Floros, M. N. Vrahatis, and N. Kanellopoulos, "Real-time drums transcription with characteristic bandpass filtering," in *Proceedings of the 7th Audio Mostly Conference: A Conference on Interaction with Sound*, 2012, pp. 152–159.

[73] N. Gajhede, O. Beck, and H. Purwins, "Convolutional neural networks with batch normalization for classifying hi-hat, snare, and bass percussion sound samples," in *Proceedings of the Audio Mostly 2016*, 2016, pp. 111–115.

[74] C. Southall, R. Stables, and J. Hockman, "Automatic Drum Transcription Using Bi-directional Recurrent Neural Networks," *New York City*, p. 7, 2016.

[75] C. Southall, N. Jillings, R. Stables, and J. Hockman, "Adtweb: An Open Source Browser Based Automatic Drum Transcription System," p. 2.

[76] R. Vogl, M. Dorfer, and P. Knees, "Recurrent neural networks for drum transcription." in *ISMIR*, 2016, pp. 730–736.

[77] D. Stowell and M. D. Plumbley, "Characteristics of the beatboxing vocal style," *Dept. of Electronic Engineering, Queen Mary, University of London, Technical Report, Centre for Digital Music C4DMTR-08-01*, 2008.

[78] R. Vogl, M. Dorfer, G. Widmer, and P. Knees, "Drum transcription via joint beat and drum

modeling using convolutional recurrent neural networks." in *Ismir*, 2017, pp. 150–157.

[79] R. Vogl, G. Widmer, and P. Knees, "Towards multi-instrument drum transcription," *arXiv:1806.06676 [cs, eess]*, Oct. 2018, arXiv: 1806.06676. [Online]. Available: http://arxiv.org/abs/1806.06676

[80] K. Choi and K. Cho, "Deep unsupervised drum transcription," *arXiv preprint arXiv:1906.03697*, 2019.

[81] Y. Wang, J. Salamon, M. Cartwright, N. J. Bryan, and J. P. Bello, "Few-shot drum transcription in polyphonic music," *arXiv preprint arXiv:2008.02791*, 2020.

[82] R. Ishizuka, R. Nishikimi, and K. Yoshii, "Global structure-aware drum transcription based on self-attention mechanisms," *Signals*, vol. 2, no. 3, pp. 508–526, 2021.

[83] G. Lemaitre, O. Houix, F. Voisin, N. Misdariis, and P. Susini, "Vocal imitations of non-vocal sounds," *PloS one*, vol. 11, no. 12, p. e0168167, 2016.

[84] A. Delgado, C. Saitis, M. Sandler *et al.*, "Spectral and temporal timbral cues of vocal imitations of drum sounds." International Conference on Timbre, 2020.

[85] M. Cartwright and B. Pardo, "Vocalsketch: Vocally imitating audio concepts," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2015, pp. 43–46.

[86] A. Mehrabi, S. Dixon, and M. B. Sandler, "Vocal imitation of synthesised sounds varying in pitch, loudness and spectral centroid," *The Journal of the Acoustical Society of America*, vol. 141, no. 2, pp. 783–796, 2017.

[87] G. Lemaitre, P. Susini, D. Rocchesso, C. Lambourg, and P. Boussard, "Non-verbal imitations as a sketching tool for sound design," in *International Symposium on Computer Music Multidisciplinary Research*. Springer, 2013, pp. 558–574.

[88] M. Cartwright and B. Pardo, "Synthassist: an audio synthesizer programmed with vocal imitation," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 741–742.

[89] M. Helén and T. Virtanen, "Audio query by example using similarity measures between probability density functions of features," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2010, pp. 1–12, 2009.

[90] G. Lemaitre, A. Dessein, P. Susini, and K. Aura, "Vocal imitations and the identification

of sound events," *Ecological psychology*, vol. 23, no. 4, pp. 267–307, 2011.

[91] D. S. Blancas and J. Janer, "Sound retrieval from voice imitation queries in collaborative databases," in *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*. Audio Engineering Society, 2014.

[92] Y. Zhang, J. Hu, Y. Zhang, B. Pardo, and Z. Duan, "Vroom! a search engine for sounds by vocal imitation queries," in *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval*, 2020, pp. 23–32.

[93] G. Lemaitre and D. Rocchesso, "On the effectiveness of vocal imitations and verbal descriptions of sounds," *The Journal of the Acoustical Society of America*, vol. 135, no. 2, pp. 862–873, 2014.

[94] Y. Zhang and Z. Duan, "Imisound: An unsupervised system for sound query by vocal imitation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Ieee, 2016, pp. 2269–2273.

[95] ——, "Retrieving sounds by vocal imitation recognition," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*. Ieee, 2015, pp. 1–6.

[96] ——, "Iminet: Convolutional semi-siamese networks for sound search by vocal imitation," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2017, pp. 304–308.

[97] ——, "Visualization and interpretation of siamese style convolutional neural networks for sound search by vocal imitation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Ieee, 2018, pp. 2406–2410.

[98] B. Kim and B. Pardo, "Improving content-based audio retrieval by vocal imitation feedback," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Ieee, 2019, pp. 4100–4104.

[99] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Context-dependent sound event detection," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2013, no. 1, pp. 1–13, 2013.

[100] A. Mesaros, T. Heittola, O. Dikmen, and T. Virtanen, "Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class

activity annotations," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP).* IEEE, 2015, pp. 151–155.

[101] V. Bisot, S. Essid, and G. Richard, "Overlapping sound event detection with supervised nonnegative matrix factorization," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2017, pp. 31–35.

[102] F. Vesperini, L. Gabrielli, E. Principi, and S. Squartini, "Polyphonic sound event detection by using capsule neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 310–322, 2019.

[103] S. Adavanne, G. Parascandolo, P. Pertilä, T. Heittola, and T. Virtanen, "Sound event detection in multichannel audio using spatial and harmonic features," *arXiv preprint arXiv:1706.02293*, 2017.

[104] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Polyphonic sound event detection using multi label deep neural networks," in *2015 international joint conference on neural networks (IJCNN).* IEEE, 2015, pp. 1–7.

[105] G. Parascandolo, H. Huttunen, and T. Virtanen, "Recurrent neural networks for polyphonic sound event detection in real life recordings," in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP).* IEEE, 2016, pp. 6440–6444.

[106] X. Xia, R. Togneri, F. Sohel, and D. Huang, "Confidence based acoustic event detection," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2018, pp. 306–310.

[107] E. Cakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, 2017.

[108] S. Jung, J. Park, and S. Lee, "Polyphonic sound event detection using convolutional bidirectional lstm and synthetic data-based transfer learning," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2019, pp. 885–889.

[109] S. Adavanne, A. Politis, and T. Virtanen, "Multichannel sound event detection using 3d convolutional neural networks for learning inter-channel features," in *2018 international joint conference on neural networks (IJCNN).* IEEE, 2018, pp. 1–7.

[110] X. Xia, R. Togneri, F. Sohel, and D. Huang, "Auxiliary classifier generative adversarial network with soft labels in imbalanced acoustic event detection," *IEEE Transactions on Multimedia*, vol. 21, no. 6, pp. 1359–1371, 2018.

[111] D. H. Klatt, "Speech perception: A model of acoustic–phonetic analysis and lexical access," *Journal of phonetics*, vol. 7, no. 3, pp. 279–312, 1979.

[112] A. A. Ali, J. Van der Spiegel, P. Mueller, G. Haentjens, and J. Berman, "An acoustic-phonetic feature-based system for automatic phoneme recognition in continuous speech," in *1999 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 3. IEEE, 1999, pp. 118–121.

[113] M. Haggard, Q. Summerfield, and M. Roberts, "Psychoacoustical and cultural determinants of phoneme boundaries: Evidence from trading f0 cues in the voiced–voiceless distinction," *Journal of phonetics*, vol. 9, no. 1, pp. 49–62, 1981.

[114] A. Fox, *Prosodic Features and Prosodic Structure: The Phonology of'Suprasegmentals'*. OUP Oxford, 2002.

[115] K. K. Paliwal, "A study of lsf representation for speaker-dependent and speaker-independent hmm-based speech recognition systems," in *International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1990, pp. 801–804.

[116] J. Ye, R. J. Povinelli, and M. T. Johnson, "Phoneme classification using naive bayes classifier in reconstructed phase space," in *Proceedings of 2002 IEEE 10th Digital Signal Processing Workshop, 2002 and the 2nd Signal Processing Education Workshop.* IEEE, 2002, pp. 37–40.

[117] M. Rizwan and D. V. Anderson, "Using k-nearest neighbor and speaker ranking for phoneme prediction," in *2014 13th international conference on machine learning and applications*. IEEE, 2014, pp. 383–387.

[118] P. Clarkson and P. J. Moreno, "On the use of support vector machines for phonetic classification," in *1999 IEEE international conference on acoustics, speech, and signal processing. Proceedings. ICASSP99 (cat. No. 99CH36258)*, vol. 2. IEEE, 1999, pp. 585–588.

[119] T. Shinozaki and S. Furui, "Error analysis using decision trees in spontaneous presentation speech recognition," in *IEEE Workshop on Automatic Speech Recognition and Under-*

standing, 2001. ASRU'01. IEEE, 2001, pp. 198–201.

[120] I. Atsonios, "Phoneme recognition via coupling landmark isomap and random forests," in *2009 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT).* IEEE, 2009, pp. 484–489.

[121] A. Waibel, "Modular construction of time-delay neural networks for speech recognition," *Neural computation*, vol. 1, no. 1, pp. 39–46, 1989.

[122] P. Schwarz, P. Matejka, and J. Cernocky, "Hierarchical structures of neural networks for phoneme recognition," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 1. IEEE, 2006, pp. I–I.

[123] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional lstm networks for improved phoneme classification and recognition," in *International conference on artificial neural networks*. Springer, 2005, pp. 799–804.

[124] D. Palaz, M. M. Doss, and R. Collobert, "Convolutional neural networks-based continuous speech recognition using raw speech signal," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2015, pp. 4295–4299.

[125] M. L. Seltzer and J. Droppo, "Multi-task learning in deep neural networks for improved phoneme recognition," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 6965–6969.

Chapter 3

Data

As we briefly discussed in the thesis's introduction chapter, there exist some issues with data scarcity in QVP that can potentially constitute a major problem when applying data-hungry algorithms like neural networks. User-driven training sets relative to amateur vocal percussion transcription often contain a few samples per user, and this is also the case for user-driven DSRV, whose datasets normally contain a few reference sounds with a single associated vocal imitation for each of them. Hence, we conducted a search for publicly available vocal percussion datasets and also the curation of additional datasets for VPT and DSRV that can support existing datasets when being fed to data-driven algorithms.

This chapter presents the datasets that are going to be used throughout the thesis for data-driven QVP, some of them already available in literature and others to be crafted.

3.1 Publicly Available Datasets

Here, we first take a look at the datasets that were already published in literature at the start time of the thesis. We included the datasets that could potentially allow VPT and DSRV algorithms to be trained and evaluated with both heuristic and data-driven algorithms.

3.1.1 Vocal Percussion Transcription

Early studies in VPT were usually carried out focusing on three classes of imitated sounds: kick drum, snare drum, and hi-hat. In most cases, a vocal percussion dataset of a few hundred examples was recorded for evaluation purposes, oriented to general rhythmic vocal percussion [1–4] or just beatboxing [3, 5–8]. While most of these studies did not publish the data they used in their experiments, two of them released the vocal percussion recordings and annotation information they used in their studies.

The first one is the *beatboxset1* (BTX) by Stowell et al. [8]. It gathers experienced beatboxers to record 14 audio files (one per participant) with a mean duration of 47 seconds which resulted in a total of 3,611 annotated vocal percussion sounds. Both typical drum sounds and beatbox-specific ones were labelled, being the audio files recorded in several environmental conditions (different microphones, equipment, noise levels...). The authors provide onset annotations and eleven types of labels including kick drum (k), k-like snare (sk), b-like and p-like snare (sb), other snares (s), closed hi-hat (hc), opened hi-hat (ho), breath sound (br), humming (m), speech and singing (v), miscellaneous (x), and undefined (?).

The second one is the *Live Vocalised Transcription* (LVT) dataset by Ramires et al. [4]. It contained 841 sounds from 20 amateur participants using three microphones with different noise levels and sound qualities (condenser, laptop, and ipad microphones). It was recorded using the "isolated samples" strategy (see section 1.1) with two files per participant being provided: one of them containing the imitation of a simple drum loop and the other a free rhythmic improvisation. The authors provide onset annotations and three types of labels including kick drum ($Kick$), snare drum ($Snare$), and closed hi-hat (HH).

Although data-driven algorithms are especially data-hungry, 3,611 beatbox sounds are arguably enough samples for neural networks to work with, provided that strong regularisation techniques like data augmentation (see 2.1.2.3) take place during the training phase. However, it cannot be said the same thing about the 841 sounds from the LVT dataset, especially if one of the goals is to explore user-driven VPT systems for amateur vocal percussion. This insufficient amount of amateur vocal percussion sounds to feed data-driven algorithms inspired us to record the dataset described in section 3.2.

3.1.2 Drum Sample Retrieval by Vocalisation

As with VPT, the vocal imitation data that was used in the majority of past DSRV studies was not released. Still, there were two publicly available datasets of interest for this thesis, one of them containing generic vocal imitations that included drum sounds and the other containing drum vocal imitations.

The first of these datasets is the *Vocal Imitation Set* (VIS) by Kim et al. [9]. This was directed to the vocal imitation of sound events in general, containing a total of 11,242 crowd-sourced imitations from 302 different classes that included environmental, synthesised, and musical sources. It featured imitations of several percussion instruments including 38 vocal imitations of a kick drum sample, 48 of a snare drum sample, 52 of a hi-hat sample, and 33 of a crash cymbal sample.

The second dataset by Mehrabi et al. [10] contains 420 vocal imitations of several drum sounds. We refer here to this the *Mehrabi Drum Vocalisations* (MDV) dataset. More specifically, 14 musicians were asked to imitate 30 percussion sounds, which included 6 kick drums, 6 snare drums, 6 hi-hats, 6 toms, and 6 cymbals. The recording was done using a condenser microphone in an acoustically treated room. The dataset also included a collection of drum-imitation similarity ratings from 63 human listeners.

The number of drum vocal imitations in the VIS dataset (171) is insufficient to train data-driven DSRV systems. The imitations are also not suitable for the evaluation of DSRV systems, as only four of the sounds provided are drum-specific single shots. Also, the reference drum samples pertain to four different instrument classes (kick drum, snare drum, hi-hat, and crash cymbal), which would not allow any within-class analysis and retrieval. While the MDV dataset is also not suitable for training due to data scarcity, its structure based on drum-imitation pairs allows it to be used as an evaluation dataset for DSRV. Not only does it contain enough reference drum samples for evaluation (30 sounds), but also includes 6 drum sounds from each of the 5 classes (kick drums, snare drums, hi-hats, toms, and cymbals), which permits evaluation on within-class imitations. Also, the imitations provided were taken from 14 different users, enabling user-driven analysis to check for individual differences in vocal imitation styles. The training of data-driven algorithms can be done in the same way as in [11]: learning embeddings using deep autoencoders and evaluating their capabilities of predicting the correct reference samples or listeners' similarity ratings.

3.2 The Amateur Vocal Percussion Dataset

As discussed in 3.1.1, we were in need of gathering more data in order to explore the potential of user- and data-driven algorithms to model amateur vocal percussion sounds. This is the main reason why we recorded the *Amateur Vocal Percussion* (AVP) dataset, whose audio files and annotations can be found in the link `https://doi.org/10.5281/zenodo.3250230`.

The AVP dataset is characterised by the following attributes:

- 28 participants.
- A total of 9,780 amateur vocal percussion sounds in 280 audio files.
- Annotated onsets and labels (kick, snare, closed hi-hat, and opened hi-hat).
- Recorded with one microphone (MacBook Pro's built-in microphone).
- 2 modalities: personal imitations (4,873 sounds) and fixed imitations (4,905 sounds).
- 5 files for each modality each (four files with sounds of the same class and one with an improvisation).

The AVP focuses exclusively on people with little or no experience in beatboxing, making it suitable for research on amateur vocal percussion transcription. It is, to our knowledge, the largest vocal percussion dataset both in the number of participants and number of vocal percussion sounds. It also incorporates a second subset of fixed drum imitations; this is, four specific syllables are given to all participants to imitate the four instruments. This fixed subset was

Instrument - '*label*'	Personal	Fixed	Improvisation
Kick Drum - '*kd*'	*799*	*818*	*1201*
Snare Drum - '*sd*'	*813*	*839*	*811*
Closed Hi-Hat - '*hhc*'	*799*	*833*	*673*
Opened Hi-Hat - '*hho*'	*816*	*830*	*548*

Table 3-A: Number of vocal percussion sounds for every drum instrument and subset.

recorded in order to investigate how practical a fixed technique of performing vocal percussion could be, as deriving accurate algorithms to classify the personal subset as a whole is relatively challenging. The total number of vocal percussion sounds belonging to each of the four classes are gathered in table 3-A.

A notable weakness of the AVP dataset is that it was recorded without using any extra microphones or in an acoustically untreated room. However, this dataset is directed to music producers, with a big percentage of them working with their laptops in regular rooms. In that way, the realism of the AVP dataset recording context, which features quiet environmental noises as well, could be proven helpful for the algorithms to perform well in non-controllable external conditions.

3.2.1 Recording

The materials used to record the dataset were a MacBook Pro laptop, GarageBand software, and a closed room of approximately 40 m^3. The experiment took an average of 15 minutes per participant.

As the first step of the process, a standard loop featuring kick and snare was presented and participants were asked to both reproduce it vocally and write down in a provided notebook the onomatopoeias of the sounds they used to imitate the drums. This performance was intended as a first contact with the process and it did not get recorded, while the annotation of the onomatopoeias was done to facilitate the recalling of imitations and to better stick to them. As it turned out to be hard for participants to imitate complex beats with hi-hat included in the preliminary tests, five isolated repetitions of a closed hi-hat and an open hi-hat sounds were presented instead. Participants decided their imitations and wrote down their onomatopoeias in the same way as with the kick drum and snare drum.

Once participants were familiarised with the task, they were asked to sit naturally in front of the computer, as they would normally do. Then, the recording of the dataset started with two modalities taking place: personal and fixed. For the personal modality, participants used their own vocal imitations to record around 25 vocal percussion sounds of kick drum, snare drum, closed hi-hat, and opened hi-hat sounds in four separated audio files. The sounds followed a simple rhythmic loop (one crotchet and two quavers) and a 90-BPM metronome track was provided through headphones to the participants in order to record them. An audio file of improvised vocal percussion featuring all or most instruments was recorded afterward. For the fixed modality, the procedure above was repeated once more, but now four specific sounds were given to the subjects. These fixed imitations were based on speech syllables so to feel natural for participants to reproduce, and their timbral characteristics were intended to mimic the imitated percussion instruments. A /pm/ syllable would correspond to the kick drum, /ta/ to the snare drum, /ti/ to the closed hi-hat, and a /tʃi/ to the opened hi-hat. The articulation of these sounds, as they were performed in a percussive setting, generally resulted in brighter transient signals and more inharmonic steady-state signals compared to usual speech sounds.

3.2.2 Post-Processing and Annotation

Once the raw audio files were recorded, two post-processing stages took place to prepare them for analysis: the trimming of silent regions at the beginning and the end of each file to reduce their size and the removal of in-between passages where the participants made accidental mistakes or notable pauses. The annotation of the files was manually carried out right after this cleaning process, using Sonic Visualiser [12] to write down the onset locations and class labels. Both onset locations and class labels were determined by one annotator and were validated and corrected by two other annotators, all of them with musical training. As illustrated in table 3-A, the tag ‘*kd*’ was used for kick drum, ‘*sd*’ for snare drum, ‘*hhc*’ for closed hi-hat, and ‘*hho*’ for opened hi-hat.

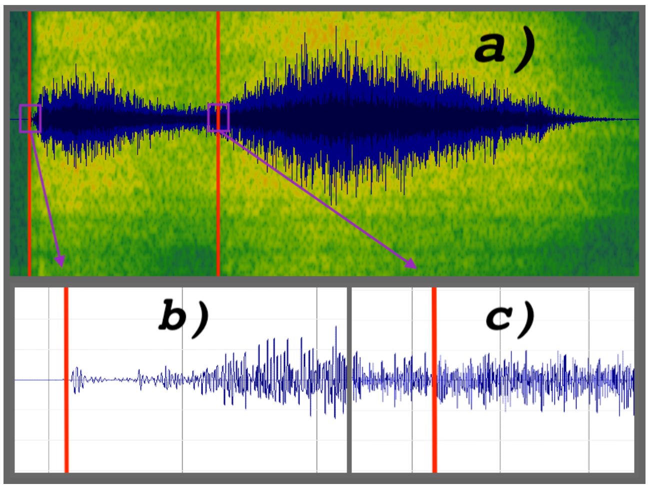


Figure 3.1: Annotation of two onsets, corresponding to two /s/ phonemes. Figure 1a) displays the vocal percussion sounds with their onsets marked in red. The waveform is plotted in blue and the magnitude spectrogram in green. Figures 1b) and 1c) are zoomed images of the first and the second onset respectively.

An important point to note here is that choice of the exact starting moment of a sound event is generally considered to be dependent on the task at hand. For instance, if the goal is preparing a vocal percussion sound with a long transient to be used as a sound triggered by a MIDI message, the onset would sometimes be preferably placed near the point of maximum energy in the signal. In our case, we decided to place the onsets at the very beginning of the sound, where the percussive transient starts to build the sound. This is due to the fact that our primary goal is to classify the vocal percussion sound, and its transient region could be informative as well, especially for the time-constrained online VPT case.

Also, on a few occasions during the annotation process, two vocal percussion sounds were very close to each other or even seemed to be part of one single utterance. A joint approach of waveform visualisation, spectrogram visualisation, and listening at quarter speed was employed to solve the ambiguity in these cases. An example of how these specific annotations were approached is illustrated in figure 3.1.

3.2.3 Observations

A list of observations worth commenting on was made while recording and listening back to the audio files in the AVP dataset. These are the following:

- A small set of recorded vocal percussion sounds exhibit a form of double percussion (such as /suk/ or /br/), which can make onset detectors output the events as a whole without splitting them.
- Various participants got mistaken in a few occasions when recording the improvisation files, using other speech-like phonemes like /tʃa/ instead of /ta/ when imitating the snare drum or /dm/ instead of /pm/ when imitating the kick drum. These passages were omitted in the final version of the dataset.
- Several participants reported that the given fixed sounds, despite their resemblance with the original drum sounds, felt unnatural for them to perform.
- Participants generally improvised non-complex and predictable loops, which could make the classification routines benefit from a rhythmic pattern analyser.

Finally, there have been three participants whose vocal percussion sounds within the personal dataset were either not consistent with each other, unintelligible, or practically indistinguishable from the rest. The audio files and annotations pertaining to these cases are stored in the "Discarded" folder, although they could still be used for classification purposes.

3.3 Other Datasets and Extra Annotations

Apart from the AVP dataset described above, we created two more vocal percussion datasets. These were composed by already made royalty-free vocal percussion recordings on the internet and also from the audio files of one of the previously mentioned publicly available datasets. We also annotated the phonetic information relative to the vocal percussion sounds from the LVT dataset and the personal subset of the AVP dataset.

3.3.1 Freesound Beatbox Dataset

In order to build the Freesound Beatbox (FSB) dataset, we collected 4,296 beatbox sounds from 552 audio files from the Freesound repository [13]. The audio files contained vocal percussion performances from different beatboxers that could be either beatbox improvisations (65 files), which composed the *Multi* subset, or single isolated beatbox samples (487 sounds), which composed the *Single* subset. We annotated the onset location of all sounds using Sonic Visualiser [12].

The purpose of this dataset was three-fold: it could serve (i) as extra data for the training and evaluation of data-driven onset detection algorithms, (ii) as extra data for the training and evaluation of data-driven DSRV algorithms, and (iii) as unlabelled data when training semi-supervised beatbox classification systems.

3.3.2 VIS Percussive Dataset

The VIS Percussive (VIS-P) dataset gathered 3,393 sounds taken from the percussive vocal utterances in the VIS dataset [9] (see 3.1.2). Apart from the imitations of the four drum sounds in the original dataset, we also included sounds from imitation audio files provided that both their reference audio file and the files themselves contained percussive sounds. As with the FSB dataset, we only annotated the onset location of sounds using Sonic Visualiser [12], with no class information being provided.

This set could also be useful as extra data for training and evaluating data-driven onset detection and DSRV algorithms. Another possible use could be the unsupervised pre-training of amateur VPT algorithms, although the data provided by the LVT and AVP datasets together is tentatively enough to carry out the supervised training and, if needed, pre-training of amateur VPT algorithms.

3.3.3 Phonetic Annotations

We manually extended the annotations of the AVP and LVT datasets (onsets and instrument labels) so that they also contained the phonetic representation of vocal percussion sounds. This

Dataset	Mic.	Place	Type	# P	# S	Instruments	Phonemes
AVP-P	Lap.	Room	Amateur Vocal Perc.	28	4,873	Kick: 1,447 Snare: 1,253 C. Hi-Hat: 1,164 O. Hi-Hat: 1,009	Onset Phonemes: /p/: 1,325, /t/: 1,255, /tʃ/: 794, /ts/: 661, /tɕ/: 238, /k/: 227, /kg/: 120, /s/: 57, /tʒ/: 56, /ʡʢ/: 51, /kʃ/: 34, /!/: 29, /dʒ/: 26. Coda Phonemes: /x/: 2,242, /h/: 763, /ɑ/: 497, /u/: 376, /ʊ/: 287, /i/: 207, /ɯ/: 98, /ə/: 70, /æ/: 67, /ɘ/: 52, /ʌ/: 47, /ɪ/: 48, /o/: 41, /e/: 27, /ɐ/: 27, /œ/: 24
AVP-F	Lap.	Room	Amateur Vocal Perc.	28	4,905	Kick: 1,364 Snare: 1,204 O. Hi-Hat: 1,195 C. Hi-Hat: 1,141	-
LVT	Cond., lap., ipad.	Acoust. treated room	Amateur Vocal Perc.	20	841	C. Hi-Hat: 334 Kick: 329 Snare: 178	Onset Phonemes: /p/: 338, /ts/: 303, /t/: 168, /k/: 8, /ʡʢ/: 8, /tʃ/: 8, /ʔ/: 4, /s/: 3, /!/: 1 Coda Phonemes: /x/: 444, /u/: 243, /a/: 151, /h/: 2, /ʊ/: 1.
BTX	Misc.	Misc.	Beatbox	14	3,611	Misc.: 926 C. Hi-Hat: 882 Kick: 627 K Snare: 384 Humming: 230 P Snare: 204 Other Snares: 115 O. Hi-Hat: 105 Undefined: 85 Breath: 47 Singing: 6	-
VIS-P	Lap.	Room	Amateur Vocal Perc.	-	3,393	-	-
FSB	Misc.	Misc.	Beatbox	-	4,296	-	-
MDV	Cond.	Acoust. treated room	Vocal Imi.	14	420	Kick: 84 Snare: 84 Hi-Hat: 84 Toms: 84 Cymbals: 84	-

Table 3-B: Table summarising the information in the datasets used in this thesis.

information is divided between that relative to the *onset phonemes*, which are the first phonemes of the syllable and are usually plosive or fricative consonants in vocal percussion, and the *coda phoenemes*, which are the last phonemes of the syllable and are usually vowels, a breath sound (h), or silence (x). These phonemes were annotated following notation conventions from the International Phonetic Alphabet (IPA) [14].

These phonetic annotations will be used as labels for supervised feature learning in section 5.2 and also as means to choose adequate phonemes for online amateur vocal percussion transcription in section 6.1.

3.4 Summary

In this chapter, we described the different publicly available vocal percussion datasets in literature and the datasets that we put together and annotated ourselves to complement the former. Table 3-B lists all of these datasets along with their distinctive features. In terms of the total number of samples, we now have enough data to approach the tasks that this thesis explores from a data-driven perspective: vocal percussion onset detection, amateur vocal percussion classification, beatbox classification, and drum sample retrieval by vocal imitation.

References

[1] T. Nakano, J. Ogata, M. Goto, and Y. Hiraga, "A drum pattern retrieval method by voice percussion," *Database (Musical Instrument Sound)*, vol. 3, p. 1, 2004.

[2] T. Nakano, M. Goto, J. Ogata, and Y. Hiraga, "Voice drummer: A music notation interface of drum sounds using voice percussion input," *Proc. of UIST 2005 (Demos)*, pp. 49–50, 2005.

[3] E. Sinyor, C. M. Rebecca, D. Mcennis, and I. Fujinaga, "Beatbox classification using ace," in *Proceedings of the International Conference on Music Information Retrieval*. Citeseer, 2005.

[4] A. F. S. Ramires, "Automatic Transcription of Drums and Vocalised percussion," *Universidade do Porto*, 2017.

[5] A. Kapur, M. Benning, and G. Tzanetakis, “Query-by-beat-boxing: Music retrieval for the dj,” in *Proceedings of the International Conference on Music Information Retrieval*, 2004, pp. 170–177.

[6] A. Hazan, “Towards automatic transcription of expressive oral percussive performances,” in *Proceedings of the 10th international conference on Intelligent user interfaces*, 2005, pp. 296–298.

[7] K. Hipke, M. Toomim, R. Fiebrink, and J. Fogarty, “Beatbox: End-user interactive definition and training of recognizers for percussive vocalizations,” in *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, 2014, pp. 121–124.

[8] D. Stowell and M. D. Plumbley, “Delayed decision-making in real-time beatbox percussion classification,” *Journal of New Music Research*, 2010.

[9] B. Kim, M. Ghei, B. Pardo, and Z. Duan, “Vocal imitation set: a dataset of vocally imitated sound events using the audioset ontology,” *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, Nov. 2018.

[10] A. Mehrabi, K. Choi, S. Dixon, and M. Sandler, “Similarity measures for vocal-based drum sample retrieval using deep convolutional auto-encoders,” *arXiv:1802.05178*, Feb. 2018.

[11] ——, “Similarity measures for vocal-based drum sample retrieval using deep convolutional auto-encoders,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Ieee, 2018, pp. 356–360.

[12] C. Cannam, C. Landone, and M. Sandler, “Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files,” in *Proceedings of the 18th ACM international conference on Multimedia*. Acm, 2010, pp. 1467–1468.

[13] “Freesound,” https://www.freesound.org.

[14] I. P. Association, I. P. A. Staff *et al.*, *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*. Cambridge University Press, 1999.

Chapter 4

Vocal Percussion Onset Detection

Onset detection is the first stage of the transcription process. The systems in charge of performing the task, called onset detectors, inform the subsequent classification algorithms of where a certain sound event starts so that they can place the starting point of their analysis windows exactly there or strategically close to it.

Vocal percussion sounds often resemble speech phonemes and syllables, making onset detection slightly more challenging with vocal percussion than with regular monophonic percussion, whose instruments exhibit shorter and better-defined attacks. Additionally, when analysing syllables that include onset and coda phonemes (see 3.3.3), onset detection algorithms need to discriminate between the beginnings of onset phoneme sounds (e.g. /p/, /t/, /k/...) and those of coda phoneme sounds (e.g. /a/, /e/, /o/...), predicting only the onsets relative to the former and not the latter. A typical example of this scenario is illustrated in figure 4.1.

Another source of confusion that can affect onset detectors' performance is the distinction between onsets associated with breathing sounds from those associated with actual vocal percussion sounds. These breath sounds are a minority in the dataset we used to train and evaluate onset detection algorithms ($\sim$ 150 sounds) and most of them have relatively slow attacks (i.e. they are not strictly percussive). Despite this, we found that heuristic methods would sometimes incorrectly predict some of these breath sounds' onsets. Likewise, it was reported in the AVP

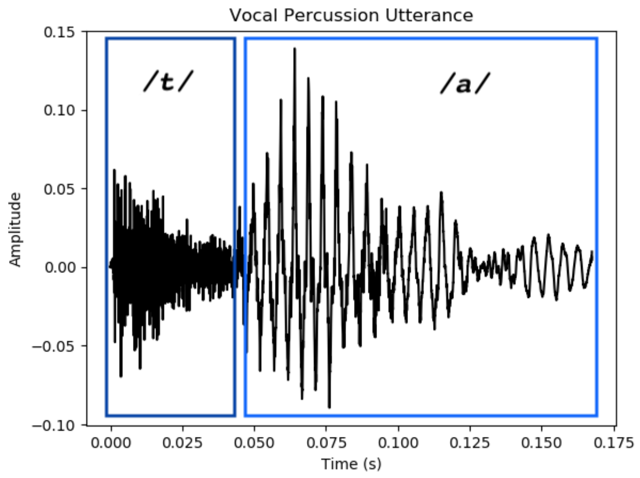


Figure 4.1: Waveform pertaining to a single vocal percussion sound event. The coda phoneme /a/ that completes the /ta/ syllable has a relatively marked transient, which could potentially confuse onset detectors.

dataset that a few participants sometimes pronounced two s-like phonemes (e.g. /ts/) very close to each other, resulting in poorly-defined onset locations for the second sounds. Also, in general, any percussive ambient sound that could have been recorded accidentally would constitute a source of confusion for algorithms.

The above-mentioned and other unidentified sources of confusion favour data-driven algorithms in terms of performance, as they are naturally drawn to recognising these challenging utterances so as to optimise onset prediction accuracy. We lean on this fact and in the high accuracies of data-driven algorithms for musical onset detection [1–4] to justify their exploration in a vocal percussion context.

In this chapter, we detail the main routines that we followed to evaluate the performances of several heuristic and data-driven onset detection methods. We describe these approaches and present the final results for both offline and online onset detection. In particular, we want to answer the following research questions:

- Are data-driven models more accurate than heuristic algorithms?
- Are the onsets predicted by data-driven models closer to ground-truth onsets than those predicted by heuristic onset detectors?

- How fast is data-driven models' inference compared to that of heuristic methods?
- Which onset detector algorithm is the most appropriate for offline[1] vocal percussion transcription?
- From what amount of time delay do vocal percussion onsets begin to be reasonably detectable in an online context?

4.1 Offline Onset Detection

In the offline scenario (i.e. non-real-time), onset detectors have access to whole vocal percussion performances and can use them for training. This contrasts with online (real-time) onset detection, where algorithms can only access a small buffer of an audio stream and have to make quick decisions on whether an onset just happened or not.

4.1.1 Methodology

Here we discuss four topics that give a complete account of our methodology. The first one provides an overview of the vocal percussion datasets used for offline onset detection and how we prepared and split them to build the final dataset. The second and the third topic would present the implementation details of heuristic and data-driven algorithms respectively. We explored three types of data-driven models for onset detection: a Bidirectional Recurrent Neural Network (BRNN), a Convolutional Neural Network (CNN), and a Convolutional Recurrent Neural Network (CRNN). Finally, the fourth topic would give an account of the training and evaluation methodology that we followed, including information about label pre-processing, parameter optimisation, and evaluation metrics among others.

4.1.1.1 Data

We trained, validated, and tested onset detection algorithms using all audio data and onset annotations from the AVP-P, AVP-F, LVT, BTX, VIS-P, and FSB datasets (see chapter 3), excluding

[1]Appropriateness of online onset detectors needs to be studied with results from online classification (section 6).

the files containing single isolated sounds. Before concatenating the audio files, we split the data into a train set with $\sim$85% of data and a test set with the remaining $\sim$15%. The files that made the test set were manually selected so that they contained performances that were representative of the whole dataset. The final set accounted for a total length of 118 minutes of audio data and 21,598 vocal percussion sound events.

To build input representations for the data-driven models (RNN, CNN, and CRNN), the dataset was first downsampled to 22,050 Hz and taken its spectrogram based on the Fast Fourier Transform (FFT) with a Hann window size of 11.6 ms and a hop size of 5.8 ms. The frequency resolution of the final representations was 128 bins. We split the data into time sequences of 16 frames with an overlap of 75% and fed these to the RNN and CNN (size 128x16). For each sequence of 16 frames (training sample), the networks calculated individual activations for those 16 time steps in the sequence, which were later flattened and averaged to give the final predicted onset activations. For CRNNs, data was also fed in sequences of 16 data points, except here each data point was a spectrogram of size 128x8 instead of a spectrum of 128 frequency bins.

4.1.1.2 Heuristic Algorithms

We evaluated the performance of heuristic methods based on the following audio descriptors: the spectral energy, phase domain, complex domain, HFC, spectral difference, KL divergence, MKL divergence, spectral flux, and superflux. Refer to section 2.1.1.1 for a theoretical overview of these descriptors.

Onset detection systems associated with the descriptors above functioned in the same way: first (i) the audio descriptor is extracted from the sounds' spectrum from the waveform (sampled at 44.1 kHz) in a frame-wise manner; then, (ii) an activation function is calculated for each of these frames based on the information that these descriptors provided; and lastly, (iii) a peak-picking process is carried out on the activation function to predict onsets on the frames whose associated activation value surpasses a certain threshold value. Section 2.1.1.1 details the calculation of activation functions and peak-picking.

Logarithmic compression and adaptive whitening of spectral data were applied in some of these onset detection approaches. In particular, logarithmic compression based on a Gamma (γ) parameter of magnitude spectrograms [5] (the smaller the value of γ, the more compressed the output spectrogram) was applied to onset detectors related to the complex domain (γ = 1), HFC (γ = 1), KL (γ = 0.02), MKL (γ = 0.02), and spectral flux (γ = 10), while adaptive whitening [6] was applied to those related to the complex domain, phase, KL, MKL, and spectral flux.

All onset detectors were implemented via the Aubio library [7] except the superflux method, for which we used the Madmom library [8].

4.1.1.3 Data-Driven Algorithms

We trained a bidirectional RNN with three bidirectional recurrent layers with 64 Gated Recurrent Units (GRU) [9] and hyperbolic tangent activation functions followed by a dropout layer of rate 0.15 and a final dense layer connecting the final GRU layer outputs to the 16 predictions relative to the 16 input frames.

The CNN was based on the state-of-the-art model described in [2] and had three convolutional blocks, a dropout layer of rate 0.5, and a final dense layer connecting the flattened feature maps to the 16 predictions. Each of the convolutional blocks had a single convolutional layer followed by a batch-normalisation layer, a Rectified Linear Unit (ReLU) activation function, and a max-pooling operation. The first block had 32 filters of kernel dimensions of 7x3 with strides of 1x1 in its convolutional layer and had a pooling kernel size of 1x4; the second block had 64 filters of dimensions 3x3 with strides 1x1 and a pooling kernel size of 2x4; and, lastly, the third block had 128 filters of dimensions 3x3 with strides 1x1 and a pooling kernel size of 2x4.

Finally, the CRNN was composed of a CNN module preceding an RNN module that operated with the feature maps from the former. The CNN module had two convolutional blocks and no dense layers. The first block had 16 filters of kernel dimensions of 7x3 with strides of 1x1 in its convolutional layer and had a max-pooling kernel size of 2x4, and the second block had 32 filters of dimensions 3x3 with strides 1x1 and a max-pooling kernel size of 2x8. Hence, the final feature maps had a dimensionality of 256x16. The RNN module had the same architecture as our RNN model but with a dropout layer of rate 0.2.

The number of filters, the number of GRU units, and the dropout rate of all networks were optimised via grid search (i.e., exhaustive search) to maximise performance accuracy.

4.1.1.4 Training and Evaluation

While one could use the whole audio data to predict all onsets in data-driven approaches, we employed a relatively small analysis window ($\sim$ 93 ms for CNNs and RNNs and $\sim$ 140 ms for CRNNs) in order to avoid unwanted effects like unnecessary processing delay and overfitting. Algorithms trained on the whole data would also be more likely to identify rhythmic patterns that affect predictions. This is usually beneficial for strictly rhythmical vocal percussion data, as it helps clear doubts when algorithms are unsure of whether an onset has happened or not based on acoustical data alone. While we encourage to explore this kind of rhythm modelling to inform onset detectors, we strictly focused on acoustical data in this project, choosing small analysis regions to make sure that algorithms have as little access to rhythmic information as possible (i.e. they are rhythm-agnostic). This way, potential users of VPT algorithms will not be bound to reproducing rhythmic performances to ensure a good transcription performance.

Data-driven models were trained for five iterations with different random seeds for weight initialisation. Their results were later averaged for reporting. We trained the models using an Adam optimisation algorithm [10], early stopping if the validation losses did not decrease after 20 epochs, and learning rate downscaling if validation loss did not decrease after 10 epochs. Instead of running a binary classification task (onset vs. non-onset), we set up a regression task with fuzzy labels [2]. In this way, we gave a class weight of 0.2 to the data frames right before the onset frames, 0.5 to the frames right after the onset, and 0.1 to the frames right after these.

To choose the decision threshold for heuristic approaches, we optimised its value using the train set and then predicted the final evaluation onsets in the test set using this value. To choose the threshold for data-driven approaches, we performed a seven-fold cross-validation routine with the train set. We calculated the decision threshold that optimised the F1-score in the validation set for each of the seven folds and then averaged these seven thresholds to get the final threshold value that is used to predict the test onsets and, hence, derive the final results. To run the evaluation, we chose the cross-validated model whose decision threshold was closer in absolute magnitude to the mean threshold value.

We found that applying a moving maximum smoothing window to the activation function had a beneficial effect for all networks performance-wise. We optimised the parameters of these moving maximum functions to optimise performance for individual networks. These parameters were the length of the moving windows from the current point to a previous point and their length from the current point to a future point.

We used the F1-Score to measure performance accuracy, which is defined as:

$$F_1 = 2 \cdot \frac{\text{Precision } \cdot \text{ Recall}}{\text{Precision } + \text{ Recall}} \tag{4.1}$$

where

$$\text{Precision } = \frac{tp}{tp + fp}, \quad \text{Recall } = \frac{tp}{tp + fn}$$

, being tp, fp and fn the number of true positives, false positives and false negatives respectively.

Apart from this accuracy score, we also reported both the absolute and relative time deviations from ground-truth onset annotations and algorithms' inference speed. The absolute time deviation shows us the amount of time imprecision that the algorithms have when placing the evaluation onset, while the relative time deviation illustrates the skewness of these imprecisions. In this way, a positive score in the relative time deviation metric means that onsets are generally predicted after the ground-truth labels, while a negative one means that predictions generally precede the loaction of these ground-truth labels. Inference speed was calculated using an audio file of 120 seconds containing random numbers between -1 and 1. It mainly includes spectrogram calculation, normalisation, division in sequences, prediction, and sequence flattening. The inference was run on a MacBook Pro 2015 with a 2.5 GHz Quad-Core CPU processor.

	Heuristic	CNN	RNN	CRNN
F1-Score	.932	.954 ± .003|.003	.964 ± .002|.001	**.965 ± .003|.002**
Precision	.903	.936 ± .007|.006	**.960 ± .002|.002**	.956 ± .003|.004
Recall	.963	.972 ± .008|.007	.968 ± .003|.002	**.975 ± .001|.001**
Deviation	4.4 ± 8.9	0.6 ± 8.3|7.3	0.7 ± 8.2|7.2	**0.4 ± 8.0|7.0**
Abs. Deviation	8.0 ± 6.9	6.8 ± 6.4|5.6	**6.4 ± 6.4|5.4**	**6.4 ± 6.3|5.5**
Inference Time	**10.6 ± 0.2|0.1**	20.9 ± 0.5|0.2	14.2 ± 0.3|0.1	126.7 ± 6.4|2.8

Table 4-A: Offline accuracies (five iterations for data-driven methods), time deviations (relative and absolute) of predicted onsets from ground truth, and inference speed (twenty iterations). Time deviations are given in milliseconds, and inference speed in milliseconds per analysed second of audio. Error is expressed in terms of standard deviation (left) and 95% confidence intervals (right). Time deviations were added half of the hop size (2.9 ms) to account for measurement uncertainty except for mean relative deviations.

A fixed tolerance window length of 30 ms was used for offline evaluation, meaning that only onsets predicted at a time between -15 ms and +15 ms from the ground truth onsets are considered to be correct. This is a smaller window than the one typically used for musical onset detection (50 ms), as we are dealing with monophonic vocal percussion sounds with generally fast attacks.

4.1.2 Results and Discussion

Results are outlined in table 4-A. The best-performing model was the CRNN, followed by the RNN, the CNN and the best-performing heuristic method. The RNN model exhibits modestly higher stability than the CNN and CRNN, and none of their lower bounds reach the heuristic method's F1-score, meaning that there's a high chance that these data-driven algorithms would consistently outperform heuristic methods for onset detection. A few natural advantages of data-driven methods related to the sources of confusion mentioned in the chapter's introduction can help explain their higher performance, which would also hold true for online onset detection.

The best-performing heuristic method was the MKL divergence. It outperformed both the HFC (F1 = .921) and the complex domain method (F1 = .919), which were observed to be optimal for vocal percussion transcriptions in the LVT and AVP dataset [11, 12]. We hypothesise that the inclusion of beatbox-related utterances specifically could have played a role in the selection of this method over the HFC and the complex domain. We can also see in the table that its precision score is significantly lower than the ones relative to data-driven models, while its recall score is on par with them. This means that the amount of incorrectly predicted onsets (false positives) is far greater for the heuristic method than for the data-driven ones, which was expected due to the sources of confusion outlined in the introduction. This signals the robustness of data-driven algorithms when faced with these sorts of distracting utterances.

We see that the CRNN and RNN methods achieved higher accuracies than the CNN model, which is considered state-of-the-art in musical onset detection. To see if this also translated to the two pretrained models in the literature, an RNN [1] and a CNN [2], we implemented them using the Madmom library and optimised their respective threshold and moving maximum parameters in the same way as for the heuristic approaches. The F1-scores we got from these two algorithms were .902 for the RNN and .941 for the CNN, meaning that our observation of the outperformance of the RNN model is not replicated by pretrained models.

A possible reason for this could be the low frequency resolution of input representations for the pretrained RNN, which are 40 Mel bands and their derivatives from two analysis frames (160 features per data point). These frames, of 23 and 46 ms respectively, had their right edges adjacent to the current time location and extended towards the past. We hypothesise that this way of arranging representations could be counterproductive performance-wise in some contexts, as information from past frames is already available in training sequences and RNNs are able to implicitly take time derivatives into account while training. If true, the RNN model would be losing frequency resolution while not benefitting from it. The pretrained CNN also includes spectral content from different frames, though it has two times the RNN frequency resolution (80 bins), which could explain its higher accuracy.

Performances relative to the RNN and CRNN models are very close to each other, both in terms of accuracy and time deviation from ground truth onsets. Although the CRNN model accounts for a slightly higher F1-Score and lower relative time deviation, its inference time is almost nine times that of the RNN. For example, for a typical vocal percussion performance of 10 seconds, the CRNN would theoretically run inference in 1.27 seconds while the RNN would do it in 0.14[2]. While the choice between one or the other would ultimately depend on the context, we would generally recommend using the RNN model, as it runs significantly faster and achieves almost identical results.

Data-driven approaches also accounted for a lower amount of time deviation from ground-truth labels in both absolute and relative terms. The RNN accounted for the lowest absolute time deviation from ground truth onsets. Although this deviation can statistically reach 10 ms, this amount is still reasonably tolerable in terms of perceived quality of drum transcriptions [13]. Therefore, even for such amounts of deviation, users are not likely to find the final transcription of their performance excessively off-rhythm. The positive magnitude of relative time deviations indicates that predictions are generally skewed to the future. Data-driven methods have the lowest mean results, although the high values of the standard deviations imply that there could be a comparable amount of negative and positive deviation measures around the ground-truth labels.

Finally, all algorithms run their inference significantly faster than real-time. The MKL method was the quickest detector, followed closely by our recommended method, the bidirectional RNN. While these inference times would likely be lowered in product deployment (e.g. optimising speed for sequence building and flattening), we were still surprised by the high inference speed of our deep learning models. We believe lower-level programming frameworks for audio processing are likely to further optimise inference speed as well.

4.2 Online Onset Detection

While offline algorithms have access to the whole audio recording, online algorithms can only process a short analysis buffer containing the most recent few milliseconds of the recorded audio stream. This constraints online models to operate using current and past information exclusively (i.e., in a causal regime), and therefore exposes them to a trade-off between delay and detection accuracy: the longer the analysis buffer, the more information the algorithm can use and the

[2]We run this experiment in particular and got means of 1.39 and 0.23 seconds for CRNN and RNN respectively.

higher the accuracy is expected, but also the more delayed the detection decision will be.

As online VPT aims at triggering drum samples practically at the same time that the user produces the vocalisation, it is important to strategically shorten the length of the analysis buffer (delay) to avoid perceptually unpleasant transcriptions without compromising detection accuracy. Stowell et al. tried to estimate the delay threshold from which human listeners begin to perceive percussive transcription as unpleasant in [14]. They concluded that this happened at around the 46 milliseconds mark of delay, which we took as the upper bound for our algorithms to operate within.

4.2.1 Methodology

This section is divided in the same way as in the offline case. It outlines information about data processing, heuristic algorithms, data-driven algorithms, and training and evaluation routines.

4.2.1.1 Data

The data, the splits, and the preprocessing steps for data-driven approaches were the same as the offline case, except for what concerns time sequences: our data-driven algorithm was trained on sequences of 10 time frames ($\sim$ 58 ms). We saw that performance did not improve substantially from this length on, so we fixed the length there to avoid extra computation delays. The sequences were exactly one time frame away from each other, i.e., they started one time frame later than the previous one.

4.2.1.2 Heuristic Algorithms

We implemented the same heuristic methods as in the offline case applying the adequate causal constraints of the online setting, which the Madmom library had support of. This way, no future-informed processing was carried out in neither the spectrum nor the activation function. We followed the same validation threshold estimation routine as in the offline case except that here we calculated five thresholds instead of one. These five thresholds were used to compute algorithms' accuracies at 1, 2, 3, 4, and 5 frames after the onset frame respectively, which would correspond to delays of 8.7, 14.5, 20.3, 26.1, and 31.9 milliseconds (adding half of the spectrogram's hop size value to account for measurement uncertainty).

4.2.1.3 Data-Driven Algorithms

Apart from heuristic methods, we built our own unidirectional (causal) RNN model based on the pretrained RNN model in [3], which is state-of-the-art in online musical onset detection [4]. We also evaluated the performance of this pretrained model, which was implemented via the Madmom library. We kept our RNN as simple as possible to achieve fast computation speeds without compromising detection accuracy. The final network architecture consisted on a single recurrent layer with 32 Long Short-Term Memory (LSTM) units [15], a dropout layer of rate 0.1, and a dense layer connecting learnt feature maps to the prediction value.

4.2.1.4 Training and Evaluation

We attempted to train our RNN model using an Adam optimisation algorithm, but we found out that it was probably the reason why the model did not generalise well to the test set at small time delays. We hence changed the optimiser to a Stochastic Gradient Descent one [16], which gave closer results to validation accuracies. We also used early stopping, and learning rate downscaling using the same parameters as in the offline case.

In contrast with offline models, our causal RNN calculated a single activation value for each sequence, the ground truth of which was the annotation of the last time frame of the sequence. We trained a total of six RNN models that used different fuzzy ground truth labels. This time, these were simply label dilations of the original onset annotations: the labels associated with the first model had class weights of 1 in the onset frames, the labels associated with the second model would have class weights of 1 in the onset frames and in the frames right after them, the labels associated with the third model would have class weights of 1 in the onset frames and the two frames after them, and so on. We named these models $RNNN - Frame$, where N is the number of frames in the future that the annotation extends to. Decision thresholds relative to activation functions were determined using a seven-fold cross-validation routine and averaging them afterwards as in the offline case.

We also optimised the moving maximum parameters of all onset detectors. As we were in an online regime, we set the windows' extensions from the current point to a future point to zero, but still optimised the length of the moving windows from the current point to a previous point. Finally, we timed the models in the same way as we did for the offline case to check for suitability for real-time processing.

4.2.2 Results and Discussion

Results related to accuracy are illustrated in Figure 4.2. This time, the best-performing heuristic onset detector was the one based on the superflux descriptor. This algorithm was also proven to be the best-performing one within the Madmom library. We hypothesise that the vibrato suppression engine that characterises the detector could explain its high performance, as it might help avoid confusion between onsets from percussive phonemes and those from vowel phonemes within the same syllable. We also observed that this algorithm significantly outperformed the rest of the heuristic ones at 8.7 and 14.5 ms of delay, where the onset vs. non-onset decision is made quicker.

Performances from RNN N-Frame models are generally superior to those from the superflux and pretrained RNN methods and appear to plateau around the 0.9 mark from 14.5 ms delay on. This might mean that there is usually no need for further delaying prediction in most situations so as to gain a little more accuracy. We can also see that, even at the lowest amount of delay (8.7 ms), RNN N-Frame models are able to reach the 0.8 accuracy mark, which tells us that vocal percussion onsets are usually highly recognisable in their very first milliseconds. The standard deviations and 95% confidence intervals of RNN N-Frame models seem small enough to consider them relatively stable to weight initialisation and our cross-validated model selection routine.

We see how the performances relative to the 0-Frame and 1-Frame RNN models fall shorter than the rest of the neural networks. Hence, in an online setting, onset annotations with 2 or

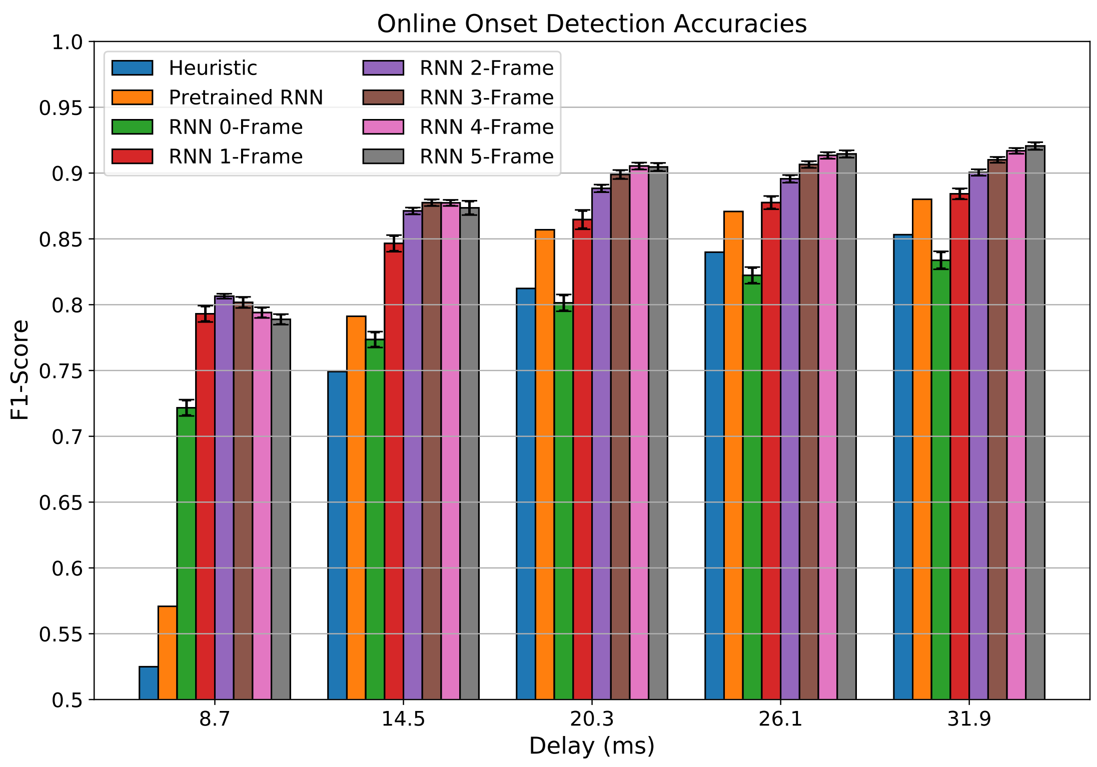


Figure 4.2: Online detection accuracies. Results relative to RNN N-Frame are averaged over their five iterations. Error is given in terms of standard deviation (long cap) and 95% confidence intervals (short cap, hardly seen) for trained RNN.

more frame dilation seem to benefit onset detectors, potentially allowing algorithms to grasp a better understanding of the onsets' characteristics.

Relative time deviations of each algorithm's predictions from ground truth onsets are displayed in Figure 4.3. One observation of interest is that predictions from heuristic and pre-trained RNN algorithms have consistently higher amounts of deviation than trained RNNs (almost double for all delays). A possible reason for this could be the fact that these onset detectors were originally engineered for musical onset detection, where some note attacks are not as well-defined as percussive ones (e.g. a violin legato). Also, the pretrained RNN model functions with a fixed hop size of 10 ms when computing the spectrogram, whereas our trained RNNs use one of 5.8 ms for a higher time resolution. This also explains the significantly lower accuracy score of this model with respect to the trained RNNs at 8.7 ms delay.

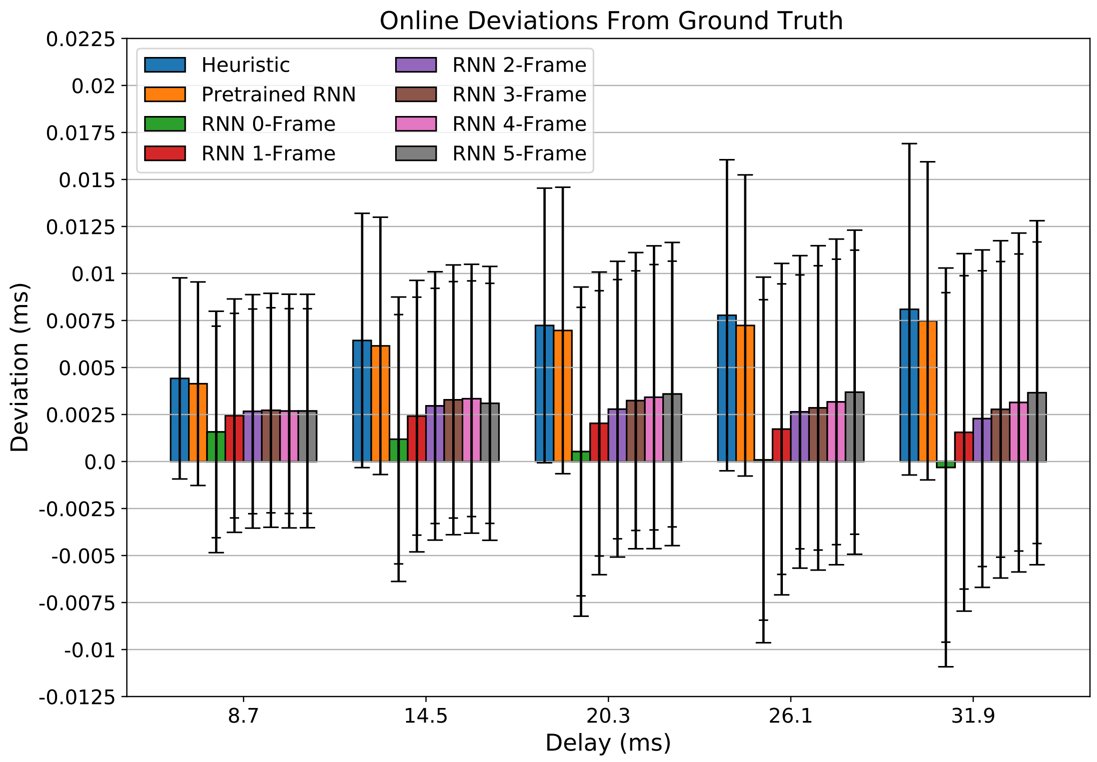


Figure 4.3: Relative time deviations of algorithms' predictions from ground truth onsets. Deviations relative to RNN N-Frame models are averaged over their five iterations. Error is given in terms of standard deviation (long cap) and 95% confidence intervals (short cap) for data-driven methods. Errors were added half of the hop size (2.9 ms) to account for measurement uncertainty.

Predictions from the RNN 0-Frame model deviated less than those from the rest of the methods, although it also had the highest standard deviations for all delays. Also, relative deviations were mostly positive, meaning that algorithms usually placed the onset somewhere after the ground truth annotations. Deviations from our trained RNNs usually do not reach the 3 ms mark in terms of mean value and 12/-10 ms in terms of 95% confidence intervals. These results look certainly promising for online VPT, as the classification process after onset detection usually works with analysis frames of 23 ms on [14, 17]. In this sense, just by (i) correcting onset placement using the mean relative time deviations and (ii) making sure that the 95% confidence intervals of relative time deviations fall in a region where classification accuracy is high and delay is low, would theoretically suffice to achieve optimum transcription performance. Finally, the RNN N-Frame models accounted for the lowest mean inference time (7.9 $\pm$ 0.3|0.1 ms) compared to those of the superflux algorithm (12.0 ± 0.2|0.1 ms) and the pretrained RNN (25.5 ± 2.1|0.9 ms), further supporting its suitability for real-time processing.

Apart from the algorithms discussed above, we explored two other online onset detection methods that were ultimately less successful. One of these was a stateful RNN that processed data frame by frame. Its results were slightly better than the rest of the algorithms for delay 8.7, with a mean F1-score of .816, but did not get past the .854 mark for the rest of the delays. We also implemented a CRNN for real-time processing. This way, convolutional layers compressed the spectrum of individual frames using 1-dimensional convolutions and those features were processed by the RNN module. This method achieved indistinguishable results from those of RNNs at the expense of significantly higher computation time.

4.3 Summary

We studied the suitability of several data-driven algorithms for vocal percussion onset detection and contrasted their performances with their heuristic counterparts. The network models we proposed consistently outperformed heuristic algorithms and pretrained models in both offline and online onset detection in terms of accuracy, time deviation from ground truth onsets, and also inference speed in the online case.

For offline onset detection, we saw that CRNNs and bidirectional RNNs gave the best performance accuracies and that the latter had a significantly lower inference time than the former. Hence, we would generally recommend bidirectional RNNs for offline processing, although the choice between one or the other would ultimately depend on the transcription context.

For online onset detection, the RNN N-Frame models were the fastest and most accurate of the evaluated methods, reaching an F1-score of 0.9 at 20.3 ms of delay. An accuracy score of 0.8 is already reached for 8.7 ms of delay, meaning that vocal percussion onsets are reasonably recognisable at very low delays. The final choice of what RNN N-Frame onset detector to use for online processing is contingent on the accuracy vs. delay trade-off of subsequent classifiers, addressed in chapter 6.

References

[1] F. Eyben, S. Böck, B. Schuller, and A. Graves, "Universal onset detection with bidirectional long-short term memory neural networks," in *Proc. 11th Intern. Soc. for Music Information Retrieval Conference, ISMIR, Utrecht, The Netherlands*, 2010, pp. 589–594.

[2] J. Schlüter and S. Böck, "Improved musical onset detection with convolutional neural networks," in *2014 ieee international conference on acoustics, speech and signal processing (icassp)*. Ieee, 2014, pp. 6979–6983.

[3] S. Böck, A. Arzt, F. Krebs, and M. Schedl, "Online real-time onset detection with recurrent neural networks," in *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx-12), York, UK*. sn, 2012.

[4] S. Böck, F. Krebs, and M. Schedl, "Evaluating the online capabilities of onset detection methods." in *Ismir*. Citeseer, 2012, pp. 49–54.

[5] M. Müller, *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015.

[6] D. Stowell and M. Plumbley, "Adaptive whitening for improved real-time audio onset detection," in *Proceedings of the 2007 International Computer Music Conference, ICMC 2007*. University of Surrey, 2007, pp. 312–319.

[7] P. M. Brossier, "Automatic annotation of musical audio for interactive applications," Ph.D. dissertation, 2006.

[8] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "Madmom: A new python audio and music signal processing library," in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 1174–1178.

[9] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[11] A. F. S. Ramires, "Automatic transcription of vocalized percussion," 2017.

[12] A. Delgado, S. McDonald, N. Xu, and M. Sandler, "A new dataset for amateur vocal percussion analysis," in *Proceedings of the 14th International Audio Mostly Conference: A*

Journey in Sound, 2019, pp. 17–23.

[13] J. Frühauf, R. Kopiez, and F. Platz, "Music on the timing grid: The influence of micro-timing on the perceived groove quality of a simple drum pattern performance," *Musicae Scientiae*, vol. 17, no. 2, pp. 246–260, 2013.

[14] D. Stowell and M. D. Plumbley, "Delayed decision-making in real-time beatbox percussion classification," *Journal of New Music Research*, vol. 39, no. 3, pp. 203–213, 2010.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[17] A. Delgado, C. Saitis, and M. Sandler, "Phoneme mappings for online vocal percussion transcription," in *Audio Engineering Society Convention 151*. Audio Engineering Society, 2021.

Chapter 5

Offline Vocal Percussion Classification

The present chapter focuses on vocal percussion classification in an offline, non-real-time context. In contrast with online classification, which will be later addressed in Chapter 6, here we have access to the whole audio file containing the vocal percussion performance. Therefore, we expect higher classification accuracies than those relative to an online setting a priori, which would only have access to the very beginning of vocal percussion sounds.

Offline vocal percussion classification is usually carried out by (i) applying onset-wise segmentation to the audio file assuming that vocal percussion sounds lie in-between onsets (ii) extracting or learning relevant audio features from each of the individual segments in a frame-wise manner, and (iii) training a machine learning algorithm to classify the vocal percussion sounds in the audio segments given the input features. While one could take advantage of the information in the rest of the audio file, e.g., rhythmic patterns, here we exclusively focus on classifying individual sounds (regions in-between onsets) irrespective of the information contained in the rest of the audio file.

We placed a high emphasis on user-based amateur vocal percussion classification [1, 2], where we explored different data-driven strategies. As discussed in earlier chapters, user-based amateur vocal percussion classification is a generally challenging task, and we specifically see three main points of difficulty that condition the design of algorithmic routines.

The first of these issues is the different vocal percussion styles among participants. While beatbox performers share a common set of techniques that makes their vocalisations sound similar to each other, amateur performers approach vocal percussion via vocal imitation and they usually have distinct ways to vocalise drum instruments.

The second point of difficulty is that participants usually do not label the sounds they record for the algorithm to train successfully, so the labels have to be made implicit in some way. In order to sort this out, amateur vocal percussion datasets are recorded in such a way that training annotations are made implicit for the classification algorithms. As introduced in 1.1, the "fixed phrase" and the "isolated samples" strategies are successful recording techniques that automatically label training samples without the need of having the users do it manually. However, depending on the way these training samples are recorded, they could end up not resembling the testing ones enough and therefore fail to generalise to improvisatory performances.

The third and last issue is the inconsistency in the articulation of vocal percussion sounds by users. This issue is especially prominent in amateur vocal percussion improvisations, which are the kind of performances that are used for evaluation. The lack of uniformity in the sounds' timbre across these evaluation samples could sometimes confuse classifiers which, if trained in a user-based manner, might experience a significant drop in performance due to the low number of samples that they learn from. Regularisation techniques for data-driven models like data augmentation, batch normalisation, dropout, and early stopping are likely to mitigate the effect of this issue to some extent.

This chapter details the classification routines and evaluation strategies we followed for both amateur vocal percussion and beatbox. It is composed of two related studies. The first one introduces the different types of datasets available for offline vocal percussion classification and explores the performance of different heuristic and end-to-end data-driven algorithms. The second set of experiments is directed to improve performances of user-based amateur vocal percussion via supervised embedding learning. This embedding learning strategy is based on the training of data-driven network algorithms using labels that describe vocal percussion sounds at different levels of abstraction, namely instrument-level, syllable-level, phoneme-level, and sound-level.

In summary, here we wanted to answer the following research questions:

- Which classification strategy is most adequate to carry out offline classification with amateur vocal percussion and beatbox sounds?
- How do data-driven algorithms perform in low data regimes such as purely user-based scenarios?
- Is embedding learning preferable to end-to-end offline classification?
- To what degree does setting a fixed way of vocalising sounds help with offline amateur vocal percussion classification? How does its performance compare to using sounds of the participants' choice?
- How fast is data-driven models' inference compared to that of heuristic methods?
- How stable to arbitrary training parameters are the performances of heuristic and data-driven classifiers?

5.1 End-to-End Offline Vocal Percussion Classification

The first study that we carried out for offline vocal percussion classification explored *end-to-end* data-driven models, comparing their performances with those of baseline heuristic approaches. In essence, the main purpose of this study was to assess the performance of data-driven classifiers in scenarios with different input data and see ways of improvement in case these accuracies fall short from expectations.

5.1.1 Methodology

In this section, we present the methodology that we followed in this first study. We talk about the data with which we train and evaluate the studied algorithms, the classification tasks that we set up based on different strategies and arrangements of the data, the heuristic and data-driven classification algorithms that we built for the tasks, and the training and evaluation routines.

5.1.1.1 Data and Tasks

We compared purely user-based approaches and heuristic approaches on three main tasks: user-based amateur vocal percussion classification (AVP UB), user-agnostic amateur vocal percussion classification (AVP UA), and user-agnostic classification with the BTX dataset (BTX UA).

In the AVP UB task, algorithms were trained in a user-based manner, i.e., each algorithm modelled and predicted the label of the sounds pertaining to a single participant. Therefore, we would have as many trained algorithms as users in the end, which were trained using the personal subset of the AVP dataset. As discussed earlier, this task is especially challenging from a classification perspective, as users were also allowed to choose the phonemes or syllables that they wanted to use to trigger the sounds relative to each drum type. Results relative to the AVP UB task are expressed in both a participant-wise (AVP UB-Part) and a sound-wise (AVP UB-Sound) manner. In the case of the former, accuracy is averaged across participants while, in the latter's case, accuracy is averaged across sounds irrespectively of the participant or the drum type they belonged to.

In the AVP UA task, algorithms were trained in a user-agnostic manner, i.e., a single algorithm modelled and predicted the label of all sounds irrespectively of the participants they belonged to. We tried three different approaches in this case, each of them classifying different types of sounds.

The first approach, AVP UA, consisted in training the classifier in the personal subset of the AVP dataset. As here participants selected the sounds that they vocalised, we expect a lower accuracy score with respect to the user-based case above, which takes information from single participants into account. Hence, this method would also serve as a baseline performance to assess the degree of success of the user-based strategy.

The second approach, AVP UA Syll, attempts at classifying the sounds pertaining to the fixed set of the AVP dataset, which are represented by the syllables /pm/, /ta/, /ti/, and /tʃi/.

Although the four syllables could be used to trigger any drum sound, the originally recommended mappings in the AVP dataset are /pm/ for kick drum, /ta/ for snare drum, /ti/ for closed hi-hat, and /tʃi/ for opened hi-hat.

Finally, in the third approach, AVP UA Phon, algorithms classify the sounds pertaining to the phonemes /p/, /k/, /t/, /ts/, and /tʃ/. The choice of these phonemes for VPT is explained by their high frequency of use among performers, their high spectral similarity with drum instruments (kick, snare, and hi-hat), and their high classification separability. These three properties of the phonemes above are experimentally justified later in this thesis, more specifically in Section 6.1. Also, we called the task "AVP UA Phon" because of the use of single phonemes in the case of /p/, /k/, /t/, despite also containing the /ts/, and /tʃ/ syllables.

Lastly, the BTX UA task also consisted of a user-agnostic classification process like AVP UA but using the sounds in the BTX dataset instead. The BTX dataset was annotated under the assumption that participants' beatbox techniques were similar enough to each other to study them jointly. The dataset also consisted of beatbox performances exclusively, without specific subsets for user-based training and evaluation.

The data for all these tasks and methods was pre-processed by a 14-fold data augmentation routine on both waveforms and spectrograms. In this way, we applied different data augmentation methods in random order for 14 iterations. For the first waveform-based augmentation routine, we applied random *pitch-shifting* (semitone range = [-1.5,+1.5]), *time-stretching* (stretch factor range = [0.8,1.2]), and *Gaussian noise* (minimum amplitude = 0.001, maximum amplitude = 0.05, probability of application = 0.5) one after the other in random order. For the second spectrogram-based augmentation routine, we used a spectral *frequency mask* (minimum mask fraction = 0.03, maximum mask fraction = 0.25, probability of application = 0.75) [3].

5.1.1.2 Algorithms

We gathered a set of 262 *heuristic* features to which we applied dimensionality reduction via Principal Component Analysis (PCA). The audio descriptors associated with these features were mainly chosen for their relevance and the good performances in VPT-related literature (see Section 2.2.1.2). They were also chosen attending to their relevance in heuristic Sound Event Detection (see [4]) and other related tasks like musical instrument classification and timbre analysis [5]. We believe these good performances in both identical and analogous tasks enough to justify their inclusion in our heuristic VPT analysis pipeline.

In this way, the final features were based in the following audio descriptors: 13 Mel Frequency Cepstral Coefficients (MFCCs), spectral energy of 8 frequency bands (0-300, 300-800, 800-1600, 1600-4000, 4000-7000, 7000-11000, 11000-16000, and 16000-22050 Hz), 4 spectral roll-off frequencies (ratios of 0.25, 0.50, 0.90, and 0.95), spectral complexity, high frequency content, spectral strongpeak, 4 spectral central moments (centroid, variance, skewness, and kurtosis), spectral crest, spectral decrease, spectral entropy, spectral flatness, root mean square, and zero-crossing rate. We extracted these audio descriptors using the Essentia toolbox [6] and aggregated them in time using the mean, variance, minimum, and maximum values of these and their first derivative; hence, a total of 8 aggregators per descriptor except for the MFCCs and the spectral energy bands, which were taken the mean value and variance from themselves and their first and second derivatives in the case of MFCCs and just the mean in the case of the spectral bands. Lastly, we also extracted four envelope descriptors which included the derivative after the maximum amplitude, the maximum derivative before the maximum amplitude, the flatness coefficient, and the temporal centroid to total length ratio.

We applied ten iterations of randomised PCA [7], each with a different random state, to these 262 features so as to reduce their dimensionality to 32 components. These components were later fed to the final classification task, where we explored the performance of seven different machine learning classifiers: five KNN algorithms with 3, 5, 7, 9, and 11 neighbours respectively, a random forest algorithm (best-performing model in [8]), and a logistic regression model. We report the best performance derived from these models, which were all implemented using the Scikit-Learn library [9].

Apart from the heuristic routine above, we used three different CNN architectures for the AVP UB, the AVP UA, and BTX UA tasks respectively. In the case of the AVP UB task, with an average of 115 training samples per user (1725 augmented), the CNN model was kept *small-sized* in order to decrease the number of parameters and hence the chance of overfitting. It took Mel spectrograms with dimensions of 16 bands by 12 frames calculated using a window size of 96 ms and a hop size 46 ms, accounting for a total of 0.55 seconds per sound. The CNN was composed of two convolutional blocks, both with a single convolutional layer of kernel size 3x3 and strides 1x1, a batch normalisation layer, a ReLU activation gate, and a max-pooling operation. The convolutional layers in the first and second block had 16 and 32 filters respectively, and the max-pooling kernels of these had a size of 2x2 and 2x3 respectively.

The CNN model for the AVP UA Phon task, with 2,226 training samples (33,390 augmented), was *medium-sized*, taking as input 64x48 Mel spectrograms calculated using a window size of 96 ms and a hop size 11 ms (0.55 seconds per sound). This CNN was composed of four convolutional blocks like the ones for AVP UB. The convolutional layers in the first, second, third, and fourth blocks had 8, 16, 32, and 64 filters respectively, and the max-pooling kernels of these had all sizes of 2x2.

Finally, the CNN model for the AVP UA, AVP UA Syll, and BTX UA tasks, with 3,227 (48,405 augmented), 3,320 (49,800 augmented), 3,611 (54,165 augmented) training samples respectively, was *large-sized*, and it took the same input representation and had the same architecture as the previous medium-sized model but with double layers in each block except for the max-pooling one. In this way, for example, the first convolutional block had one convolutional layer with 8 filters, a batch normalisation layer, a ReLU activation gate, another convolutional layer with 8 filters, another batch normalisation layer, another ReLU activation gate, and the max-pooling operation at the end.

The MLP models were also built considering the size of the dataset to be modelled. In this way, the model relative to the AVP UB task had two hidden layers of 32 neurons each, the model relative to the AVP UA Phon had two hidden layers with 64 neurons each, and the model relative to the AVP UA, AVP UA Syll, and BTX UA tasks had three hidden layers of 64 neurons each. We test two types of MLP-based approaches, the MLP-Heur and the MLP-Spec, with the former models taking the 32 PCA component features as input and the latter taking the flattened version of a 16x12 spectrogram as input, which was calculated with a window size of 96 ms and a hop size 46 ms (0.55 seconds per sound).

5.1.1.3 Training and Evaluation

We carried out the train-validation routines of the AVP UB task with the sounds provided for training and the evaluation routines using the sounds provided for testing. The final train-evaluation split was approximately 60-40%. For the AVP UA task, we carried out the train and validation routines with the sounds provided for training and some of the sounds provided for testing in order to get a 75-25% train-evaluation split. We reserved the remaining testing sounds for evaluation. Lastly, both the train-validation and the evaluation routines relative to the BTX UA task were built with the sounds in the BTX dataset with a 75-25% train-evaluation split.

CNN and MLP models were trained using an Adam optimisation algorithm, early stopping if the validation losses did not decrease after 20 epochs, and learning rate downscaling if validation loss did not decrease after 10 epochs. We also ensured that all train-validation-evaluation splits had the same percentages of sound labels.

For each of the three types of tasks, AVP UB, AVP UA, and BTX UA, we averaged results through the ten train-test iterations, taking their standard deviation and their 95% confidence interval as error measures. For the AVP UB task, we reported final results in two modalities: *participant-wise*, where accuracy scores of single test participants are calculated and averaged, and *sample-wise*, where accuracy scores are calculated for all evaluation samples independently of the participants they belong to.

In the case of AVP UB models, training times were measured and averaged across the 10 iterations of the 28 participants. These training times would give us an idea of how long would users have to wait for the algorithm to be trained on the examples they just provided before being ready to predict samples within vocal percussion performances. These timing experiments were performed on the same machine (MacBook Pro 2015 laptop) and under the same conditions.

	Heuristic		End-To-End	
	KNN-Heur	MLP-Heur	MLP-Spec	CNN
AVP UB-Part	.730 ± .014|.008	.714 ± .031|.019	.779 ± .051|.031	**.789 ± .031|.019**
AVP UB-Sound	.731 ± .014|.009	.708 ± .026|.016	.762 ± .053|.032	**.770 ± .030|.019**
AVP UA	.578 ± .004|.003	.623 ± .012|.008	.644 ± .010|.006	**.692 ± .012|.007**
AVP UA Syll	.815 ± .003|.002	.859 ± .008|.005	.887 ± .006|.004	**.916 ± .005|.003**
AVP UA Phon	.834 ± .008|.005	.715 ± .012|.007	.820 ± .012|.008	**.970 ± .009|.006**
BTX UA	.744 ± .008|.005	.669 ± .012|.007	.742 ± .019|.012	**.832 ± .013|.008**

Table 5-A: Offline amateur vocal percussion and beatbox classification accuracies.

5.1.2 Results and Discussion

Final results are presented in table 5-A, with the best-performing heuristic classifier being KNN with 3 neighbours.

We see how the end-to-end data-driven models (MLP-Spec and CNN) consistently outperform heuristic approaches (KNN-Heur and MLP-Heur), and that the CNN method in particular is the best-performing model for all tasks. In particular, the difference in performance of the CNN method with respect to the other methods is significant for the AVP UA Phon and BTX UA one, and slightly more modest for the AVP UB, AVP UA, AVP UA Syll ones. The KNN-Heur and the MLP-Spec methods perform better than each other depending on the task, with the former achieving better accuracies overall. This shows how simple non-parametric models like a KNN classifier can achieve better accuracies than data-driven models with a large number of parameters like MLPs when taking heuristic features as input, i.e., when not trained end-to-end. When trained end-to-end with input spectrograms, as is the case with the MLP-Spec approach, we can see how the performance is consistently superior to both heuristic methods.

Heuristic strategies were generally the most stable to random parameters than end-to-end methods. In that manner, one can also note the significantly higher stability of heuristic and CNN methods in user-agnostic (UA) tasks than user-based (UB) ones. This could be explained by the fact that UA tasks are single classification tasks using a large number of samples (>2,000 unaugmented) whereas UB ones are 28-way classification tasks, one per participant, using a notably smaller amount of data per subtask (<150 unaugmented), which potentially leads to higher variance in performance over the ten iterations.

For the AVP UB case, we got average training times of 5.29, 23.41, 22.80, and 177.61 seconds for the KNN-Heur (PCA included), MLP-Heur (PCA included), MLP-Spec, and CNN. Hence, the CNN model, while being the most accurate one, trains significantly more slowly than the rest of them, almost reaching the 3-minute mark. The inference times of all methods were similar to each other, taking around 50 milliseconds per predicted label. Therefore, if waiting for training to finish is not an issue, the user could use the end-to-end CNN algorithm to classify sounds as fast as the other methods and with higher accuracy. Nevertheless, as we wrote in Section 4.1.2 regarding the RNN and CRNN onset detectors, we would recommend the use of the MLP-Spec algorithm instead of the CNN one in practice, as the former achieves similar accuracies to the latter at a much lower training time (more than 10 times faster).

We have also demonstrated our hypothesis that the user-based method (AVP UB) performs significantly better than the user-agnostic one (AVP UA) when participants vocalise the sounds that they like for each of the instruments. The user-based method shows a superior performance via its best-performing algorithm (CNN) and the best score from its related models (.708 for MLP-Heur) is still higher than the lowest score from the related models of the user-agnostic method (.692 for CNN). However, the highest user-based performance (.770 sound-wise for CNN) is still notably poor with respect to the performance of the AVP UA Syll and AVP UA Phon methods, which might mean that user-agnostic strategies are still better approaches than user-based strategies when choosing specific fixed input phonemes and syllables, as AVP UA Phon achieves almost perfect accuracies (.970 for CNN).

We can also see how performances related to the AVP UA Syll task are better than AVP UA Phon for some algorithms (MLP-Heur and MLP-Spec) and lower for others (KNN-Heur and CNN). There are several reasons that might explain this result. One of them could be related to the fact that the number of training parameters was possibly too large for medium-sized MLPs to model the input data of the AVP UA Phon task or for large-sized CNNs to model that of the AVP UA Syll task. This may have resulted in some degree of overfitting for both cases. Another thing that may have affected performance in both tasks is the fact that for the AVP UA Phon method a small part of sounds taken from improvisation performances, which were supposed to be used for evaluation, were used for training and validation instead. While none of these sounds nor their augmented versions were used for evaluation, algorithms could learn features that are particular to these sounds in the improvisation performances and use them for evaluation. This does not occur in the case of the AVP UA Syll method, where algorithms are trained using the training set (isolated samples) exclusively and evaluated on the testing set (improvisation), and therefore may have been at disadvantage with respect to the AVP UA Phon one, especially considering successful end-to-end methods like the CNN.

One issue we noted regarding evaluation was that, in the case of the AVP UA Phon task, the dataset had notably less samples relative to the /k/ phoneme (262) compared to the amounts relative to the /p/ (532), /t/ (608), /ts/ (658), and /tʃ/ (723). This is concerning a priori, as most of the mistakes that the classifiers commit in the evaluation stage could be related to the /k/ phoneme, as it has the least amount of samples and algorithms could neglect it in favour of higher accuracies. However, we checked the confusion matrices of all ten iterations of the CNN algorithm and the models consistently got all of the samples relative to the /k/ phonemes correct, which included a total of 80, 39, 91, 99, and 109 samples for /p/, /k/, /t/, /ts/, and /tʃ/ respectively. We mostly attribute this finding to both the high separability of the /k/ phoneme with respect to the rest of them and to the stratified train-validation-evaluation splits that had the same percentage of labels.

Lastly, we consider results related to the BTX UA task underwhelming and pointing towards the difficulty of the task in itself. More research is warranted to assess to what extent individual beatbox labels are separable between each other in a classification task and to weigh the need of recommending specific beatbox sounds to improve accuracies.

5.2 Representation Learning for Offline Amateur Vocal Percussion Classification

In the last section, we argued that the low number of data samples that user-based algorithms take as input is an important issue to consider when trying to apply data-driven methods to the task, as it limits the amount of input audio features that classifiers can take for modelling so as to minimise their risk of overfitting. On the other hand, as we saw in the results from the previous study, data-driven models like MLPs and CNNs outperform heuristic approaches despite taking significantly more time to train, especially in the case of CNNs.

This section is an attempt to mitigate the tendency to overfitting and the long training times of data-driven methods via deep representation learning. In particular, we explore the potential of several *supervised embedding learning* approaches to generate informative feature sets for amateur vocal percussion classification. These embedding learning models have been proven to be powerful feature extractors for high-dimensional data including sound events, music, and speech [10], so we hypothesise that they could likewise be beneficial for our vocal percussion classification task.

We supervised deep neural networks on four different types of label sets and took the values in their penultimate layer as the final feature sets to be evaluated. The four label sets that supervised the algorithms describe vocal percussion sounds at different levels of abstraction, namely at instrument-level, syllable-level, phoneme-level, and sound-level. We assessed the informative power of each of the learnt feature sets in terms of their classification accuracy and the stability of the metric to random train-validation splitting and initialisation routines. Finally, we carried out a complementary investigation of how relevant different regions in the spectrogram are for the models by applying backpropagation-based saliency maps [11]. The related code is published in an open-source repository[1].

[1]*https://github.com/alejandrodl/vocal-percussion-transcription*

	AVP Dataset	LVT Dataset
Number of Participants	28	20
Number of Sounds	4,873	841
Recording Strategy	Isolated Samples	Fixed Phrase
Instrument Labels	kd, sd, hhc, hho	kd, sd, hhc
Phoneme Labels	Yes	Yes

Table 5-B: Summary of datasets' contents (kd = kick drum, sd = snare drum, hhc = closed hi-hat, hho = opened hi-hat).

5.2.1 Methodology

In this section, we provide an in-detail account of the main data sources, algorithms, and routines used throughout our study. In section 5.2.1.1, we talk about the two datasets that we used (AVP and LVT), how we joined them and expanded their annotations so as to include phonetic information, and how we built input representations and carried out the data augmentation process. In section 5.2.1.2, we describe the architecture of the embedding learning model and the seven types of label sets that we used for its supervision. In section 5.2.1.3, we present the three baseline methods whose performances were compared to those of the embedding learning model supervised on different label sets and in section 5.2.1.4 we show how both the training and the evaluation processes were carried out.

5.2.1.1 Data and Pre-Processing

We used two publicly available vocal percussion datasets throughout the study: the AVP dataset [12] and the LVT dataset [1]. We contrast some of these datasets' characteristics in Table 5-B. To train our acoustic models, we exclusively used the personal subset of the AVP dataset (participants vocalising sounds of their choice), although we also used the fixed subset (participants vocalising the same sounds) to train the sequential module of the baseline speech recognition model (see section 5.2.1.3). Regarding the LVT dataset we exclusively use its third subset, as the recordings' quality and background noise level are similar to those from the AVP dataset.

We manually expanded the annotations (onsets and instrument labels) of both datasets so as

Onset Phonemes	**Coda Phonemes**
/t/ and /!/	/ɑ/, /æ/, /ɐ/, and /ʌ/
/ts/ and /s/	/e/, /œ/, and /ə/
/tʃ/, /tɕ/, /dʒ/, and /tʒ/	/i/, /y/, and /ɪ/
/kx/, /k/, and /kʃ/	/o/ and /ʊ/
/p/ and /ʡʢ/	/u/ and /ɯ/

Table 5-C: Phoneme groupings for the reduced sets.

to include the syllabic representation of vocal percussion sounds. The syllables were composed of a first *onset phoneme*, usually plosive or fricative, and a second *coda phoeneme*, usually a vowel, a breath sound, or silence (no coda phoneme). These phonemes were annotated following notation conventions from the International Phonetic Alphabet (IPA). Apart from the *original* phoneme set, we also elaborated a *reduced* phoneme set in which several similar-sounding phonemes were put together to form single classes. For this reduced version of phoneme annotations, onset and coda phonemes were grouped as shown in Table 5-C. We also make final AVP-LVT dataset with expanded annotations publicly available[2].

Joining the AVP and LVT datasets, we had a total of 5714 vocal percussion sounds. As this amount of data was relatively modest for deep embedding modelling, we applied *waveform data augmentation* to individual sounds, specifically random *pitch-shifting* (semitone range = [-1.5,+1.5]) and *time-stretching* (stretch factor range = [0.8,1.2]), one after the other in random order. This kind of data augmentation is standard in audio signal processing [13] and it has been proven to improve the accuracy of vocal percussion classification algorithms [8]. We applied ten iterations of random data augmentation in this manner, ending up with a total of 62854 vocal percussion sounds in the final dataset.

As input to neural networks, we built Mel spectrogram representations from each vocal percussion sound using 64 Mel frequency bands and a hop size of 12 milliseconds. We used 48 time steps ($\sim$ 0.56 seconds) so that the final sound spectrograms had a final dimension of 64x48 and we explored frame sizes of 23, 46, and 93 milliseconds, ultimately reporting the one that brought the best results in Section 5.2.1.1. We post-processed spectrograms with a logarithmic

[2]*https://zenodo.org/record/5578744#.Yfpu9PXP30o*

transform ($log(spectogram + 0.0001)$) and normalised them to a [0,1] range.

5.2.1.2 Supervised Embedding Learning

We used the penultimate layers of several CNN classifier models [14] as the final embeddings to perform evaluation. They all had four convolutional blocks with 8, 16, 32, and 64 filters respectively and two fully-connected (FC) linear layers, one connecting the flattened feature maps to the embedding space and another one connecting the embedding space to the labels. Each convolutional block had two convolutional layers with kernels of size 3x3 and stride 1x1, each one followed by a batch normalisation module and a ReLU activation gate. A final max-pooling operator with kernel size 2x2 is applied at the end of each convolutional block so as to progressively downsample the feature maps.

We explored seven different supervision strategies to train and validate the above-mentioned CNN classifiers. These strategies use different types of labels that describe the same input data at different levels of abstraction.

Instrument-Level Annotations: We used two types of drum instrument annotations: the ones relative to the *original* set (kick drum, snare drum, closed hi-hat, and opened hi-hat) and the ones relative to a *reduced* version of it (drum and hi-hat). These constituted the first and the second supervision strategies.

Syllable-Level Annotations: We also used syllable labels, put together by joining the onset and coda phonemes in the *original* and the *reduced* phoneme sets (see table 5-C). These constituted the third and the fourth supervision strategies.

Phoneme-Level Annotations: We used individual phoneme labels, which also came from the *original* and the *reduced* phoneme sets. These constituted the fifth and the sixth supervision strategies. It is worth noting that, while the two phoneme-level sets contained the same information as the two syllable-level ones, here the CNN classifier predicts onset and coda phoneme labels separately in a multi-task way, managing two different validation losses and accuracies.

Sound-level Annotations: Finally, we used sound labels to describe vocal percussion sounds at the lowest level of abstraction. These sound classes were integrated by sounds that had different associated syllables and also pertained to different participants. This constitutes the seventh supervision strategy.

5.2.1.3 Baseline Algorithms

We compare the performance of learnt embeddings with two baseline feature sets and a speech recognition model.

The first baseline model is the *timbre feature set*, which is made of Mel Frequency Cepstral Coefficients (MFCC) and envelope features. The MFCCs are computed using the same frame-wise analysis parameters as for the spectrogram representations (see 5.1.1.2). For the final 32-dimensional feature vector (see 5.2.1.4), we take the mean of the first 14 MFCCs, the mean of their first derivative, and four envelope descriptors: the derivative after the maximum amplitude, the maximum derivative before the maximum amplitude, the flatness coefficient, and the temporal centroid to total length ratio. For the 16-dimensional feature vector, we take the mean of the first 12 MFCCs along with the four envelope descriptors.

In a similar way as with the embedding learning methods, we built seven importance-based *feature selection* algorithms whose base algorithms were supervised using the same seven supervision strategies described in Section 5.2.1.2. We only report the best result from these seven approaches in Section 5.2.2. Here, instead of learning embeddings from spectrogram representations, we extracted a set of 258 engineered features from the spectrum and the envelope of vocal percussion sounds, derived feature importances from a random forest base algorithm through a 10-way feature permutation process [15], and selected the most informative features to build the final set. In the case of phoneme-level supervision strategies, the final selected features were drawn from the intersection of the two independent feature importances arrays, gathering the features that were considered most important to classify both onset and coda phonemes. We use the Essentia toolbox [6] to calculate these audio descriptors and the ones mentioned in the paragraph above.

For the last baseline method, we tackled vocal percussion classification by means of a *GMM-*

HMM speech recognition system, an approach proposed by Evain et al. in [16] that was originally applied to beatbox classification with satisfactory results and that we bring to amateur vocal percussion classification in order to provide context for our methods. Here, each instrument type to detect is considered as a word to be recognized. First, an *acoustic model* was trained to learn a relationship between acoustic speech features and phonemes. These features were 13-band MFCCs computed at 16 kHz sampling rate and using a frame length of 25 ms. The mapping between instrument types and their corresponding phonemes was established through a pronunciation dictionary, converting phoneme posterior probabilities to word type probabilities. We exploited both instrument- and phoneme-level annotations to construct this pronunciation dictionary. Word probabilities were then smoothed using a language model so as to obtain grammatically sensible transcriptions. The *language model* was 5-gram trained on the transcriptions from training data, and the recognizer was trained via the Kaldi GMM-HMM recipe [17] using the code in *https://github.com/emirdemirel/ALTA* [18]. Our final model was a triphone GMM-HMM model trained with speaker adaptive features [19]. Hyperparameters were determined empirically and they are available in the paper's repository.

5.2.1.4 Training and Evaluation

A schematic diagram of the training and evaluation processes is provided in figure 5.1.

We chose four participants from the AVP dataset and four from the LVT dataset to compose the evaluation set. Two women and two men per dataset were selected for evaluation on the basis of acceptable pronunciation and overall representativeness of the dataset. In the end, we had a total of 8 participants in the evaluation set and 40 participants in the train-validation set. We provide the distribution of the instrument, syllable, onset phoneme, and coda phoneme labels in the project's code repository.

We trained our seven CNN models (see section 5.2.1.2) using the train-validation dataset. We set up a 5-fold cross-validation routine in which models were trained and validated for 5 iterations per fold, each with different initialisation parameters. Therefore, we end up with 25 deep embedding models per supervision strategy that encapsulate two main sources of training

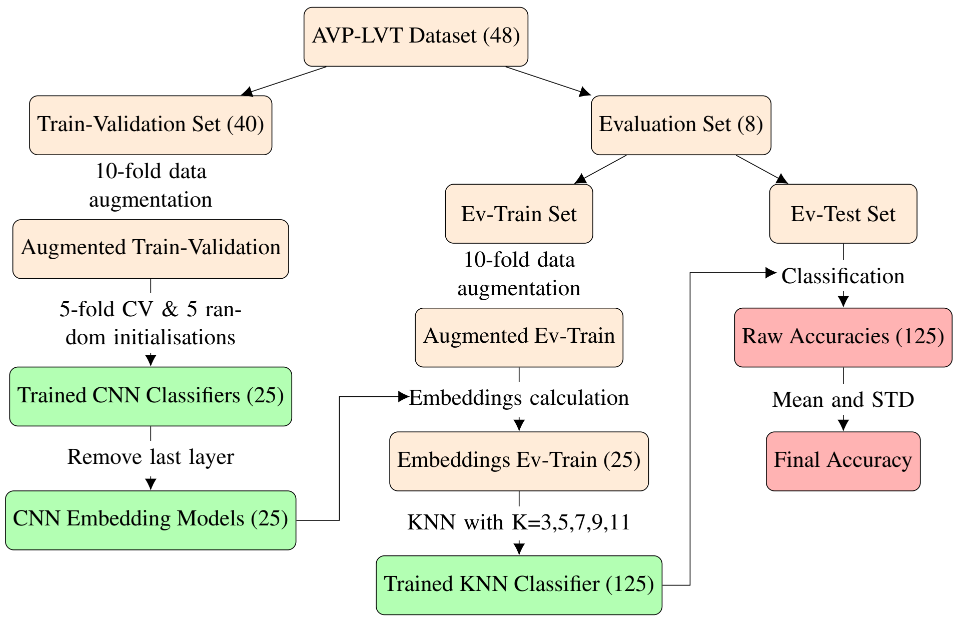


Figure 5.1: Diagram of the training-evaluation process. Background colour code: orange = dataset-related, green = model-related, red = results-related.

arbitrariness: the content of train-validation splits and random weight initialisation. We compute the mean and standard deviation of the subsequent 25 evaluation performances when reporting final results for a given supervision strategy. The same train-validation arrangement applies to the seven baseline feature selection methods. We trained the models using an Adam optimisation algorithm [20], early stopping if validation loss has not decreased after 10 epochs, and a further regularisation routine that downscales the learning rate if validation loss has not decreased after 5 epochs. In the case of phoneme-level supervision, we explored two settings of loss weights to compute the joint loss value after each training batch. The onset-coda weight percentages for those two settings were 50-50% and 60-40%.

As explained in section 5.2.1.1, each of the 8 participants in the evaluation set have their own train and test subsets. From here on, we refer to these as the *ev-train* and the *ev-test* sets respectively to improve readability. During evaluation, a KNN algorithm was trained on the ev-train set of each participant and evaluated on the ev-test, taking the learnt CNN embeddings and the baseline feature sets as input representations. As these embeddings and feature sets are expected to have different information for the KNN algorithm to rely on, they are also expected to make the classifier perform better or worse given the underlying suitability of these feature sets for classification. As KNN algorithms are purely distance-based and non-parametric, our assumption is that a high classification accuracy using them is more likely to translate into a high accuracy using other types of machine learning algorithms. In Section 5.2.2, we had the opportunity to briefly test this hypothesis by replacing the KNN classifier with three other popular machine learning classifiers whose results are commented on but not reported.

The evaluation procedure above is applied to all proposed and baseline methods except for the GMM-HMM-based speech recognition model. This model does not extract embeddings but rather predicts the ev-test labels directly ignoring the ev-train subset (user-agnostic). Also, to alleviate the disparity in the amount of ev-train data in the AVP and the LVT datasets (1,000 vs. 220 augmented data samples approx.), we extract two different amounts of embeddings and selected features: 32 to carry out evaluation on AVP participants and 16 to do it on LVT ones. This means that for each supervision method we end up having 25 feature sets of size 32 to evaluate on AVP data and other 25 of size 16 to do it on LVT data.

We report final results in two modalities: *participant-wise*, where accuracy scores of single test participants are calculated and averaged, and *sound-wise*, where accuracy scores are calculated for all evaluation vocal percussion sounds in a participant-agnostic way. We express error measures from several training iterations via the standard deviation and the 95% confidence interval.

Then, in order to compare the performance of the best embedding learning model with the ones related to the end-to-end algorithms in 5.1.2, we evaluate on the AVP dataset exclusively. To that end, we evaluated models on one participant at a time while training them with the data pertaining to the remaining 27 participants (leave-one-out strategy), giving a total of 28 models. We ran ten training iterations of each of the models with different weight initialisations and train-validation splits and performed classification with a random forest, a logistic regression algorithm, and a KNN algorithm with 3, 5, 7, 9, and 11 neighbours. We reported the mean accuracy and its associated errors for both participant-wise and sound-wise performances featuring the best-performing classification algorithm.

Finally, in order to better interpret our models' decision-making process, we calculated several *saliency maps* [11] via backpropagation using the embedding learning models trained with the original instrument-level labels. These maps highlight the sections of the input spectrograms that receive a stronger amount of gradient, which correlate with the regions that these models consider important for correct classification. These maps are computed given an input x and the penultimate layer output of a deep learning model A as: $\frac{dA_i}{dx}$, where i is the label with respect to which the saliency map is computed. In our case, A is the CNN model conditioned on the original instrument labels. We aggregated the saliency map by taking the absolute value and thresholding it, so that the top 10% of the map is set to one and the rest is set to zero. Then we averaged the saliency map over the 25 models.

5.2.2 Results and Discussion

5.2.2.1 Classification Performance on AVP-LVT Dataset

Final evaluation accuracies for all methods are gathered in Table 5-D. Best results were obtained using a frame size of 46 ms and, in the case of phoneme-level classification, loss weights of 0.6 and 0.4 for onset and coda phonemes. Also, the best-performing feature selection routine was the one using the original syllable label set.

We can observe in the table that all supervised embedding models except for those supervised with instrument-level classes are consistently superior to baseline approaches, including feature selection approaches. This observation lies in accordance with previous literature on the usefulness of deep learning models as feature extractors for speech utterances [10]. It also highlights the unsuitability of instrument-level classes for embedding learning supervision, which was somewhat expected given that participants have different ways of vocalising drum instruments. Thus, label sets of such (high) level of abstraction are undesirable for embedding learning in our case.

We see that the best performance for both participant-wise and sound-wise evaluation metrics is achieved by supervised embedding learning models that used the original syllable-level label set. This difference is notable not only for the high mean accuracy score but also for its standard

	Baseline			Instrument-Level	
	Timbre	Selection	HMM*	Original	Reduced
Part-wise	.840	.827 ± .078|.030	.725	.812 ± .095|.037	.779 ± .087|.034
Sound-wise	.835	.795 ± .083|.033	.734	.774 ± .097|.038	.738 ± .086|.033

* *User-agnostic model*

	Syllable-Level		Phoneme-Level	
	Original	Reduced	Original	Reduced
Part-wise	**.899** ± **.066**|**.025**	.883 ± .078|.030	.876 ± .071|.028	.874 ± .075|.030
Sound-wise	**.874** ± **.074**|**.029**	.852 ± .080|.031	.840 ± .074|.029	.838 ± .081|.032

	Sound-Level		
	End-To-End	Siamese	Triplet
Part-wise	.861 ± .076|.030	.849 ± .068|.027	.857 ± .084|.033
Sound-wise	.832 ± .078|.031	.831 ± .069|.027	.834 ± .089|.035

Table 5-D: Final evaluation accuracies from generated feature sets. Accuracies are given participant-wise and sound-wise, and best performances for both modalities are highlighted in bold font. For feature selection, only the best performance is reported (reduced syllable-level). Errors from several training iterations are expressed via the standard deviation (left) and the 95% confidence interval (right).

deviation, which is the lowest for both participant-wise and sound-wise metrics. This means that the informative power of the resulting embeddings, apart from being the most prominent, is also the most robust to the two sources of training arbitrariness that we studied here (see Section 5.2.1.4). We carried out experiments using other different classifiers than KNN, namely logistic regression, random forest, and extreme gradient boosted trees. There, we observed that the original syllable-level method still outperforms the rest of the approaches, which further reinforces its suitability for classification. All methods performed similarly on these extra algorithms, both in terms of absolute and relative performances.

Models supervised using original and reduced phoneme-level classes also yielded similar scores to those of the ones with syllable-level supervision, although still lower and generally less robust to training arbitrariness, possibly due to the extra complexity of the multi-task learning approach. The same applied to sound-level supervision, which performed slightly worse than phoneme-level supervision, possibly indicating that very low levels of supervision abstraction are relatively counterproductive for vocal percussion classification engines.

Another reason that could explain this lower performance for sound-level supervision could be its large amount of output labels ($\sim$150 sound types), which complicates validation. In order to tackle this issue, we built and trained an alternative siamese network model [21] with the same architecture as the CNN except for the last fully connected layer, which was removed. This network uses metric learning directly on embeddings to discriminate between same-class sound pairs and different-class ones, therefore reducing our large label vector to a "same-different" binary auxiliary vector. In the end, its final accuracy was found to be moderately lower than the one relative to the CNN classifier, so we kept the latter's result.

We also notice that the generic (user-agnostic) HMM-based speech recognition model performs worse than any other user-based method. This result further evidences the importance of taking user idiosyncrasies into account in amateur vocal percussion classification, making user-based strategies preferable to generic user-agnostic ones. Finally, we observe that accuracies derived using the timbre feature set are higher than all the ones pertaining to baseline feature selection algorithms. This result could indicate an excessive information redundancy of selected features compared to that of the timbre set, which is a more internally cohesive feature set.

A clear limitation of the syllable-level embeddings as inputs to amateur vocal percussion classifiers is that these were learnt using samples recorded with electronic devices in a small room with little background noise, which emulates the typical recording setting of amateur music producers. This is likely to work similarly well in recording studios, where audio quality is higher, but may very well lose a significant part of its accuracy when faced with low-quality recordings or contexts with too much noise. The timbre feature set could be of great help for these situations, as its features are context-agnostic and therefore adapt well to challenging recording scenarios.

5.2.2.2 Classification Performance on AVP Dataset

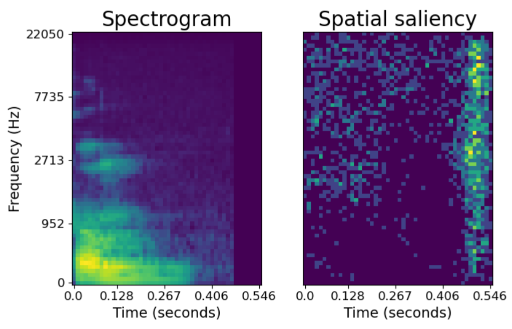
The syllable-level embedding model trained and evaluated on the AVP dataset and a KNN algorithm with 3 neighbours (best-performing final classifier) obtained a participant-wise accuracy score of .784 $\pm$ .087|.054 and a sound-wise accuracy score of .767 $\pm$ .082|.051. These results are slightly lower than the ones we obtained with the end-to-end CNN (.789 and .770) and higher than those from the rest of the methods, including the MLP-Spec (see Section 5.1.2). In particular, we can see how it is significantly more accurate than both the KNN-Heur and MLP-Heur methods, which used 32 heuristic features for classification. This showcases the superior performance of learnt features relative to heuristic ones in our specific classification context. Another important thing to note is that the embedding learning method is also significantly less stable than the end-to-end CNN method, with standard deviations of .087 participant-wise and .087 sound-wise for the former and .031 and .030 for the latter respectively.

Hence, performing feature extraction with our syllable-level embedding model and classification with the KNN algorithm with 3 neighbours is notably faster (the embedding learning model requires no training time), generally prevents overfitting (only 32 features to model with a KNN algorithm), and achieves a comparable performance to that from the end-to-end CNN classifier. In general, we believe that this embedding learning strategy is, overall, the most practical one for user-based amateur vocal percussion classification. However, this comes at the expense of having potentially lower performance stability related to training parameters and validation splits, which might be undesirable under some circumstances.

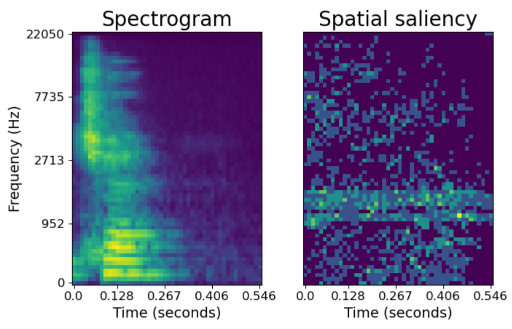
5.2.2.3 Saliency Maps

Four representative examples[3] of these maps are shown in Figure 5.2. We found that, in general, models tended to focus more on frequencies between $1000Hz$ and $2000Hz$ for vocal percussion sounds associated with snare drum, closed hi-hat, and opened hi-hat. This region usually coincides with a high-energy one for these sounds, which often share phonetic representations. It also might indicate a useful thresholding point for the network to tell the sounds apart. Hence,

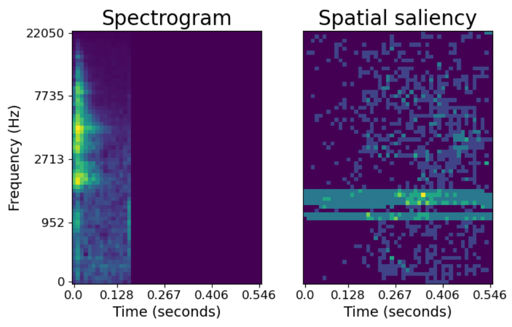
[3]See project's repository for more examples of saliency maps.



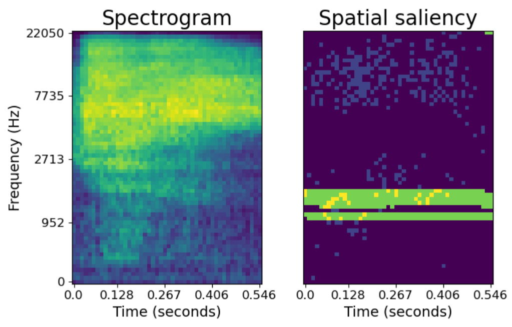
(a) Kick drum



(b) Snare drum



(c) Closed Hi-Hat



(d) Opened Hi-Hat

Figure 5.2: Log mel spectrogram (left) and corresponding saliency map (right) of four sounds of different instrument class.

this could be implying that a higher frequency resolution in this region could potentially improve the accuracy of future amateur vocal percussion classifiers.

The network also appears to attend to silences and regions of lower spectral energy in the case of the kick drum and closed hi-hat. This could mean the absence of energy, especially at the high end of the spectrum, might also be a key feature for the models to differentiate the kick drum and closed hi-hat from the rest of the instruments. The high attention density in silences specifically could also be implicitly suggesting that the duration of vocal percussion sounds is a key factor to distinguish the kick drum and closed hi-hat from the snare drum and opened hi-hat, as the sounds associated with these last two instruments tend to be longer.

5.3 Summary

In this chapter, we have taken a look at vocal percussion classification in an offline context. We have explored and compared the performance of heuristic, end-to-end, and feature learning algorithms. We also weighted their suitability for offline classification attending to several parameters of interest apart from accuracy, like performance stability and training time in the case of user-based approaches.

In our first study, we built several heuristic and end-to-end data-driven algorithms and compared their classification performances using the AVP and the BTX datasets for training and evaluation. We saw that, for amateur vocal percussion, user-agnostic algorithms with fixed sounds significantly outperformed user-based methods, meaning that it is generally desirable to make the users stick to a fixed set of sounds to vocalise for each instrument rather than letting them use the sounds of their choice. We also saw that the user-based methods that used the CNN classifier needed around 3 minutes to train on a CPU, which could potentially be impractical from a user-satisfaction perspective when implementing the algorithm in real-world scenarios. Also, the low classification accuracies relative to beatbox sounds might signal a need for a more thorough study on this form of vocal percussion for offline classification purposes.

In the second study, we explored embedding learning techniques to shorten the training time of user-based amateur vocal percussion classification algorithms and improve their generalisation power by minimising the chances of overfitting. We built several CNNs supervised on four types of label sets (instrument-level, syllable-level, phoneme-level, and sound-level) and used the 32 features in their penultimate layer as input to a KNN classifier. We discovered that the embeddings extracted from the CNN model supervised with syllable-level labels were the best-performing ones in both accuracy and stability to training conditions. Apart from bringing the training times down and reducing overfitting, this embedding learning strategy also achieved comparable performances to those relative to the end-to-end CNN model, making embedding learning an appropriate alternative when classifying user-based vocal percussion sounds. Saliency maps derived from these embedding learning models also revealed that spectrogram regions in the 1-to-2 kHz frequency band and those with absence of energy seem to be of special relevance for these algorithms when classifying drum instruments.

References

[1] A. F. S. Ramires, “Automatic transcription of vocalized percussion,” 2017.

[2] K. Hipke, M. Toomim, R. Fiebrink, and J. Fogarty, “Beatbox: End-user interactive definition and training of recognizers for percussive vocalizations,” in *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, 2014, pp. 121–124.

[3] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, “Specaugment: A simple data augmentation method for automatic speech recognition,” *arXiv preprint arXiv:1904.08779*, 2019.

[4] T. K. Chan and C. S. Chin, “A comprehensive review of polyphonic sound event detection,” *IEEE Access*, vol. 8, pp. 103 339–103 373, 2020.

[5] M. Caetano, C. Saitis, and K. Siedenburg, “Audio content descriptors of timbre,” in *Timbre: Acoustics, perception, and cognition*. Springer, 2019, pp. 297–333.

[6] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. Zapata, and X. Serra, “Essentia: an open-source library for sound and music analysis,” in *Proceedings of the 21st ACM international conference on Multimedia*, 2013, pp. 855–858.

[7] N. Halko, P.-G. Martinsson, and J. A. Tropp, “Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions,” 2009.

[8] A. Delgado, S. McDonald, N. Xu, C. Saitis, and M. Sandler, “Learning models for query by vocal percussion: A comparative study,” in *Proceedings of the International Computer Music Conference*, 2021, accepted.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, “Scikit-learn: Machine learning in python,” *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[10] S. Latif, R. Rana, S. Khalifa, R. Jurdak, J. Qadir, and B. W. Schuller, “Deep representation learning in speech processing: Challenges, recent advances, and future trends,” *arXiv preprint arXiv:2001.00378*, 2020.

[11] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps." [Online]. Available: http://arxiv.org/abs/1312.6034

[12] A. Delgado, S. McDonald, N. Xu, and M. Sandler, "A new dataset for amateur vocal percussion analysis," in *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound*, 2019, pp. 17–23.

[13] L. Nanni, G. Maguolo, and M. Paci, "Data augmentation approaches for improving animal audio classification," *Ecological Informatics*, vol. 57, p. 101084, 2020.

[14] M. Lim, D. Lee, H. Park, Y. Kang, J. Oh, J.-S. Park, G.-J. Jang, and J.-H. Kim, "Convolutional neural network based audio event classification," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 12, no. 6, pp. 2748–2760, 2018.

[15] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: a corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 2010.

[16] S. Evain, B. Lecouteux, D. Schwab, A. Contesse, A. Pinchaud, and N. H. Bernardoni, "Human beatbox sound recognition using an automatic speech recognition toolkit," *Biomedical Signal Processing and Control*, vol. 67, p. 102468, 2021.

[17] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. Conf. IEEE Signal Processing Society, 2011.

[18] E. Demirel, S. Ahlbäck, and S. Dixon, "Automatic lyrics transcription using dilated convolutional neural networks with self-attention," in *2020 International Joint Conference on Neural Networks (IJCNN)*. Ieee, 2020.

[19] T. Anastasakos, J. McDonough, R. Schwartz, and J. Makhoul, "A compact model for speaker-adaptive training," in *Proceedings of the Fourth International Conference on Spoken Language Processing*. Ieee, 1996.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[21] G. Koch, R. Zemel, R. Salakhutdinov *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2. Lille, 2015.

Chapter 6

Online Vocal Percussion Classification

Once the onsets relative to vocal percussion sounds are detected in audio streams, the *classification* process takes place. This is the last stage of the transcription process and consists in assigning a class label to the detected sounds based on the audio information right after the predicted onset. In this chapter, we take a look at vocal percussion classification in an online (real-time) context.

While in offline mode classifiers have access to the whole audio file containing the vocal percussion performance, in online mode classifiers have access to a short analysis buffer that contains the most recent few milliseconds of the recorded audio stream. An online transcription system would have to detect, classify, and usually trigger a response shortly after the sound event is recorded. For instance, an online system would trigger a snare drum sound almost at the same time as the performer vocalises the percussive sound event that is supposed to trigger it. This online procedure puts an important constraint on the system, forcing a trade-off between delay (length of the analysis buffer) and performance (detection and classification accuracy). In this sense, the longer the analysis buffer, the more information is available to the algorithm and the better the performance is expected; but also the more delayed the triggered response will be, which could be perceptually unpleasant if it exceeds a certain threshold that usually depends on the nature of the task at hand.

In this chapter, we firstly investigated which phoneme-to-instrument mappings were the most adequate to perform online amateur vocal percussion classification with. For that, we used three different evaluation criteria to base our decision upon: frequency of use of phonemes among different performers, spectral similarity to reference drum sounds, and classification separability. Following such criteria, the final recommended mappings would potentially feel natural for performers to articulate while enabling classification algorithms to achieve the best performance possible.

In our second study, we evaluated the performance of several algorithms when classifying the above-mentioned sounds for amateur vocal percussion and those in the BTX dataset for beatbox. As input representations, we explored several analysis frames of different lengths and placements along a short audio buffer right after each sound's onset. Like in the case of onset detection, we evaluated the performance of different heuristic and data-driven models given the spectral content within the input analysis windows. We then put the final results in context with those from Chapter 4 (onset detection) and offer a final discussion on which onset detection and classification models to choose for online beatbox and amateur vocal percussion transcription.

All in all, we wanted to answer the following research questions:

- Which phoneme-to-instrument mappings are the most adequate to perform online amateur vocal percussion classification with?
- Which classifier is most adequate to carry out online vocal percussion classification with?
- Are beatbox sounds more separable than amateur vocal percussion sounds in an online context or is it vice versa?
- How do processing times of data-driven algorithms compare to those of heuristic methods?
- How reliable and stable are heuristic and data-driven classifiers?
- Which is the recommended transcription delay to avoid perceptual unpleasantness from response delay while achieving high transcription accuracies?

6.1 Phoneme Mappings for Online Amateur Vocal Percussion Classification

Vocal percussion sound events are mostly composed of plosive, fricative, and affricative phonemes. As the delay buffer to trigger percussive responses in real-time is generally short, some of these phonemes are more distinguishable than others within their first few milliseconds. For instance, the phonemes /p/ and /s/ are vocally articulated in a different way and are likely to be easily discernible, while the phonemes /p/ and /b/ are articulated in a similar way and are usually harder to separate in a classification task, even when having access to the complete sound events. Also, for the process to feel natural to performers, usually music producers and musicians, it is convenient for them to vocalise specific phonemes on the basis of their auditory resemblance with their response triggers. For example, if the response trigger is a kick drum sample, one would prefer a /p/ sound to an /s/ sound, as the spectral content and dynamics of the former sound are generally closer to those of a generic kick drum.

The motivation for this first study is urged by the two above-mentioned facts. More specifically, we try to answer the following question: which group of phonemes provide optimal online Vocal Percussion Transcription (VPT) performance while remaining natural for the performer to use?

We evaluated the appropriateness of each phoneme to make the final set on the basis of three criteria. The first criterion was the *frequency of use* of the phonemes, i.e., how many participants chose these phoneme-to-instrument mappings. The second one was *spectral similarity* to reference drum sounds; that is to say, how similar the phonemes' sounds are to those of real drums. Finally, the third one was *classification separability*, which evaluated how reliably algorithms can distinguish between different pairs of phonemes so as to maximise classification accuracy.

6.1.1 Relevant Work and Data

To the best of our knowledge, this was the first study that addressed the problem of phoneme recommendation for online VPT, although earlier work touched on some aspects of our three criteria. Picart et al. [1] gathered data about the frequency of use of several phonemes among two beatboxers and Stowell et al. [2] released a dataset with fourteen beatbox performances from different participants with phoneme annotations in most of their sound events. In contrast with our case, these sound events are sometimes polyphonic (e.g. /ps/ = kick drum + opened hi-hat) and are heavily influenced by already established beatboxing techniques.

Perhaps the most similar study to the present one regarding the spectral similarity between drum and phoneme sounds is that of Patel et al. [3], where the authors compared several audio features extracted from both tabla sounds and their traditionally associated syllable sounds and found strong correlations between them, which suggests that onomatopoeia might have played an important role in the origin of such tabla vocalisations. A related recent study tied drum sounds with their vocal imitations directly using both engineered features and features learnt by a neural network from the input spectrograms [4]. However, unlike the previous study in tabla sounds, this study did not carry out phoneme-wise feature analysis nor spectral analysis.

An especially relevant piece of research for classification separability and for this study in general was that of Stowell et al. [2]. In this work, which looked at online VPT with beatbox sound events, the authors explored classification accuracies under different frame delays from the onset times and also conducted a listening experiment on the perceived quality of different response delays so as to provide an upper bound to the length of the analysis audio buffer. The accuracies they reported, however, took drum instruments as classes instead of phonemes, and therefore we could not extract phoneme-wise classification separability observations from such results.

Another study [5] presented a VPT system with custom vocal percussion phonemes that could operate in online mode and also informed the users about the classification separability of their chosen sounds. This study did not include an investigation of phoneme-wise classification separability as well. However, a few studies in acoustic-phonetic analysis [6, 7] provided results on plosive phoneme classification separability. They used engineered features like the Mel Frequency Cepstral Coefficients (MFCC) and machine learning models like Hidden Markov Models (HMM) to classify plosive phonemes and illustrated results via confusion matrices. We got inspiration from such methods when planning the methodology of the present study; although, in contrast with our work, the whole phoneme sounds were taken for classification instead of just their first few milliseconds.

In our study, we analysed vocal percussion sounds and drum samples data from two datasets. We used the Amateur Vocal Percussion (AVP) dataset [8] as our reference dataset for vocal percussion sounds. We chose this dataset because it is the one that has the widest variety of phonemes and all its vocal percussion sounds have annotated onsets, instrument labels, and phonetic representations. We specifically used the 4873 sound events that made the personal subset, where participants were asked to trigger the drums in their own particular style. For all participants, we used the sound events from their training (isolated sound events) and testing (beatbox-like improvisation) sets. In the case of the reference drum sounds, we took samples from InMusic's BFD3 library [9] at velocities 64 and 127. These included 150 kick drum samples, 274 snare drum samples, 276 closed hi-hat samples, and 276 opened hi-hat samples. All sounds were sampled at 44100 Hz, and we applied dither to downscale the bit-depth of reference drum sounds from 24 to 16 bits so that it matched that of AVP sounds.

6.1.2 Analysis

6.1.2.1 Frequency of Use

To calculate the frequency of use of each phoneme in the AVP dataset, we first extracted the annotations regarding onset and coda phonemes (see 3.3.3) of sound events contained in the personal subset. If we represent sound events as syllables, onset phonemes would be their first part, generally a plosive, fricative, or affricative phoneme. Coda phonemes would be the second part of the syllable, coming immediately after onset phonemes and usually being vowel phonemes. While coda phonemes are not necessary to build the sound event, they come in handy for offline VPT, providing a further degree of freedom to construct sound events upon. For the frequency of use criterion, we considered the onset phonemes of all sound events, with and without coda phonemes, while for the rest of the criteria we analysed those sound events that consist of just one onset phoneme without coda phoneme (see section 6.1.2.2).

	Kick Drum		**Snare Drum**		**Closed Hi-Hat**		**Opened Hi-Hat**		**All Instruments**	
/dʒ/	1	3.6%	-	-	-	-	-	-	1	3.6%
/k/	-	-	3	10.7%	-	-	2	7.1%	5	17.9%
/kʃ/	-	-	-	-	-	-	1	3.6%	1	3.6%
/kx/	-	-	3	10.7%	1	3.6%	-	-	4	14.3%
/p/	**22**	**78.6%**	3	10.7%	1	3.6%	-	-	22	78.6%
/s/	-	-	-	-	1	3.6%	2	7.1%	3	10.7%
/t/	4	14.3%	**7**	**25.0%**	**19**	**67.9%**	7	25.0%	**24**	**85.7%**
/ts/	-	-	3	10.7%	7	25.0%	**11**	**32.1%**	16	57.1%
/tʃ/	-	-	6	21.4%	4	14.3%	9	32.1%	14	50.0%
/tɕ/	-	-	4	14.3%	-	-	2	7.1%	6	21.4%
/tʒ/	1	3.6%	1	3.6%	-	-	-	-	2	7.1%
/ʡʢ/	1	3.6%	-	-	-	-	-	-	1	3.6%
/ǃ/	-	-	1	3.6%	-	-	-	-	1	3.6%

Table 6-A: Frequency of use of onset phonemes in the AVP dataset. These are the percentages and the raw numbers of participants (out of 28) that used the phonemes to trigger each of the four drum instruments (first four pairs of columns) and that used the phonemes to trigger at least one of the instruments ("All instruments" pair of columns).

We looked at how many participants used each onset phoneme to trigger each of the four instruments. For instance, how many participants used the onset phoneme /t/ to trigger the snare drum sound. The assumption we make here is that the higher the number of participants that decided to use a phoneme to trigger a particular drum sound, the more natural it would feel for a generic performer to trigger that drum sound with that phoneme sound. Results from this analysis are shown in Table 6-A.

There are two clarifications to be made regarding this table. The first one is that the "All Instruments" column contains the raw numbers and percentages of participants that used each phoneme. Note that this is not constructed simply by adding the terms in the rows of the four instruments, as one participant can use the same onset phoneme to trigger more than one drum instrument, in which case it would be still counted as one. The second clarification is that some participants ended up using two or more different onset phonemes interchangeably to trigger a single instrument, in which cases we included both phonemes. For instance, if we take a look at the kick drum column and sum up the numbers we get 29 participants (instead of 28). This is because the participant that used the phoneme /tʒ/ to trigger the kick drum also used the phoneme

/dʒ/ for that purpose.

Looking at the table, we see that the dataset featured several plosive phonemes (/k/, /p/, /t/), fricative phonemes (/s/), affricative phonemes (/dʒ/, /kʃ/, /kx/, /ts/, /tʃ/, /tɕ/, /tʒ/, and /ʔʕ/), and even click phoenes (/!/). The phoneme /p/ was the preferred one to trigger the kick drum, /t/ to trigger the snare drum, /t/ again to trigger the closed hi-hat, and /ts/ to trigger the opened hi-hat. The phoneme /t/ was the one used by the highest number of participants (24) followed by /p/ (22), /ts/ (16), and /tʃ/ (14). It was also the only one used to trigger all four instruments. Interestingly, we found that the phonemes that participants used to trigger the snare drum were significantly varied compared to the rest of the instruments, which would deserve a closer look in future studies.

Here we also selected the most popular phonemes to be considered and analysed in the following sections. We carry out this filtering to both focus on the potentially most natural phonemes to do VPT with and to ensure representative sample sizes for the classification task. Onset phonemes that had a significantly similar method of vocal articulation were grouped together and reduced to one phoneme. This way, the phoneme /k/ accounted for the phonemes /k/ and /kx/; the phoneme /tʃ/ accounted for the phonemes /tʃ/ and /tɕ/; and /tʒ/ accounted for the phonemes /tʒ/ and /dʒ/. By defining such groupings, the "All Instruments" column in Table 6-A changed accordingly and we got that the /k/ group was used by 8 participants, the group /tʃ/ was used by 18 participants, and the group /tʒ/ was used by 3 participants. In order to count the number of samples that each phoneme had, accounting for the newly-defined groups, we took the vocal percussion sound events associated with them that had no coda phonemes (i.e., pure onset phonemes). That way, the phoneme /k/ had 263 samples associated with it, the phoneme /p/ had 532, the phoneme /s/ had 23, the phoneme /t/ had 617, the phoneme /ts/ had 652, the phoneme /tʃ/ had 720, the phoneme /tʒ/ had 56, the phoneme /ʔʕ/ had 46, and the phoneme /!/ had 29. We observed that there was a pronounced jump between the number of samples associated with the phoneme /tʒ/ (56) to that associated with the phoneme /k/ (263). We considered the latter as an appropriate sample size for classification, with its associated phoneme (/k/) being used by more than a quarter of participants (28.6%). Therefore, we kept all phonemes whose sample size lied above 263 samples, leaving us with /p/, /k/, /t/, /ts/, and /tʃ/ as the final phoneme set to be analysed.

6.1.2.2 Spectral Similarity

This part of the process deals with the spectral analysis of the sound events relative to phonemes (/p/, /k/, /t/, /ts/, and /tʃ/) and reference drum sounds (kick drum, snare drum, closed hi-hat, opened hi-hat). For that, we take a single fixed-length frame from the first few milliseconds of each sound, compute its Fast Fourier Transform (FFT), and then apply an Equivalent Rectangular Bandwidth (ERB) scaling operation to the spectrum's frequency axis (ERB-rate scale) to derive the ERB spectrum [10]. This ERB scaling operation approximates the bandwidths of the filters in human hearing and, therefore, the kind of spectra derived from it would allow us to visually compare the sounds on the basis of their perceptual similarity. Our hypothesis here is that the more similar the ERB spectrum of a certain phoneme is to that of a drum sound, the more natural it would feel to performers to trigger that specific drum by vocalising that phoneme.

We also had to select an appropriate frame size to compute the ERB spectrum from. We considered lengths of 512, 1024, 2048, 4096, and 8192 samples, which would be approximately equivalent to 12, 23, 46, 93, and 186 milliseconds respectively. We based our decision on two main properties of sound events. The first one is the sound events' effective duration, i.e., the amount of time that the sound's energy envelope lies above a certain threshold. The second one is how recognisable are sound events when cropped to different frame sizes, both visually and auditorily. For that, we took a look at the sounds' waveforms cropped at different lengths, listened to them, and discussed which length was the one that included the least amount of silence time while allowing samples to remain perceptually discernible. We concluded that 4096 samples was an ideal frame length to conduct the spectral analysis.

An important difference between the drum sounds in the BDF dataset and the phoneme sounds in the AVP dataset is that the former were recorded with professional microphones in a noise-isolated room, while the latter were recorded in an acoustically untreated room with a laptop microphone. This introduces noise when comparing the spectra of a drum sound to that

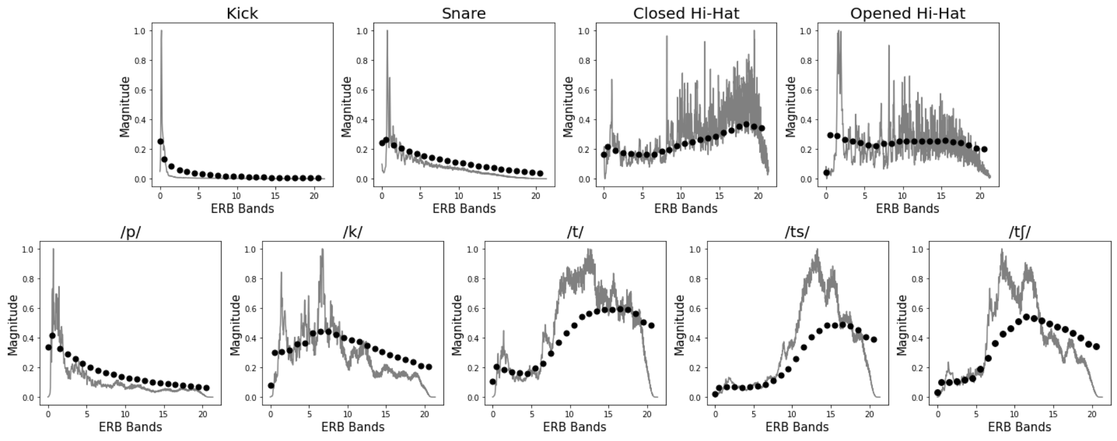


Figure 6.1: Normalised mean ERB Spectra of the first 93 ms of the four drum types and the five phonemes (grey) and mean value of each individual ERB band (black dots).

	/p/	**/k/**	**/t/**	**/ts/**	**/tʃ/**
Kick Drum	**.169**	.063	.042	.022	.033
Snare Drum	**.492**	.416	.303	.228	.303
Closed Hi-Hat	.312	.418	.512	**.522**	.437
Opened Hi-Hat	.321	.448	**.487**	.478	.440

Table 6-B: Mean cosine similarities between individual ERB spectra from drum instruments and those from phoneme sound events. Values in bold are the highest ones per drum instrument.

of a phoneme sound. Fortunately, the AVP dataset includes a fifty-second audio file containing room noise; thus, in order to correct this issue to a reasonable extent, we subtracted the average magnitude spectrum of 500 non-overlapping noise frames to all the phoneme sounds' spectra, clipping results to zero so as to avoid negative spectral magnitudes. This removed some of the components of the recordings' noise, including those belonging to the room and the laptop microphone.

After this, we took the average spectrum of the ERB spectra pertaining to a certain class (e.g. /p/ phoneme, snare drum,...) and we normalised it so that it had a minimum value of 0 and a maximum of 1. These ERB spectra, nine in total, are shown in Figure 6.1. We also compute the cosine similarities between individual ERB spectra from drum instruments and those from phoneme sound events. Table 6-B collects the mean values of the resulting similarity matrices for each drum-phoneme combination.

Several observations about sound similarity could be derived from Figure 6.1 and Table 6-B. For instance, the average spectrum taken from /p/ phonemes looks very similar to the one taken from snare drums. This means that it would potentially feel most natural for users to vocalise this particular phoneme to trigger a snare drum sound. The average spectrum of the /p/ phonemes also bears a marked resemblance to that pertaining to the kick drum, making it a second option to consider. Both these observations are corroborated by looking at the mean cosine similarities of the phoneme /p/ with the snare drum and the kick drum respectively.

The phoneme /k/ average spectrum, on the other hand, seems to resemble that of the opened hi-hat and, looking at the mean cosine similarities, it also appears to bear some similarity to the snare drum, achieving the second-highest similarity score for that particular instrument. We can also see that the average spectrum of phonemes /t/, /ts/, and /tʃ/ seem to be most similar to those of closed and opened hi-hat, especially to the former. This can also be corroborated by looking at the table with the mean cosine similarities, although by this metric the cosine similarity between the phoneme /tʃ/ and the opened hi-hat is slightly higher than the one between such phoneme and the closed hi-hat.

6.1.2.3 Classification Separability

An ideal online VPT system would trigger the response at the exact time that the vocal percussion sound event (the stimulus) begins; but, of course, this is impossible, as the algorithm has no prior information available to base its decision upon, and thus the need for an analysis buffer. Depending on the task at hand, one would require a longer or shorter maximum buffer length and, in the case of VPT, this maximum length is dictated by the perceptual unpleasantness of the system's response delay. Stowell et al. [2] conducted a listening experiment in which a reactive system, right after the input is detected, outputs a mixture of all possible waveforms (kick, snare... etc) until a certain moment (e.g. 11.6 ms after) when this mixture waveform becomes the waveform of the correct class via crossfading. This showed that, for percussive events, listeners perceive delays as acceptable up to a buffer length of 34.8 ms, which we adopted as a reference in our study. Therefore, a VPT system operating in online mode would have to

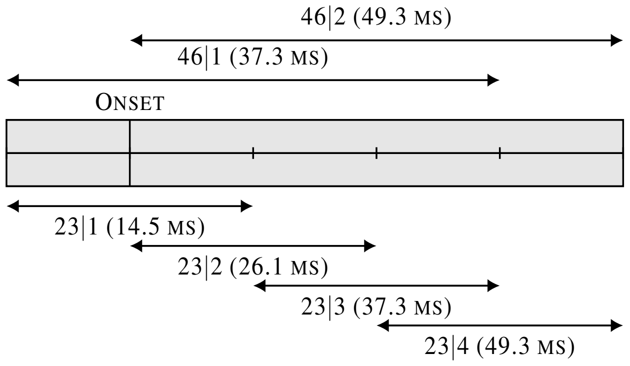


Figure 6.2: Frames for online vocal percussion classification with lengths 23 ms (below) and 46 ms (above). Prediction delays with extra 2.9 ms of measurement error are next to their nomenclature in parenthesis.

detect and classify vocal percussion sound events within 34.8 ms after they are produced by the performer.

Stowell et al. used 23.2-ms long analysis frames with a hop size of 11.6 ms. A delay of 34.8 ms, as they refer to in their paper, means that the analysis frame of 23.2 ms can be centered at 34.8 ms from the onset time at most. If it is centered further than 34.8 ms, performers would experience a delay in the system's response that is likely to not be well-tolerated perceptually speaking. Therefore, if a frame of 23.2 ms is centered at 34.8 ms after the onset, it means that we can analyse up to 46.4 ms from the sounds' onset. Apart from this, we would also want to include some samples before the onset in our analysis buffer, as sound onsets are sometimes not accurately annotated or detected and some part of the sounds' transient might remain before the onset time. We take 11.6 ms before the onset and incorporate them into the buffer. Hence, all in all, the length of the analysis buffer that we used to conduct online VPT was of 58 ms, which would be composed of the 11.6 ms immediately before the onsets and the 46.4 ms after the onsets.

Our buffer could be analysed in several ways. For instance, we could take single analysis frames of different sample lengths centred at different points within the buffer and extract features from them, or even concatenate the features from different frames into a single feature vector. The latter approach was explored by Stowell et al. [2] and showed little to no improvement

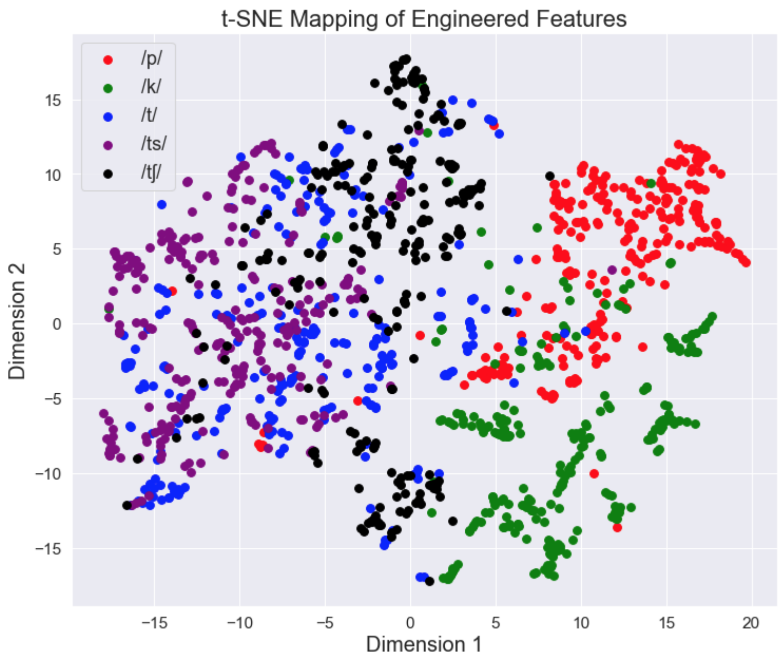


Figure 6.3: Two-dimensional t-SNE mapping of phonemes' engineered features.

in classification separability. Therefore, taking into account that the higher the feature dimensionality the higher the algorithms' risk of overfitting, we decided to take single frames from the buffer and explore them individually. We took six frames whose positions and extensions in the buffer are illustrated in Figure 6.2 and explored their classification performance.

We extracted 38 engineered features from these six frame types. In the case of spectral features, we multiplied the frames by a Hann window before taking their FFT, from which we extracted the features. We extracted a total of 38 features, which included 13 Mel Frequency Cepstral Coefficients (MFCCs), spectral energy of 8 frequency bands (0-300, 300-800, 800-1600, 1600-4000, 4000-7000, 7000-11000, 11000-16000, and 16000-22050 Hz), 4 spectral roll-off frequencies (ratios of 0.25, 0.50, 0.90, and 0.95), spectral complexity, high frequency content, spectral strongpeak, 4 spectral central moments (centroid, variance, skewness, and kurtosis), spectral crest, spectral decrease, spectral entropy, spectral flatness, root mean square, and zero-crossing rate. We also reduced the sample size of the /p/, /t/, /ts/, and /tʃ/ phonemes to 263 via random sampling to ensure that all phoneme classes had an equal number of classes and thus have a balanced-class classification task.

	46|1	**46|2**	**23|1**	**23|2**	**23|3**	**23|4**
Nearest Centroid	**.673**	**.673**	.638	.662	.657	.670
Naive Bayes	.659	**.670**	.544	.649	.665	.624
Single-Layer Perceptron	.792	**.798**	.737	.778	.778	.784
Linear SVM	.782	**.792**	.736	.768	.773	.776
K-Nearest Neighbours	.830	<u>**.841**</u>	.751	.810	.817	.828
Decision Tree	.710	**.728**	.661	.700	.711	.714
Random Forest	.802	**.821**	.739	.789	.801	.809
Extreme Trees	**.704**	.693	.622	.692	.678	.653
Extreme Gradient Boost	.808	**.832**	.758	.803	.815	.804

Table 6-C: Best classification performances from each machine learning algorithm using each of the six frame types as inputs. Results are given in raw accuracy from 0 to 1. Bold numbers are the best performances with respect to the six frame types and the underlined bold number is the best performance overall.

We applied t-distributed Stochastic Neighbor Embedding (t-SNE) [11] to reduce the dimensionality of the feature map from 38 to 2 so as to better visualise phoneme class separability. Figure 6.3 displays such feature map. There, we can see that the /p/ and /k/ phonemes are both moderately distinguishable from each other and very distinguishable from the /t/, /ts/, and /tʃ/ phonemes in a feature-wise sense. This was somewhat expected, as we could eventually see how their average spectra reflected this difference earlier in section 6.1.2.2. It is also worth noting how the data points pertaining to the phonemes /t/, /ts/, and /tʃ/ are very close together in the feature space, possibly meaning that they would be difficult for algorithms to separate. Although the phonemes /ts/ and /tʃ/ display moderate separability from each other, the data points relative to the phoneme /t/ are scattered all over the /ts/ and /tʃ/ regions, making this a possible conflicting phoneme when attempting classification.

The resulting feature vectors were normalised via the z-score before being fed to the classification algorithms. We explored 9 different machine learning-based algorithms (see Table 6-C) so as to provide an algorithm-independent measure of phoneme separability. We implemented the first eight algorithms via the Scikit-Learn library [12] and the ninth with the XGBoost library [13]. We performed hyperparameter optimisation by conducting a grid search for most of the algorithms' hyperparameters, applied 10-fold cross-validation to improve the generalisability and statistical significance of the output accuracies, and reported the best results from each

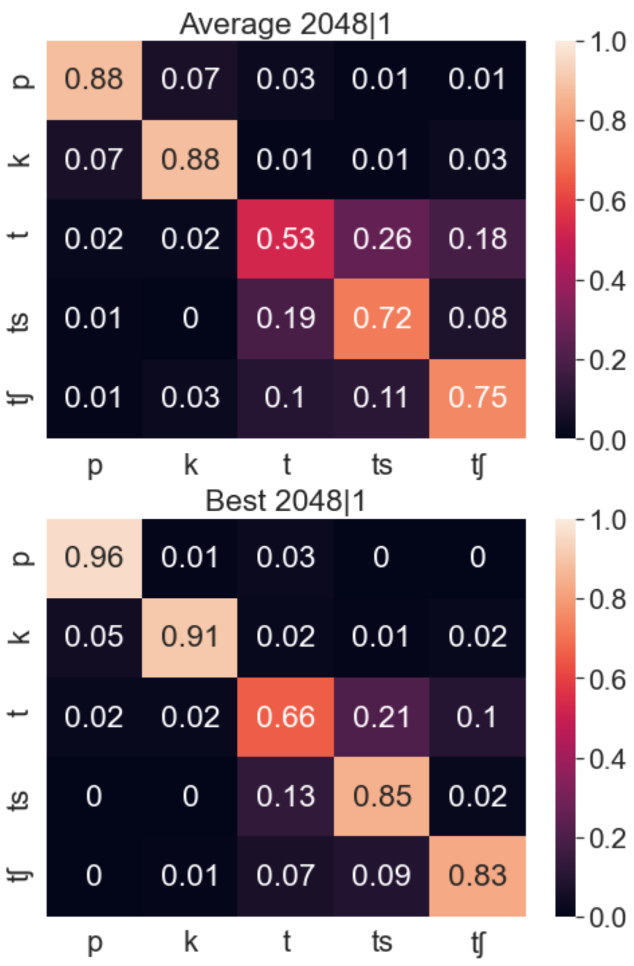


Figure 6.4: Above: confusion matrix taken from the best-performing frame type (46|2) averaged across all nine algorithms. Below: confusion matrix taken from the best classification performance (frame 46|2 + k-nearest neighbours).

method in Table 6-C. There, we could see which algorithm was potentially the best suited for separating the five phonemes and which of the six frame types allowed the algorithms to perform best in general. Most importantly, we wanted to visualise the classification separability of each phoneme with each other, which could be easily done via confusion matrices. We constructed two confusion matrices: one from the best-performing frame type's results averaged through the nine algorithms and another one from the best performance, both algorithmic and frame-wise. These two confusion matrices are displayed in Figure 6.4.

In Table 6-C, we see that the k-nearest neighbours algorithm performed best for all frame types but the 23|1 one, where the extreme gradient boost performed best. Considering that the k-nearest neighbours algorithm simply operates by measuring the Euclidean distance between feature vectors, this highlights the general a priori adequacy of the extracted features. Regarding frame types, we observe that the vast majority of best performances are achieved using the 46|2 frame, i.e., the frame that extends from the onset time to the end of the audio buffer. This is somehow understandable, as this frame was the only one that covered the whole sound from its onset, so it had access to all the changes happening within that small portion of sound that maybe the rest of the frames could miss up to some point.

A more detailed view of the way the algorithms classified vocal percussion sound events into individual phonemes is given by Figure 6.4. We can immediately see that the observations that we drew earlier from Figure 6.3 perfectly translate to both confusion matrices. Essentially, the /p/ and /k/ phonemes are relatively separable from the rest, which lies in accordance to [7], while the phoneme /t/ is hard to separate from the /ts/, and /tʃ/ phonemes, especially in the case of the former.

6.1.3 Discussion

The results drawn from the previous analyses highlighted several properties of phoneme and drum sound events that were relevant for our goal, which was to choose the most appropriate phonemes to do online VPT with. In this section, we integrated the insights from these results and discussed the potential consequences that they may have in a real-life online VPT context. This ultimately led us to the final choice of phonemes which, as expected, depended significantly on the application setting.

The first part of the analysis, which looked at the *frequency of use* of onset phonemes in the AVP dataset, told us that the phonemes that people used the most were the /p/, /k/, /t/, /ts/, and /tʃ/ phonemes. This is the case for amateur vocal percussion, i.e., participants that have not learnt beatbox technique, but looking at the phonemes preferred by beatboxers we realise that they happen to be very similar to those chosen by amateur participants [1, 2]. This suggests that the choice of these phonemes to trigger certain drum samples is based on the vocal imitation of such drum samples to some extent and therefore the phonemes usually feel natural for the generic performer to use. We would therefore recommend using these phonemes for both online and offline VPT.

The second part of the analysis, which covered the *spectral similarity* among phonemes and drum sounds, gave us hints of which phoneme sound events typically sound more similar to a certain drum sound regarding their respective frequency distributions. We saw that the kick drum sound resembled the /p/ phoneme most, which explains the fact that the vast majority of participants (78.6%) selected such phoneme to trigger the kick drum sound. The average spectrum of the snare drum sound was also significantly similar to that of the /p/ phoneme, which is also reflected in their cosine similarity measure. This finding clashes with that of the frequency of use of the /p/ phoneme to trigger the snare drum (only 10.7% of participants) and the generally dispersed frequencies of use among phonemes for that particular drum sound. We suspect that this could be due to cultural conventions inspired by vocal percussion techniques like beatboxing or that it could also be a consequence of having already selected the /p/ sound for the kick drum and being inclined to choose a different one for the snare drum. Either way, this would require a follow-up investigation that falls beyond the scope of the thesis. Given that the /t/, /ts/, and /tʃ/ phonemes resembled closed and opened hi-hat sounds the most, we would be initially inclined to leave the /p/ sound to trigger the kick drum and the /k/ sound to trigger the snare drum, which were the first and the second most-featured phonemes for such instruments respectively.

The picture is completed via the third part of the analysis, which studied the *classification separability* among phonemes. Looking at the final confusion matrices, we saw that the phonemes /p/ and /k/ were generally well separated by the algorithms. This finding, along with the previous ones, provides a further reason to adopt these phonemes for the kick drum and snare drum respectively, as they are often used to trigger these instruments, they are spectrally similar to such instruments, and they can be separated in a classification process with relative ease. Also, the phoneme /t/ is especially hard to separate from the /ts/ phoneme and moderately hard to separate from the /tʃ/ phoneme. This tells us that, although the /t/ phoneme was the most fea-

K	p	**S, HO**	k, ts/tsh
S	p	**HC, HO**	ts, tsh
HC	t	**K, S, HC**	p, k, ts
HO	ts/tsh	**K, S, HO**	p, k, ts
K, S	p, k	**K, HC, HO**	p, ts, tsh
K, HC	p, t	**S, HC, HO**	k, ts, tsh
K, HO	p, ts/tsh	**K, S, HC, HO**	p, k, ts, tsh
S, HC	k, ts	**K, S, HC, HS, HO**	p, k, t, ts, tsh

Table 6-D: Phoneme recommendations for different drum set configurations. Notation: K = kick drum, S = snare drum, HC = closed hi-hat, HO = opened hi-hat, and HS = semi-opened hi-hat.

tured one in the dataset, especially for hi-hat sounds, it might be best avoided so as to optimise classification performances.

In order to select the final phonemes for recommendation, we need to take into account the application context. In a VPT process, both offline and online, the users can select the number of instruments that they want to use for transcription. For instance, they could decide to only choose two sounds to trigger the kick drum and the snare drum, with no triggers for hi-hat sounds. Therefore, if the users could choose among four instruments and include any number of them in the set, we would have a total of 15 potential drum set configurations that would require individual analyses.

Also, the case of hi-hat sounds is usually a complex one. This instrument is able to make timbres that are very different from each other depending on how it is played. Here, we considered the fully closed and the fully opened case, while there also exists the half-opened sound of the hi-hat, which does not let the cymbals vibrate for as long as the fully opened hi-hat sound by bringing them closer to each other, where they clash and thus attenuate each other's sound.

Integrating the insights drawn from the previous discussion and taking the last paragraph's point into account, we built Table 6-D. This table lists all 15 possible drum set configurations along with their recommended trigger phonemes considering each case carefully, including a 16th combination featuring the semi-opened hi-hat as an extra instrument in case performers want to trigger it along with the other four drums. These recommendations are made considering the three criteria that we applied in the present paper, so we did not take into account the ease of their vocal articulation, for instance, or the skill level and consistency that a certain user may have when pronouncing these phonemes, which both fall beyond the scope of our paper. In any case, the phonemes featured in Table 6-D are only meant to be recommendations for the users of the online VPT system, so it would be up to them to decide which phonemes to use in the end.

6.2 Online Beatbox and Amateur Vocal Percussion Classification

For the second and last study in this chapter, we evaluated the performance of heuristic and data-driven algorithms for the online classification of vocal percussion sounds. We attempted classification using the sounds relative to the phonemes that we derived in the last section (/p/, /k/, /t/, /ts/, and /tʃ/) and also using beatbox sounds in a separate classification task (BTX dataset). In their study, Stowell et al. showed that these sounds were reasonably separable from each other in real-time scenarios, whose labels were annotated independently of the participant to whom they belonged [14].

6.2.1 Methodology

Similar to the separability experiments in Section 6.1.2.3, we placed ten frames along the analysis audio buffer whose positions and extensions are displayed in Figure 6.5. This way, we wanted to know the dependence between decision time delay, window length, and classification accuracy.

The frames could be either of length of 23 ms (7 frames) or 46 ms (3 frames), and they operated on the buffer region where the sound classification delay is perceptually tolerable for generic users [14]. The audio content within each frame was sampled at 44.1 kHz. We applied an 10-fold *waveform data augmentation* [15] to these vocal percussion sounds, specifically random *pitch-shifting* (semitone range = [-1.5,+1.5]) and *time-stretching* (stretch factor range = [0.8,1.2]), one after the other in random order.

For the first classification method, we extracted the same 38 descriptors as in Section 6.1.2.3

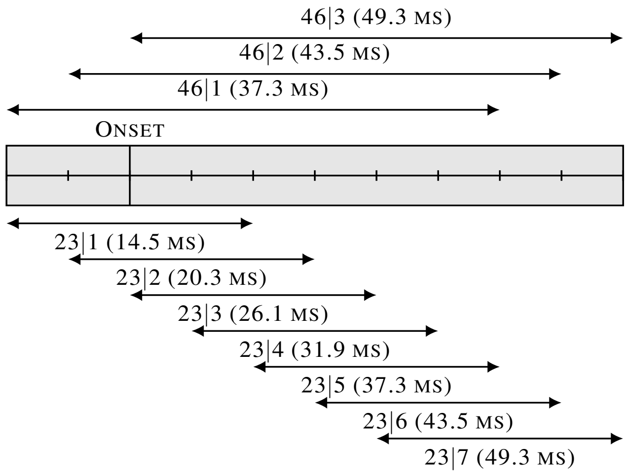


Figure 6.5: Frames for online vocal percussion classification with lengths 23 ms (below) and 46 ms (above). Prediction delays with extra 2.9 ms of measurement error are next to their nomenclature in parenthesis.

and fitted three machine learning algorithms to see which of them maximised accuracy. Two of them were the ones that obtained the highest accuracies in Table 6-C (random forest and extreme gradient boost) and the remaining one was a logistic regression algorithm. We excluded KNN algorithms because of their computational inference time, which grows linearly with the number of samples in the training dataset and could render data augmentation impractical. The fact that the inference times of the remaining algorithms did not depend on the training set size allowed us to keep data augmentation routines in place, which we saw had a notably beneficial effect in this particular task.

We also coded a Multi-Layer Perceptron (MLP) and a 1-Dimensional CNN (1D-CNN) as our data-driven approaches. Both of these took a 128-dimensional Mel spectrum from each of the frames in Figure 6.5, whose audio content was previously multiplied by a Hann window function.

The MLP model had two hidden dense layers, each one with 64 and 32 neurons and a final dense layer connecting to the labels. Each of the hidden layers was followed by a batch normalisation layer and a ReLU activation gate. We found out that batch normalisation on its own gave a better performance for MLP than dropout and a mix of the two.

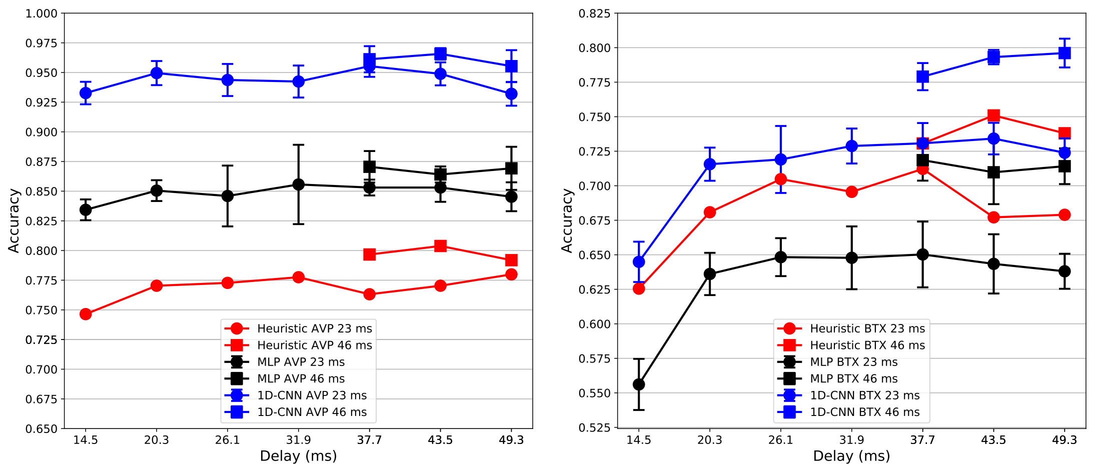
The 1D-CNN model had three convolutional blocks followed by a dense layer with 64 neurons, a ReLU activation function, a dropout layer with a rate of 0.1, and a final dense layer connecting to the labels. Each of the convolutional blocks had a zero-padding layer that padded the input with 12 zeros on both of its extremes, a 1-dimensional convolutional layer using filters of size 15 and strides of 3, and a ReLU activation function. The three convolutional layers had 16, 32, and 64 filters respectively.

The number of neurons in the MLP layers, the number of filters in the CNN layers, and the kernel size of these convolutional layers were optimised using grid search. These models were trained using an Adam optimisation algorithm, early stopping if the validation losses did not decrease after 20 epochs, and learning rate downscaling if validation loss did not decrease after 10 epochs. We trained 10 model iterations with different train-validation-test splits and weight initalisations for each of the input frame types. All splits were stratified, meaning that they were built so that all of them had the same proportion of labels.

Lastly, we conducted a timing experiment in which the inference times of classifiers were measured and compared with each other. This gives us an impression of how much extra processing delay is to be expected given the use of different algorithms. We measured both the preprocessing time and the inference time. Preprocessing time included feature extraction in the case of the heuristic method, spectrum calculation in the case of data-driven methods, and data normalisation for both cases. Inference time was simply the time that took the algorithms to predict the label of the input sample. We run 10,000 iterations for each method and reported the average time.

6.2.2 Results and Discussion

Results for amateur vocal percussion and beatbox are both illustrated in Figure 6.6. We plotted the classification accuracy against the buffer delay in milliseconds. Error bars displayed for MLP and 1D-CNN accounted for the standard deviation of the accuracies relative to their 10 model iterations. The best-performing heuristic algorithm, both for amateur vocal percussion and beatbox, was the logistic regression one.



(a) Amateur Vocal Percussion (b) Beatbox

Figure 6.6: Online classification.

We can see how the 1D-CNN models outperform the rest of the methods for both amateur vocal percussion and beatbox. These models had also more stable performances than those derived from the MLP. An intriguing point of divergence between results on both types of vocal percussion sounds is that, while the MLP achieves significantly better accuracies than the logistic regression for amateur vocal percussion, the opposite is true for beatbox. Although we think this could be related to a high variation of the descriptive power of heuristic features when applied to amateur vocal percussion and beatbox, we could not find any reasonable explanation for this result.

One can also observe how accuracies relative to amateur vocal percussion sounds greatly outperformed those relative to beatbox sounds. This phenomenon could be explained by the fact that amateur vocal percussion sounds had a total of 5 labels (phonemes /p/, /k/, /t/, /ts/, and /tʃ/) while beatbox sounds had 11 labels (see [14] or Section 3.1.1 for more details). We also hypothesize that the 5 amateur vocal percussion phonemes could be better separable than some of the sounds in the BTX dataset. Furthermore, amateur vocal percussion phonemes are better defined than beatbox sounds, as two of the labels in the BTX dataset gathered miscellaneous sounds (identifiable, but not fitting one of the other categories) and sounds where annotators were unsure of classification.

In the case of amateur vocal percussion phonemes, we can see how the 1D-CNN model achieves very high accuracies at small delays (95% at 20.3 ms). This confirms that online amateur vocal percussion classifiers can reach near-perfect performance with an appropriate set of input phonemes, resembling the result obtained in section 5.1.2 for the AVP UA Phon offline VPT task (.970 for CNN). Hence, this kind of input data was not only the best-performing in online classification but also in the offline case. We can also see how 46-ms frames consistently perform better than 23.2-ms ones. This was a rather expected result, as the 46-ms have access to twice the amount of information than the 23.2-ms frames. The difference in performance is especially acute for beatbox sounds, which might be in need of a higher buffer size to achieve optimal accuracies. Despite their higher accuracies, 46.4-ms frames also display higher delays within the analysis buffer, which could be undesirable in some contexts.

Stowell et al. found in [14] that, for beatbox sounds, optimal classification performance with heuristic methods for 23.2-ms frames was achieved with the frame centered at 23.2 ms after the onset, accounting for a total delay of 37.7 ms. In fact, we can see that the maximum classification accuracies for our heuristic method using 23.2-ms frames is precisely reached at a delay of 37.7 ms. However, we can see how accuracies for 23.2-ms frames generally start plateauing at 20.3-ms delay for both amateur vocal percussion and beatbox, a phenomenon we also observed for online onset detection with the same delay. This means that, while maximum accuracy could be reached at later delays and with longer frames, a delay of 20.3 ms using a 23.2-ms frame is usually enough to achieve near-optimal and sometimes optimal transcription performances (onset detection followed by classification). Therefore, if a drum sample is supposed to be triggered right after transcription, we would generally advise restricting the online transcription process to a maximum delay of around 20.3 ms so as to improve the perceptual pleasantness of the triggered response without sacrificing much accuracy.

Putting these classification results in a transcription context, we see that the slightly suboptimal accuracies of online onset detection methods ($\sim$ 90%) could be a limiting factor in amateur vocal percussion transcription performance. However, as we discussed in the introductory paragraph of Chapter 4, online onset detection algorithms deal with several sources of confusion in the detection process, and most of these sources could have an arguably minor effect when detecting the onsets of /p/, /k/, /t/, /ts/, and /tʃ/ sounds. Additionally, these phonemes have higher "plosiveness" or "impulsiveness" than other percussive phonemes included in the final onset detection dataset like /b/, /d/, or /g/. These two facts indicate that the accuracy of online onset detection algorithms could be higher when classifying these amateur vocal percussion phonemes.

Finally, the timing experiments we conducted output average preprocessing times of 2.9 μs for the logistic regression method and 2.1 μs for both the MLP and 1D-CNN methods, and average inference times of 0.8 μs, 24.6 μs, and 69.9 μs for logistic regression, MLP, and 1D-CNN respectively. While the 1D-CNN is significantly slower than the logistic regression and the MLP algorithms, these times are around three orders of magnitude lower than the recommended delay relative to the analysis buffer (20.3 ms). The same applies to the online onset detection preprocessing times in Section 4.2.2: the average processing time for the RNN online onset detector was 7.9 ms per analysed second of audio, which translates to 45.9 μs per frame given 5.8 ms of hop size. Therefore, these added preprocessing and inference times of both onset detection and classification algorithms can be considered residual when calculating the total amount of response delay.

Finally, more research is needed to bring up beatbox transcription accuracies, which could potentially benefit from a better-defined taxonomy, better regularisation techniques, and novel strategies to extend the analysis buffer length without rising response unpleasantness from high delays among others.

6.3 Summary

This chapter has explored the problem of online vocal percussion classification for both amateur vocal percussion and beatbox. It was composed of two independent but related studies, where the findings of the first one were used in the second one.

In the first study, we tried to find the group of phonemes that were the most appropriate for online vocal percussion classification. This appropriateness was estimated via three criteria: the frequency of use of the phonemes, the spectral similarity to reference drum sounds, and their classification separability. We concluded that the /p/, /k/, /t/, /ts/, and /tʃ/ phonemes were all popular choices by participants, had a spectral shape that resembled drum instruments (/p/ and /k/ for kick and snare drum and /t/, /ts/, and /tʃ/ for closed and opened hi-hat), and were relatively separable using machine learning algorithms based on input heuristic features.

In the second study, we applied several heuristic and data-driven algorithms to online vocal percussion classification. For amateur vocal percussion, we built the classification task using the five phonemes derived from the first study as input (/p/, /k/, /t/, /ts/, and /tʃ/). For beatbox, we used the sounds contained in the BTX dataset in a separate classification task. We found that one-dimensional CNNs (1D-CNNs) with input spectral data outperformed MLPs with input spectral data and logistic regression models with input heuristic features. Results also show how the amateur vocal percussion phonemes display high classification accuracy at low delays which, coupled with robust onset detection routines, is likely to achieve a near-perfect online transcription performance. In contrast, the performances of online classification algorithms with beatbox sounds as inputs resembled those in the offline case to a certain extent, meaning that these vocal percussion sounds are also challenging to classify in an online scenario.

References

[1] B. Picart, S. Brognaux, and S. Dupont, “Analysis and automatic recognition of human beatbox sounds: A comparative study,” in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. Ieee, 2015, pp. 4255–4259.

[2] D. Stowell and M. D. Plumbley, “Delayed decision-making in real-time beatbox percussion classification,” *Journal of New Music Research*, 2010.

[3] A. D. Patel and J. R. Iversen, “Acoustic and perceptual comparison of speech and drum sounds in the north indian tabla tradition: An empirical study of sound symbolism,” in *Proceedings of the 15th international congress of phonetic sciences (ICPhS)*, 2003, pp. 925–928.

[4] A. Mehrabi, K. Choi, S. Dixon, and M. Sandler, “Similarity measures for vocal-based

drum sample retrieval using deep convolutional auto-encoders," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* Ieee, 2018, pp. 356–360.

[5] K. Hipke, M. Toomim, R. Fiebrink, and J. Fogarty, "Beatbox: End-user interactive definition and training of recognizers for percussive vocalizations," in *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, 2014, pp. 121–124.

[6] L. Molho, "Automatic acoustic-phonetic analysis of fricatives and plosives," in *ICASSP'76. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1. Ieee, 1976, pp. 182–185.

[7] B. D. Sarma and S. M. Prasanna, "Acoustic–phonetic analysis for speech recognition: A review," *IETE Technical Review*, vol. 35, no. 3, pp. 305–327, 2018.

[8] A. Delgado, S. McDonald, N. Xu, and M. Sandler, "A new dataset for amateur vocal percussion analysis," in *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound*, 2019, pp. 17–23.

[9] [Online]. Available: https://www.fxpansion.com/products/bfd3/

[10] B. R. Glasberg and B. C. Moore, "Derivation of auditory filter shapes from notched-noise data," *Hearing research*, vol. 47, no. 1-2, pp. 103–138, 1990.

[11] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[13] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[14] D. Stowell and M. D. Plumbley, "Delayed decision-making in real-time beatbox percussion classification," *Journal of New Music Research*, vol. 39, no. 3, pp. 203–213, 2010.

[15] L. Nanni, G. Maguolo, and M. Paci, "Data augmentation approaches for improving animal audio classification," *Ecological Informatics*, vol. 57, p. 101084, 2020.

Chapter 7

Drum Sample Retrieval by Vocalisation

In this chapter, we detail the experiments we carried out for Drum Sample Retrieval by Vocalisation (DSRV). DSRV, as described in section 1.2, is essentially a similarity estimation task: the goal is to find audio features that can best link reference sounds with their vocal imitations.

We split the chapter into two parts. The first one was a preliminary analysis of the MDV dataset [1] to explore user differences in vocal imitation styles, discover audio features that these users imitate skilfully, and inform data-driven DSRV systems on how to represent the audio data that their learning algorithms take as input. For that, we used time-frequency visualisation techniques, hierarchical clustering, and statistical tests like the Mantel test [2] among others.

In the second part of the study, we used the insights from the first part to explore the potential of data-driven techniques to learn useful features for DSRV. The experiments here were carried out in a purely user-agnostic way, as data-driven user-based techniques require more reference-imitation sound pairs than there were publicly available at the time.

All in all, we wanted to answer the following research questions:

- How similar are drum sounds and their vocal imitations?
- Which audio features correlate best the acoustic space of drum sound with that of vocal

imitations?

- To what extent are learnt embeddings more informative than heuristic feature when linking drum sounds with their vocal imitations?
- Does conditional feature learning produce more informative features than unconditional feature learning?

7.1 Preliminary Study: Spectral and Temporal Cues

In this first part of the chapter, we analyse the MDV dataset via spectrogram representations and study the correlation of several audio descriptors extracted from drum samples with the same descriptors extracted from vocal imitations. These experiments tell us to what extent listeners can communicate salient spectral and temporal features via vocal imitations while giving us hints on how to calculate input spectrograms for data-driven DSRV to maximise their informative power.

We specifically analyse the audio files contained in the MDV dataset [1] which, as seen in Chapter 3, includes 30 drum sounds from acoustic and electronic sources and their respective vocal imitations performed by 14 participants. This accounts for a total of 30 reference drum samples and 420 vocal imitations (30 per participant). Reference sounds included 6 kick drums, 6 snare drums, 6 hi-hats, 6 toms, and 6 cymbals, and all audio files were sampled at 44.1 kHz.

7.1.1 Experiment 1: Average Spectrograms

For this first experiment, we took the sounds pertaining to each source (reference or imitation) and instrument type (cymbal, hi-hat, kick drum, snare drum, and tom) and averaged their spectrograms for visualisation. This way, we had 5 plots averaging the 6 spectrograms from each reference drum instrument and other 5 plots averaging the 6 spectrograms from each imitation sound for the 14 users (84 sounds in total). This would let us see how sounds from different sources and instrument types compare to each other and assess the potential of the spectrogram representation as input for data-driven DSVR algorithms.

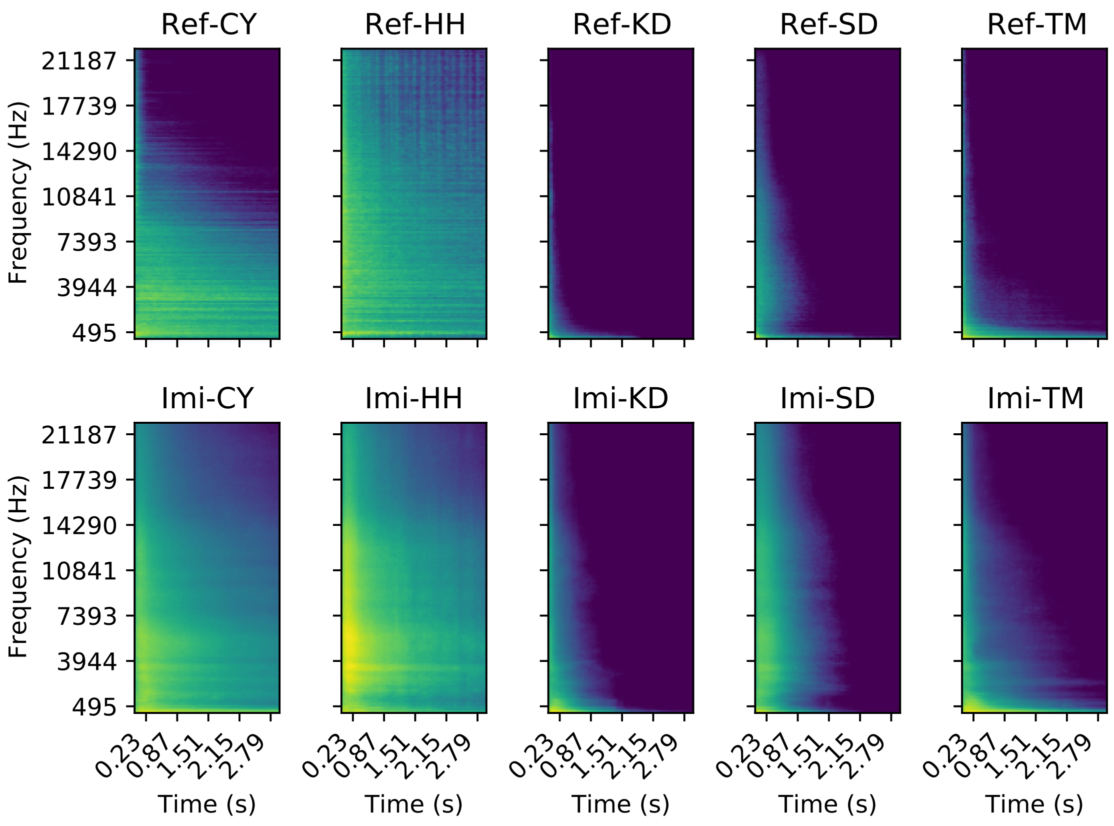


Figure 7.1: Normalised average FFT spectrograms of reference and imitation sounds per instrument category. Magnitude is given in decibels. Ref = reference sounds; Imi = vocal imitations; CY = cymbals; HH = hi-hats; KD = kick drums; SD = snare drums; TM = toms.

The sounds' average FFT spectrograms, calculated at 44,100 Hz of sample rate with 46.4 ms of frame size and 2.9 of hop size, are displayed in Figure 7.1. We can see how most of the energy contained in reference drum sounds and vocal imitations is found in the lower half of the spectrograms. This implies that a band-based compression technique that augments the resolution of the low end of spectrograms could make spectrograms better represent spectral features related to lower frequencies, where most information concentrates. This kind of compression can be achieved by postprocessing the FFT spectrogram with non-linear frequency scales like the Mel, Bark, or ERB ones. We also see that practically all the sounds' information, especially for kick drums, snare drums, and toms, is contained in the first 1.5 seconds of audio. We thus consider this an appropriate truncation length when calculating input spectrograms for DSRV systems.

7.1.2 Experiment 2: Timbre Analysis

In this second round of experiments, we analysed the patterns of occurrence of three timbre descriptors in drum sounds and vocal imitations. This is to evaluate to what extent participants reproduced these timbre features as appearing in reference drum sounds and to explore inter-personal differences in such reproduction patterns. This way, if most participants reproduced a certain feature, it would mean that such a feature is useful for user-agnostic DSVR. Furthermore, the acoustic interpretation of the timbre descriptors would signal best practices when calculating input spectrogram representations for data-driven DSRV algorithms.

7.1.2.1 Timbre Descriptors

We extracted three timbre features. Two of them, the *log attack time* and the *derivative after maximum* describe the envelope's shape. The remaining one, the mean *spectral centroid*, describes the energy ratio between low and high frequencies. For the frame-wise calculation of the spectral centroid, we used a Hann window with size of 46.4 ms, a hop size of 5.8 ms, and the mean statistical functional to aggregate the descriptor through time. The three descriptors were all computed using the Essentia library [3].

Log Attack Time: This is the logarithm in base 10 of the attack time of the sound's envelope, which is the time between the sound's onset and its maximum amplitude. We place the start of the attack where the envelope's amplitude is one-tenth of its maximum value to avoid noise interferences and its end when it reaches the maximum amplitude.

Spectral Centroid: This descriptor, informally referred to as the "center of gravity" of the spectrum, is defined as the normalised average frequency weighted by amplitudes. This descriptor highly correlates with the perception of sound "brightness" and, like the log attack time, it has been found to be an essential descriptor in sound timbre analysis.

Derivative After Maximum: This descriptor measures the slope of the envelope's region right after its maximum amplitude. More technically, it calculates the mean of the envelopes' derivative weighted by its amplitude.

The log attack time and the derivative after maximum were chosen because they are highly characteristic of impulsive sounds. Taking the ADSR model of the sound envelope [4], the log attack time would relate to the attack region and the derivative after maximum would relate to the decay, sustain, and release regions. However, the envelope of impulsive sounds like drums and their vocal imitations can be mostly described by its attack and release regions, as it has negligible decay and sustain parts. In this way, both descriptors help discriminate between impulsive sounds via their "impulsivity": the lower the value of these two descriptors for a certain sound, the more impulsive this is.

The spectral centroid and the log attack time are also well-established salient descriptors of timbre [5]. Moreover, the spectral centroid was the most selected descriptor across participants and models for pure user-based amateur vocal percussion transcription, as seen in section 5.2.1.4. Assuming that the trigger sounds that participants chose for VPT were inspired by the original drum sounds to a certain extent, we considered that the fact that algorithms relied on the spectral centroid while classifying vocal percussion sounds could signal that participants often used this descriptor to differentiate between their own trigger sounds. If that is the case, there would be a high chance that the spectral centroid is also being used to imitate reference drum sounds in a DSRV context.

Although the derivative after maximum is an envelope descriptor and hence timbral, we suspected that, for impulsive sounds, it could be highly correlated with the duration of the sound, which is non-timbral by definition. The reason for this was that the maximum amplitude of impulsive sounds is usually located very near their onset, and their release section accounts for most of the sound's duration. The release section of drum sounds in particular is characterised by the stabilisation of the resonant percussive body and the early and late reflections from the room where the sound took place. To check if this correlation between the derivative after maximum and the duration could exclude the former from our timbral feature set, we measured the Pearson correlation coefficient between these two descriptors across all reference and imitation sounds. We found this correlation coefficient to be of r=.63 ($p<.001$). Although this is a moderately high correlation, we considered it to not be high enough to regard the derivative after maximum as a non-timbral descriptor in the context of this experiment.

7.1.2.2 The Mantel Test

The Mantel test is a statistical test that measures the correlation between two symmetric matrices of the same dimensions. In our case, one of such matrices relates to the acoustic space of drum samples and the other one to the acoustic space of vocal imitations. In general, the higher the mantel score (with a significantly low p-value) that a feature has for a certain imitator, the more faithfully the imitator reproduces the feature vocally.

To calculate the Mantel score and its associated p-value, as detailed in [2], we constructed two Euclidean distance matrices for a certain audio feature and imitator. Both matrices are given, respectively, by:

$$D_{ij}^{ref} = \left| x_i^{ref} - x_j^{ref} \right|^2 \tag{7.1}$$

$$D_{ij}^{imi} = \sum_{n=1}^{N} \left| x_{n,i}^{imi} - x_{n,j}^{imi} \right|^2 \tag{7.2}$$

where $i \in [1, 30]$ and $j \in [1, 30]$ are the indices of the drum sound classes, N is the number of imitators (14 in our case), x^{ref} are the feature magnitudes from the reference drum sounds, and x_n^{imi} are the feature magnitudes from the vocal imitations pertaining to the n-th user with $n \in [1, 14]$. The first matrix (eq. 7.1) was built by calculating the Euclidean distance from single descriptor values taken from all drum samples. It was, therefore, a symmetric matrix of dimensions 30x30. For the second distance matrix (eq. 7.2), we first calculated the Euclidean distances from descriptor values taken from vocal imitations of individual participants and we then averaged the 14 resulting matrices into a single one of size 30x30.

Once we had the two symmetric distance matrices, we performed and averaged five iterations of the Mantel test [2] on them, measuring the degree of statistical (Pearson) correlation between them with the associated p-value. The Mantel test scores relative to the derivative after maximum descriptor for all 14 imitators gave a mean result of $\overline{r}$=.561, a standard deviation of σ_r=.114, and a maximum p-value of p<.001. For the spectral centroid, the results were $\overline{r}$=.428, σ_r=.153, p<.024 for a subset of 13 participants, and $\overline{r}$=.102, p=.411 for one participant alone. For the log attack time descriptor, the scores were $\overline{r}$=.011, σ_r=.130, p<.997. Therefore, in summary, we found that the spectral centroid and the derivative after maximum descriptors were consistently found to be relevant DSRV descriptors across the 14 imitators, while the log attack time did not.

The fact that the log attack time descriptor failed to be a good predictor for one acoustic space given the other indicates that, despite playing an important role in timbre perception, log attack time might be difficult to reproduce vocally with enough precision, at least in the case of percussive sounds, which have a fast attack. Instead, it appears that listeners can better imitate a temporal envelope cue related to the length of the sound's tail, i.e. the derivative after maximum. This observation inspires further research from a timbre perception perspective, as the derivative after maximum could play a role in the perception of percussive sounds and impulsive sounds in general. Interestingly, participants appear to also imitate the spectral centroid cue dexterously, which seems to reinforce its role in describing one of the most salient dimensions of timbre.

Both the log attack time and the derivative after maximum descriptors benefit from a higher time resolution for accurate measuring while the spectral centroid benefits from a higher frequency resolution instead. The derivative after maximum descriptor is not as sensitive to time resolution as it is the log attack time, as the attack region of percussive sounds is significantly shorter than their release region. However, the representation of the derivative after maximum in input spectrograms might be affected if time resolution is too low. Therefore, we consider that DSRV algorithms may generally benefit from higher frequency resolution than time resolution, but not too high so that the derivative after maximum could be faithfully represented in input spectrograms.

7.1.2.3 Hierarchical Clustering and Other Plots

For the final part of this experiment, we picked the two best-performing timbre descriptors, the spectral centroid and the derivative after maximum, and plotted their values against each other for

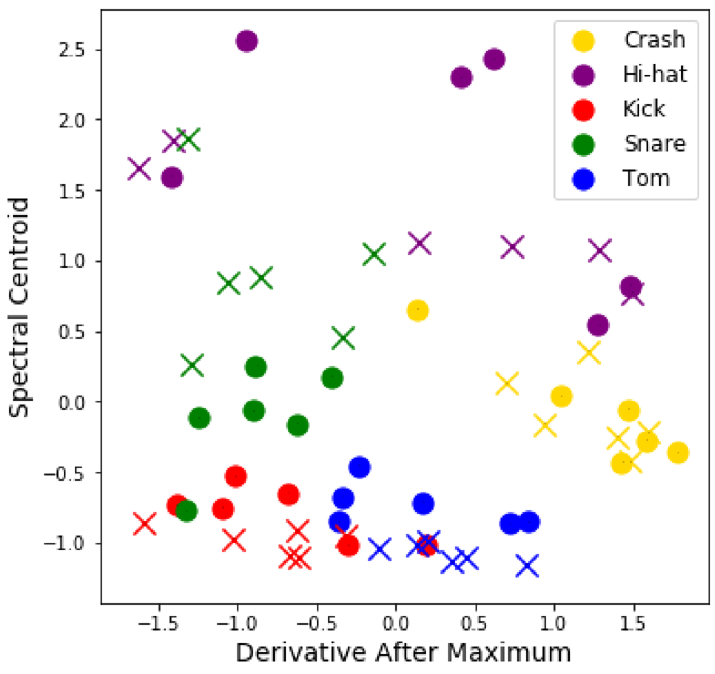
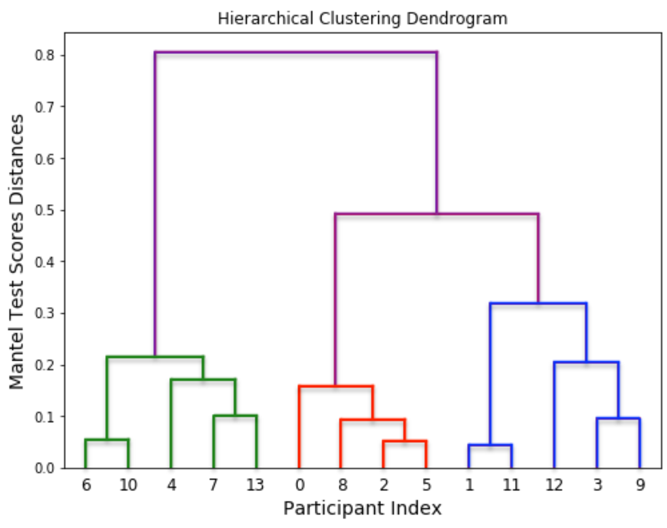


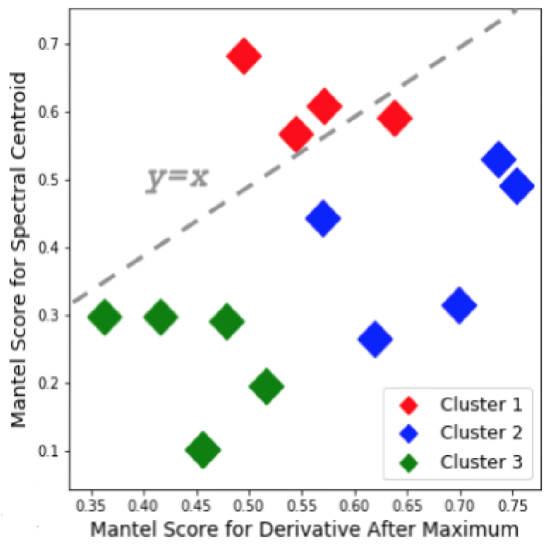
Figure 7.2: Normalised mean values of spectral centroid plotted against derivative after maximum for all sounds. We used circle markers for drum sounds and crosses for vocal imitations averaged across imitators.

drum sounds and vocal imitations, using different colours for the five imitated instruments. In the plot, shown in Figure 7.2, we can see how the two descriptors can indeed help cluster the different instruments and their imitations in the timbre space. Also, we observe that descriptors' values from same-category drum instruments are generally close to those from their same-category vocal imitations. This could be telling us that imitators were aware of characteristic timbral features from different drum instruments and used them when imitating their sound.

Lastly, we also applied an *agglomerative hierarchical clustering* algorithm to group imitators based on the two-dimensional Euclidean distances between their Mantel test scores for the spectral centroid and derivative after maximum descriptors. Figure 7.3 shows the dendrogram derived from the hierarchical clustering process with three clusters and the clusters in the two-dimensional plane where the distances were calculated. It can be observed how, as reported earlier, the imitations of most participants are better captured by the derivative after maximum than by the spectral centroid (points below the y=x line) except for three participants belonging to Cluster 1.



(a) Dendrogram



(b) Clustering Plane

Figure 7.3: (a) Dendrogram from hierarchical clustering process and (b) Mantel scores from spectral centroid plotted against those from derivative after maximum for all imitators with applied hierarchical clustering.

We can also observe some relevant interpersonal differences in the way these descriptors were imitated. For instance, participants in Cluster 1 imitated both the derivative after maximum and the spectral centroid most skilfully, especially the latter. This indicates that they attended to spectrum-related and also to envelope-related timbral cues in drum sounds and faithfully reproduced them in their vocal imitations. Participants belonging to Cluster 2 imitated the derivative after maximum better than the spectral centroid, which signals a more temporal, envelope-focus vocal imitation style. Finally, participants grouped in Cluster 3 also imitated the derivative after maximum better than the spectral centroid but significantly worse than participants from Cluster 2. These results signal that making DSRV algorithms aware of personal idiosyncrasies, i.e., making them user-based, could potentially improve their retrieval accuracies.

7.1.3 Experiment 3: Psychoacoustical Analysis

In this final experiment, we evaluated a larger psychoacoustically-driven feature set made from audio descriptors related to timbre, pitch, loudness, and duration. The set was composed of the first 12 Mel Frequency Cepstral Coefficients (MFCC) excluding the zeroth coefficient, their first derivatives, the duration, the derivative after maximum, and the mean and standard deviation of the loudness, the spectral centroid, and the pitch, which was calculated using the time-domain YIN method [6]. The MFCCs, widely used in timbre analysis [7–9], were computed using 40 Mel bands, a frame size of 46 ms, and a hop size of 11 ms.

We performed five iterations of the Mantel test as described in 7.1.2 and plotted the heatmap of mean Mantel scores and p-values for each of the features and for each of the imitators in Figure 7.4. This helped us better visualise the relevance of these descriptors for DSRV and their implications when building spectrogram-based input representations for data-driven models.

We see that the p-values related to the derivative after the maximum of the envelope are very low for all imitators. This relates to the finding in [10] that imitators tend to reproduce descending loudness ramps with high accuracy. We also note low p-values for the mean of the spectral centroid, MFCC 1, and MFCC 2. These coefficients, much like the spectral centroid, relate to the energy ratio between low and high frequencies and the perceived "brilliance" of the sounds [5]. The high Mantel scores and low p-values for the duration of the sound indicate that this feature was also emulated dexterously by all imitators, which was found to be of critical importance in [11] for generic sounds. We can also see how the mean of the derivative of the MFCCs, except for the MFCC 1 and MFCC 2, has significantly higher Mantel scores and lower p-values than the mean of the MFCC vectors. This could mean that sounds' timbral

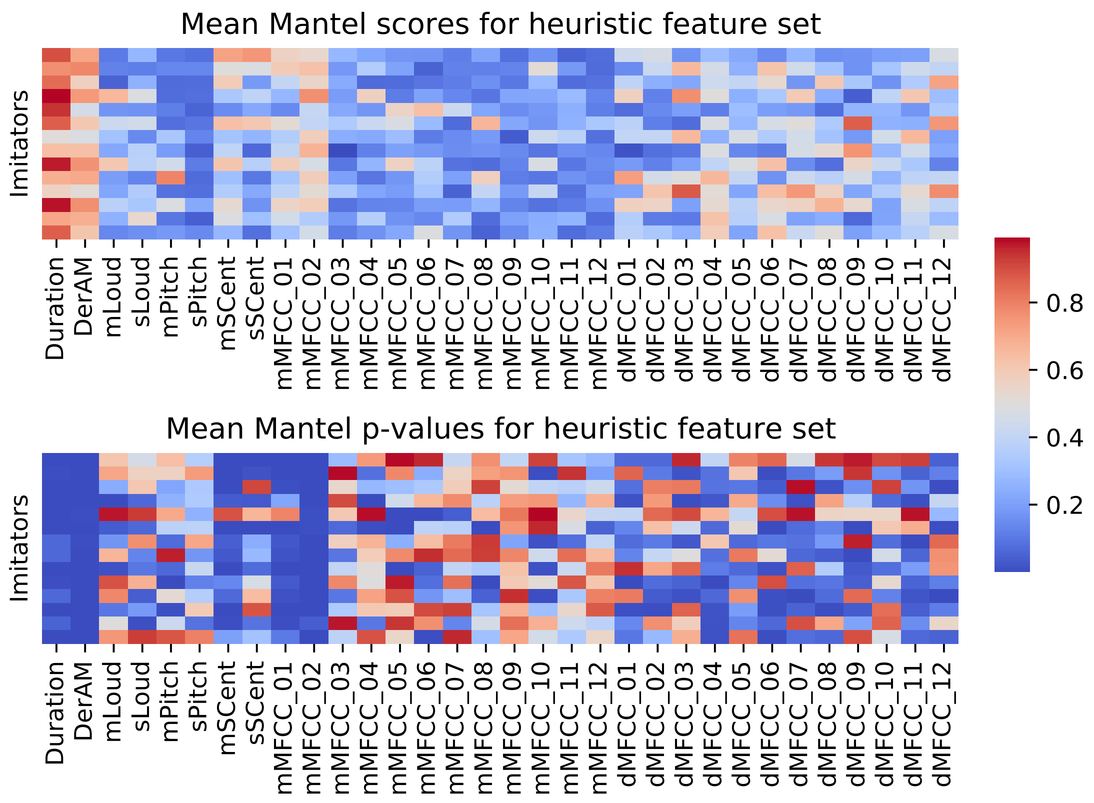


Figure 7.4: Mean Mantel scores and p-values (5 iterations) for each imitator using the features in the heuristic set. DerAM = envelope's derivative after maximum; Loud = loudness; SCent = spectral centroid; m = mean; s = standard deviation; d = derivative.

dynamics play an important role in the vocal imitation of drum sounds, pointing toward the need for a relatively high temporal resolution when building the spectrogram representations for data-driven DSVR algorithms.

A final observation here is that both [11] and [10] found that pitch was a key feature in vocal imitations, but it does not appear to be the case for drum sounds. This could be explained by some reasons, one of them being the fact that percussion sounds are generally unpitched. Another reason is that pitched percussion sounds usually have a fundamental frequency that is not as loud as their noisy components and, therefore, non-salient perceptually speaking. Lastly, another explaiation is that imitators reproduce the musical note related to the pitch but in another octave, sing out of tune, or right out ignore the pitched component of the percussion sound. We have indeed observed these three behaviours in the MDV dataset and, while a few participants relied on pitch to imitate percussion sounds, it was generally not the case on a regular basis (see Figure 7.4).

7.2 Conditional Auto-Encoders for DSRV Feature Learning

In this section, we use the observations from the above study to explore the potential of data-driven feature learning algorithms for DSRV. In particular, we investigated the potential of conditional autoencoder models to learn informative features in this context. We assessed the usefulness of their embeddings using four evaluation metrics, two of them relative to their acoustic properties and two of them relative to their perceptual properties via human listeners' similarity ratings. Then, we also looked into user differences in vocal imitation style via the Mantel test like we did in Section 7.1.3.

This study can be seen as a continuation of the work of Mehrabi et al. [1], which featured convolutional autoencoders as the data-driven feature learners. As our work is specifically directed to improve DSRV applications, we included two acoustics-based drum-imitation similarity metrics to complement the two perception-based similarity metrics in Mehrabi et al.'s paper. Acoustics-based metrics differ from perception-based ones in that the former investigate the correspondence of variations in the acoustic space of drum sounds with those in the acoustic space of vocal imitations, while the latter investigate the correspondence of drum-imitation feature distances with human listeners' drum-imitation similarity ratings. Apart from the inclusion of such metrics, we (i) explored three types of label conditioning, (ii) investigated how networks' architectures and hyperparameters affected performance, (iii) explored imitators' differences in vocal imitation styles, (iv) included new drum and vocal percussion datasets for training purposes, and (v) modelled single hits from vocal percussion and drum sounds exclusively, in particular those related to kick drum, snare drum, closed hi-hat, and opened hi-hat.

In Section 7.2.1, we present the dataset of drum sounds and vocal percussion used throughout the study and detail the implementation of the proposed deep conditional models, baseline

Dataset	Type	Usage	# Samples
Misc	Drums	Training	7,900
BFD3 [12]	Drums	Training	976
AVP [13]	Imitations	Training	5,220
BTX [14]	Imitations	Training	2,317
LVT [15]	Imitations	Training	1,682
BTX2 [16]	Imitations	Training	191
MDV [1]	Both	Evaluation	270

Table 7-A: Datasets of drum sounds and vocal imitations used throughout this study.

methods, and evaluation metrics and routines. We present the final results and discuss their implications for DSRV in Section 7.2.2.

7.2.1 Methodology

The main goal of this study was to discover audio embeddings that can best link vocal imitations with their reference drum samples both acoustically and perceptually. This section details the methodology followed in the present work to get the above-mentioned embeddings, including routines for data curation and preprocessing, model design and training, and final evaluation. The implementation of these can be found in our project's repository[1].

7.2.1.1 Data and Pre-Processing

We used several open-source and commercial datasets of drum samples and vocal percussion performances to train and evaluate our proposed algorithms. These are gathered in Table 7-A. All drum samples and vocal percussion sounds had associated a label relative to the drum instrument they described or emulated: kick drum, snare drum, closed hi-hat, and opened hi-hat.

Training files relative to *vocal imitations* (9,410 sounds) were taken from three vocal percussion datasets: *AVP* [13], *BTX* [14], *BTX2* [16], and *LVT* [15] (second and third subsets). In the case of the AVP dataset, we used the sounds contained in the personal subset and in seven improvisation performances from the fixed dataset, specifically those from participants 4, 8, 12, 16, 20, 24, and 28. We also exclusively used the samples in the BTX and BTX2 datasets that were annotated as kick drums, snare drums, closed hi-hats, and opened hi-hats. The *real drum*

[1]https://github.com/alejandrodl/drum-sample-retrieval-vocalisation

samples (8,876 sounds) were taken from miscellaneous sound libraries containing both acoustic and electronic sources. We only took the files whose names or directories reveal their class label (e.g. “snare01.wav”, “SN01.wav”, “/snares/01.wav”...).

The files we used to evaluate the embeddings were taken from the *MDV* dataset [1]. This included 30 drum sounds from acoustic and electronic sources and their respective vocal imitations performed by 14 participants. After removing the samples relative to cymbals and toms, we had 18 drum samples and 252 vocal imitations (18 per participant). Reference sounds included 6 kick drums, 6 snare drums, 2 closed hi-hats, and 4 opened hi-hats. Silences sections were trimmed for all sounds.

We applied an 8-fold *waveform data augmentation* [17] to drum samples and vocal percussion sound events, specifically random *pitch-shifting* (semitone range = [-1.0,+1.0] for drums and [-1.5,+1.5] for vocal imitations) and *time-stretching* (stretch factor range = [0.7,1.3] for drums and [0.8,1.2] for vocal imitations), one after the other in random order.

After this waveform data augmentation process, we calculated the spectrogram representations based on the results that we derived from the preliminary analysis (Section 7.1). We decided to use the Bark scale [18] to compress the spectrogram, as it is psychoacoustically motivated and was featured in the study that we are basing ours on [1]. Considering that we needed both frequency and time resolution to accurately depict spectral and temporal audio cues in vocal imitation, we compared different frame sizes (23.2, 46.4, and 92.8 ms) and hop sizes (5.8, 11.6, and 23.2) that would strike an appropriate time-frequency resolution balance while covering most of the relevant region of the sounds. We chose a frame size of 46 ms and a hop size of 11 ms and calculated spectrograms with final dimensions of 128 frequency bins by 128 frames. The representations, therefore, extended to a length of approximately 1.5 seconds, which we considered appropriate in Section 7.1.1. Finally, following [1], we applied the Terhardt’s ear model curves [19] to scale the spectrograms to decibels.

7.2.1.2 Baseline Methods

Apart from extracting the same psychoacoustically-motivated feature set as in 7.1.3, we also implemented the regular *Convolutional Autoencoder* (CAE) model in [1]. We reproduced the architecture and the training routine of the best-performing model as reference. As we are evaluating sets of 32 features, we added a final fully-connected layer to the original model that connects its latent space of size 128 to an adapted one of size 32.

7.2.1.3 Proposed Models

We adopted the CAE in [1] as the base architecture of our proposed model. From there, we applied several modifications that we later evaluated in terms of performance both individually and jointly (see section 7.2.2). These modifications to the original model included:

- A variational latent space [20].
- The replacement of upsampling layers and convolutional layers with one unique transposed convolution in the decoder.
- The use of max pooling operations instead of strided convolutions in the encoder.
- A change in the number of filters per layer: [4,8,16,32], [16,16,32,32], and [8,16,32,64].
- A change in the kernel dimensions of the filters: 3x3, 3x5, 5x3, 3x7, 7x3, 5x5, 5x7, 7x5, 7x7, 9x9, and 11x11.

The modifications above were explored considering hyperparameter combinations that ensured that the networks had a similar amount of training parameters ($\sim$ 400,000).

Once our final proposed model was built, we investigated the impact of the inclusion of label information in the model's architecture. This is known as *model conditioning* [21] and it has been proven beneficial in some audio feature learning tasks [22, 23]. We built three models with the same architecture as our proposed CAE but conditioned on different label sets. These sets were (i) *Sound-type Labels* (SL) to distinguish between drums and vocal imitation (two labels);

(ii) *Drum-type Labels* (DL) to distinguish between kick drum, snare drum, closed hi-hat, and opened hi-hat (four labels); and (iii) *Sound and Drum-type Labels* (SDL) to distinguish between kick drum, snare drum, closed hi-hat, and opened hi-hat and whether they are drums or vocal imitations (eight labels). We fed the labels by concatenating a one-hot encoded vector to the flattened feature maps after the last convolutional block in the encoder and to the ones before the first convolutional block in the decoder.

Models were trained using an Adam optimisation algorithm [24], early stopping if validation loss has not decreased after 10 epochs, and downscaling of the learning rate by a factor of 0.2 if validation loss has not decreased after 5 epochs.

7.2.1.4 Similarity Metrics

A good DSRV algorithm would rank drum samples in an appropriate order of perceptual similarity to an input vocal imitation. Measuring this similarity, however, is essentially task-dependent and hence challenging to estimate. If imitations are associated with a single sound, like in our case, one can use query-by-vocal-imitation metrics like the mean reciprocal rank to evaluate systems [25–27], although it does not assess the similarity of lower-ranked samples with the imitation. To account for the latter, other metrics and annotations like drum-imitation perceptual similarity ratings from listeners are needed [1].

With this in mind, we evaluated the relevance of our feature sets via four similarity metrics. On the one hand, the *Mean Reciprocal Rank* (MRR) [28] and the *Mantel Score Significance* (MSS) [2] measure similarity between sounds and vocal imitations by correspondences between both acoustic spaces. We refer to these as *acoustics-based* similarity metrics. On the other hand, the *Akaike Information Criterion* (AIC) [29] and the prediction *accuracy* measure the capacity of feature sets to model and predict human listeners' drum-imitation similarity scores [1]. We refer to these as *perception-based* similarity metrics.

Acoustics-based Metrics

For a collection of N vocal imitations per imitator (18 in our case), the Mean Reciprocal

Rank (MRR) is defined as:

$$\mathrm{MRR} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{\mathrm{rank}_n} \tag{7.3}$$

where $rank_n$ is the rank of the relevant reference drum sound for the n-th imitation. As the MRR is inversely proportional to the rank number, it acts as a penalisation factor in a simulated DSRV task: the more delayed the retrieval of the correct sample, the lower the MRR score.

The second acoustics-based metric, the Mantel Score Significance (MSS), measures the degree of global correspondence between the acoustic space of drum samples and that of vocal imitations for all imitators. To calculate it, we ran the Mantel test [2] (see section 7.1.2) on the individual features that compose embeddings and reported the percentage of statistically significant Mantel scores (p<0.05). We ran 5 Mantel tests per feature and later averaged these results.

Perception-based Metrics

We followed the same procedure as in Mehrabi et al. [1] to calculate the two remaining metrics. In this study, 63 human listeners rated the perceived similarity between a vocal imitation and same-category drum sounds within the MDV dataset. They were presented with 30 test pages (trials), of which 28 were unique and 2 were random duplicates used to assess listeners' reliability. The listeners that were able to replicate their responses for at least one of the duplicates with a Spearman rank correlation coefficient of $\rho \geq 0.5$ were considered reliable. In this study, we got 51 reliable listeners and a total of 5,532 similarity ratings. For each feature set, the Euclidean distance was measured between each of the 252 imitations and their respective 6 within-class sounds, giving 1512 distance values. We calculated the percentage of imitated sounds for which the fitted slope of the regression model $rating \sim distance$ was significantly below 0. This is an implicit measure of *accuracy*, with a negative slope meaning an inverse relationship between distances and similarity ratings. A slope's value is significantly below 0 if the upper 95% confidence interval (CI) is negative.

One of the limitations of the regression model above mentioned is that it only takes sounds' features and similarity measures into account. It, therefore, fails to capture the influence of other determinant factors like which imitator produced the current sound or whose listener is the current similarity score. To account for these idiosyncrasies, we fitted a Linear Mixed-Effect Regression (LMER) model using R's lme4 package [30] which was specified as $rating \sim distance * sound + (1|listener/trial) + (1|imitator)$, with the similarity *rating* as the response variable, fixed effects of *distance* and drum *sound* with an interaction term, and random intercepts for *listener*, *imitator*, and *trial* (nested in listener). We used the *Akaike Information Criterion* (AIC) as a metric to evaluate model fitness. Given two models, the one with the lowest AIC displays less information loss and is, thus, a better fit. While the absolute value of the AIC has no statistical meaning, a practical heuristic to compare scores is to consider two models with $\Delta_{AIC} > 10$ as one fitting the data significantly better than the other [31].

Finally, note that results for these two perception-based metrics are not directly comparable to those in [1], as we did not analyse toms and cymbals, used a lower embedding size, used a different dataset containing instrument labels, and averaged results over five models per method.

7.2.2 Results and Discussion

Applying the modifications in 7.2.1.3 to the baseline model (CAE-B) [1], we found that the model benefited from (i) an increase in the number of filters from [8,16,24,32] to [8,16,32,64], (ii) a change in internal filter sizes from 10x10 to 9x9, which avoids aliasing from odd dimensions, and (iii) max-pooling operations in the encoder. The use of a variational latent space, an increased depth of the networks, and the use of transposed convolutions instead of upsampling layers in the decoder had a minor effect on performance while sometimes lowering the quality of sample reconstructions. Hence, we kept the regular latent space and upsampling layers in our proposed model (CAE).

Results for all evaluation metrics are displayed in Table 7-B. We see that the four proposed CAE models' results equal or surpass the ones from the baseline CAE-B model. Also, conditional models tend to perform better than their non-conditional counterpart (CAE) and the CAE-SDL model performs better than or like the rest of methods in all metrics.

	Baseline (7.2.1.2)		Proposed (7.2.1.3)			
	HS	CAE-B	CAE	CAE-SL	CAE-DL	CAE-SDL
MRR	34.7	39.6 ± 1.4|1.2	**42.1** ± **1.2**|**1.0**	41.2 ± 1.1|1.0	41.4 ± 1.5|1.3	41.7 ± 0.7|0.6
MSS	27.6	44.2 ± 3.8|3.3	43.0 ± 3.5|3.1	44.8 ± 3.4|3.0	45.0 ± 2.5|2.2	**47.5** ± **2.8**|**2.5**
Acc.	**55.6**	43.3 ± 2.2|1.9	47.8 ± 4.4|3.9	48.9 ± 6.5|5.7	47.8 ± 6.6|5.8	54.4 ± 2.1|1.9
AIC	999	658 ± 58|51	625 ± 61|53	609 ± 64|56	613 ± 59|51	**576** ± **66**|**58**

Table 7-B: Results for the four evaluation metrics. Error is expressed in terms of standard deviation (left) and 95% confidence intervals (right) for five iterations. MRR is expressed in terms of percentage and AIC values are subtracted a constant value to improve readability. Numbers in bold indicate best performances for the related metric. Notation: HS = heuristic set, CAE = convolutional autoencoder, -B = baseline, -SL = sound-level labels, -DL = drum-level labels, -SL = sound and drum-level labels.

Regarding *acoustics-based* similarity metrics, MRR and MSS, CAE embeddings performed consistently better than the heuristic feature set. Similar results were observed in [25] for MRR, where learnt features from a stacked autoencoder performed better than a set of 13 MFCCs and their first and second derivatives. We also noted relatively high MSS scores for CAE embeddings, reaching 47.5% in the case of CAE-SDL. This means that statistically speaking, a generic imitator would dexterously reproduce with the voice 15 out of the 32 CAE-SDL features as heard in the reference drum sounds.

Regarding *perception-based* similarity metrics, accuracy and AIC, we find that deep embeddings from CAE models perform significantly better than heuristic features in AIC but the opposite is true in the case of the accuracy metric. This is a rather surprising result considering the higher accuracy scores by learnt features in [1], which could be due to the differences in the experimental setting between the study and ours (see section 7.2.1.4). AIC scores from heuristic features are far from CAE-related ones, as also observed in [1], which might mean that heuristic features do not benefit as much as CAE embeddings from knowing about the current listener, imitator, drum sound, and trial variables when computing similarity. This could be due to the fact that CAE embeddings are learnt using datasets containing a wide range of vocal percussion and drum sounds from various performers while heuristic features are extracted from sounds in a context-agnostic way. We also see that the CAE-SDL accuracy score (54.4%) is relatively close to that of the heuristic set (55.6%). The CAE-SDL's AIC score is also the lowest one, although 95% confidence intervals from AIC scores highly overlap with each other, meaning that we cannot ensure the superiority of one method over the rest. Increasing the number of models per method might clear doubts in this respect.

Apart from results in Table 7-B, we also trained the CAE-B model with the dataset used in the original study [1], which gave scores of 39.2|1.2, 45.8|3.7, 48.9|6.5, and 579|94 for MRR, MSS, accuracy, and AIC respectively. It, therefore, outperformed the CAE-B trained with our dataset in perception-based metrics but did not outperform the best-performing model (CAE-SDL) in any metric. Also, all methods significantly outperformed random feature vectors (19.3|0.9, 4.7|1.6, 7.8|3.9, and 1035|129).

In summary, results show that conditioning DSRV systems on sound- and drum-type labels is very likely to augment the degree of similarity between imitations and retrieved samples. We believe these results can be extrapolated to general query by vocal imitation to some degree and that query-by-vocal-imitation algorithms can potentially benefit from this kind of conditioning, including metric learning systems trained with sample-imitation pairs [27].

Finally, we also plotted the mean Mantel scores and p-values from the five CAE-SDL embeddings in Figure 7.5. In comparison[2] with the results for the feature set in Section 7.1.3 (see Figure 7.4), we see how the mantel scores associated with CAE-SDL features are generally higher than those relative to the heuristic set while their p-values are lower, which is also reflected in their MSS scores in Table 7-B. We also found that, while some learnt features are imitated skilfully by all imitators, many other features find notable disagreements between imitators. This further supports the idea of approaching DSRV via user-based systems [32], which could fine-tune retrieval scores to specific imitators by taking their vocal imitation styles into account.

[2]A direct comparison cannot be made, as here we are analysing 18 of the 30 reference sounds.

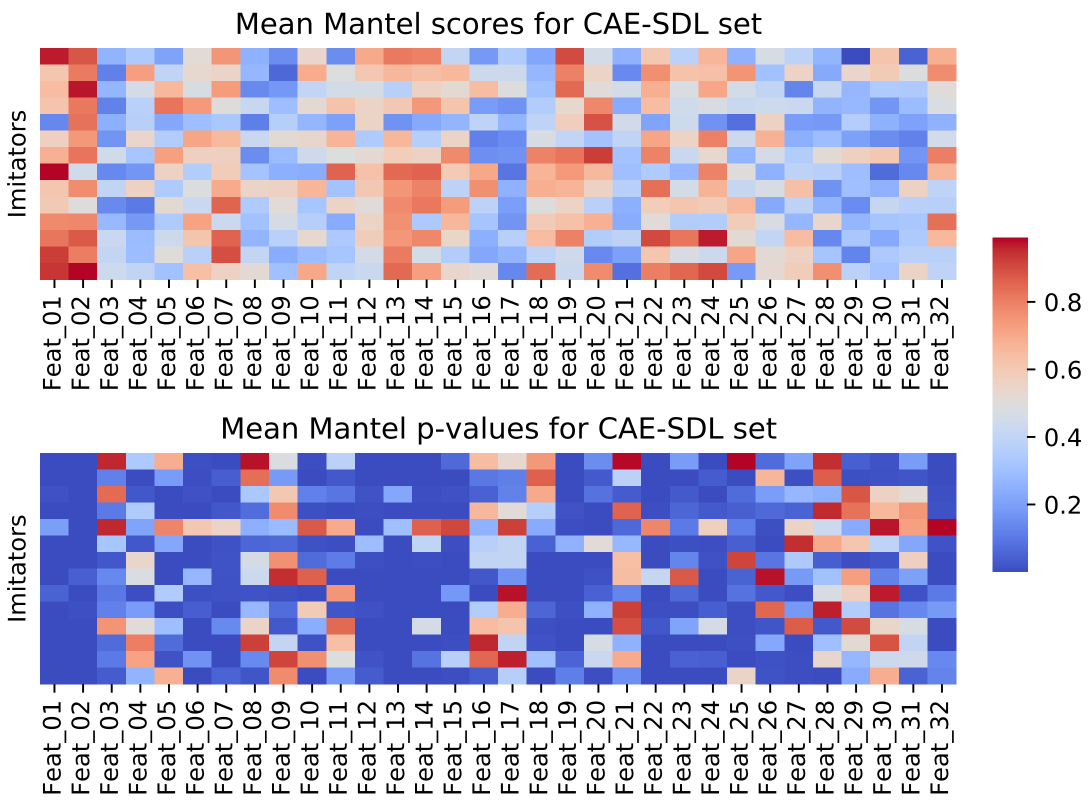


Figure 7.5: Mean Mantel scores and p-values (5 iterations) for each imitator using the features in the CAE-SDL embeddings. DerAM = envelope's derivative after maximum; Loud = loudness; SCent = spectral centroid; m = mean; s = standard deviation; d = derivative.

7.3 Summary

In this chapter, we have explored techniques to approach DSRV from a data-driven perspective. We conducted several analyses on the MDV dataset for the first part of the chapter. These analyses were aimed at discovering timbral relationships between drum samples and vocal imitations and informing data-driven routines on how to construct audio input representations for network algorithms. In the second part of the chapter, we used the insights from the previous study to construct a data-driven feature learning system based on conditional auto-encoders.

In the first study, we observed that the derivative after the maximum of the sound envelope and the spectral centroid are good predictors of the acoustic space of drum sounds given that of vocal imitations and vice versa. This means that participants imitated these descriptors dexterously in their vocalisations. Conversely, the log attack time descriptor was not found to be a relevant descriptor in this respect. These discoveries implied that both time and frequency resolutions played an important role in describing the sounds in the timbre space. We also found that 1.5 seconds was an appropriate truncation length when calculating spectrogram representations for data-driven DSRV and that there exist relevant user differences in vocal imitation styles that encourage the use of user-based retrieval strategies.

In the second study, we evaluated the potential of several types of deep convolutional autoencoder models to learn useful feature sets for DSRV. We used four different interpretable metrics to investigate how well these embeddings could link vocal imitations with their reference drum sounds and found that models conditioned on both sound- and drum-type labels (CAE-SDL) excelled in both acoustics- and perception-based metrics. We also found that learnt features, like heuristic ones, are reproduced with different levels of skill across imitators, which further promotes retrieval fine-tuning by building user-based DSRV systems, perhaps based on these user-agnostic learnt embeddings.

References

[1] A. Mehrabi, K. Choi, S. Dixon, and M. Sandler, “Similarity measures for vocal-based drum sample retrieval using deep convolutional auto-encoders,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* Ieee, 2018, pp. 356–360.

[2] P. E. Smouse, J. C. Long, and R. R. Sokal, “Multiple regression and correlation extensions of the mantel test of matrix correspondence,” *Systematic zoology*, vol. 35, no. 4, pp. 627–632, 1986.

[3] D. Bogdanov, N. Wack, E. Gómez Gutiérrez, S. Gulati, H. Boyer, O. Mayor, G. Roma Trepat, J. Salamon, J. R. Zapata González, X. Serra *et al.*, “Essentia: an open source library for audio analysis,” *ACM SIGMM Records. 2014; 6 (1): 18-21.*, 2014.

[4] J. J. Burred, A. Robel, and T. Sikora, “Dynamic spectral envelope modeling for timbre analysis of musical instrument sounds,” *IEEE Transactions on Audio, Speech, and Language*

Processing, vol. 18, no. 3, pp. 663–674, 2009.

[5] M. Caetano, C. Saitis, and K. Siedenburg, "Audio content descriptors of timbre," in *Timbre: Acoustics, perception, and cognition*. Springer, 2019, pp. 297–333.

[6] A. De Cheveigné and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[7] F. De Leon and K. Martinez, "Enhancing timbre model using mfcc and its time derivatives for music similarity estimation," in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*. IEEE, 2012, pp. 2005–2009.

[8] H. Terasawa, M. Slaney, and J. Berger, "The thirteen colors of timbre," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.* IEEE, 2005, pp. 323–326.

[9] ——, "Perceptual distance in timbre space." Georgia Institute of Technology, 2005.

[10] A. Mehrabi, S. Dixon, and M. B. Sandler, "Vocal imitation of synthesised sounds varying in pitch, loudness and spectral centroid," *The Journal of the Acoustical Society of America*, vol. 141, no. 2, pp. 783–796, 2017.

[11] G. Lemaitre, O. Houix, F. Voisin, N. Misdariis, and P. Susini, "Vocal imitations of non-vocal sounds," *PloS one*, vol. 11, no. 12, p. e0168167, 2016.

[12] "Bfd3 drum library," https://www.fxpansion.com/products/bfd3/.

[13] A. Delgado, S. McDonald, N. Xu, and M. Sandler, "A new dataset for amateur vocal percussion analysis," in *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound*, 2019, pp. 17–23.

[14] D. Stowell and M. D. Plumbley, "Delayed decision-making in real-time beatbox percussion classification," *Journal of New Music Research*, vol. 39, no. 3, pp. 203–213, 2010.

[15] A. Ramires, R. Penha, and M. E. Davies, "User specific adaptation in automatic transcription of vocalised percussion," *arXiv preprint arXiv:1811.02406*, 2018.

[16] J. Zhu, J. Wang, J. Liu, and X. Zhang, "Study on the classification of beatbox sounds based on timbre features," in *2020 International Conference on Culture-oriented Science & Technology (ICCST)*. Ieee, 2020, pp. 543–545.

[17] L. Nanni, G. Maguolo, and M. Paci, "Data augmentation approaches for improving animal

audio classification," *Ecological Informatics*, vol. 57, p. 101084, 2020.

[18] J. O. Smith and J. S. Abel, "Bark and erb bilinear transforms," *IEEE Transactions on speech and Audio Processing*, vol. 7, no. 6, pp. 697–708, 1999.

[19] E. Terhardt, "Calculating virtual pitch," *Hearing research*, vol. 1, no. 2, pp. 155–182, 1979.

[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[21] Y. Choi, M. El-Khamy, and J. Lee, "Variable rate deep image compression with a conditional autoencoder," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3146–3154.

[22] S. Dahmani, V. Colotte, V. Girard, and S. Ouni, "Conditional variational auto-encoder for text-driven expressive audiovisual speech synthesis." in *Interspeech*, 2019, pp. 2598–2602.

[23] B. Chettri, T. Kinnunen, and E. Benetos, "Deep generative variational autoencoding for replay spoof detection in automatic speaker verification," *Computer Speech & Language*, vol. 63, p. 101092, 2020.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] Y. Zhang and Z. Duan, "Retrieving sounds by vocal imitation recognition," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*. Ieee, 2015, pp. 1–6.

[26] ——, "Imisound: An unsupervised system for sound query by vocal imitation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Ieee, 2016, pp. 2269–2273.

[27] Y. Zhang, B. Pardo, and Z. Duan, "Siamese style convolutional neural networks for sound search by vocal imitation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 2, pp. 429–441, 2018.

[28] M. Müller, *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015.

[29] H. Bozdogan, "Model selection and akaike's information criterion (aic): The general theory and its analytical extensions," *Psychometrika*, vol. 52, no. 3, pp. 345–370, 1987.

[30] D. Bates, M. Mächler, B. Bolker, and S. Walker, "Fitting linear mixed-effects models using

lme4," *arXiv preprint arXiv:1406.5823*, 2014.

[31] D. Anderson and K. Burnham, "Model selection and multi-model inference," *Second. NY: Springer-Verlag*, vol. 63, no. 2020, p. 10, 2004.

[32] B. Gong, M. Kaya, and N. Tintarev, "Contextual personalized re-ranking of music recommendations through audio features," *arXiv preprint arXiv:2009.02782*, 2020.

Chapter 8

Conclusion

In this thesis, we have explored the task of Query by Vocal Percussion (QVP) from a data-driven perspective. We have conducted experiments in two of the most important subfields in QVP: Vocal Percussion Transcription (VPT) and Drum Sample Retrieval by Vocalisation (DSRV). The former aimed at detecting and classifying vocal percussion sound events in both online and offline scenarios while the latter aimed at finding the best set of audio features to link drum sounds with their vocal imitations.

We conclude our study in this last chapter by outlining its main findings and discussing the challenges to overcome in order to advance QVP.

8.1 Summary of Findings

In this thesis, we have dedicated one chapter to introduce the topic of QVP (Chapter 1), one chapter to provide theoretical and referential background (Chapter 2), one chapter to introduce the audio data and annotations used in this thesis (Chapter 3) and a total of four chapters to carry out the experiments in QVP (Chapters 4, 5, 6, and 7).

In Chapter 3, we presented several publicly available vocal percussion datasets and detailed the creation of several other datasets to ensure that we had enough data to apply data-driven methods to QVP. The already publicly available datasets were the LVT dataset, containing 841 amateur vocal percussion sounds, the BTX dataset, containing 3,611 beatbox sounds, and the Mehrabi Drum Vocalisations (MDV) dataset, containing 420 vocal imitations of drum sounds. We realised that, while we arguably had enough beatbox sounds to train and evaluate VPT algorithms and enough vocal imitations to evaluate DSRV algorithms, we lacked enough amateur vocal percussion sounds to train and evaluate VPT algorithms. This led us to the curation of the Amateur Vocal Percussion (AVP) dataset, which contained vocal percussion recordings from 28 participants with little or no experience in beatboxing. The dataset was divided into the personal subset (AVP-P), containing 4,873 vocal percussion sounds, and the fixed subset (AVP-F), containing vocal percussion 4,905 sounds. In the AVP-P, participants were asked to imitate the drums in their own way, while in the AVP-F participants were asked to imitate the drums using specific syllables. It provides annotations for four drum instruments (kick drum, snare drum, closed hi-hat, and opened hi-hat) and for individual phonemes in the case of the personal subset. Apart from the AVP dataset, we also put together and annotated the VIS-P dataset (3,393 percussive vocal imitations by amateur performers) and the FSB dataset (4,296 beatbox sounds), which were used in onset detection experiments.

In Chapter 4, we explored several data-driven algorithms for vocal percussion onset detection, comparing their performance with baseline heuristic algorithms. We trained a BRNN, a CNN, and a CRNN for offline onset detection and several N-Frame RNN models for online onset detection. We discovered that these trained neural networks consistently outperformed heuristic baselines and some pretrained musical onset detection models in both the offline and the online case for all metrics (accuracy, time deviation from ground truth onsets, and inference speed). In particular, BRNNs and CRNNs performed best for offline onset detection, while the latter had a significantly lower inference time than the former, making it more recommendable for generic use cases. The accuracy measure for these models was near-perfect (F1-score of .965) while displaying a small deviation from real onsets (6.4 ms of absolute deviation). In the case of online onset detection, the RNN N-Frame models were the best-performing ones in terms of accuracy (F1-score of 0.9 at 20.3 ms delay) and speed (7.9 ms of inference time). We also noted that high accuracies (F1-score of 0.8) were already reached for 8.7 ms of delay, which means that vocal percussion onsets are usually recognisable at low delays. We recommend the usage of 5-Frame RNNs for online onset detection, although the choice between these algorithms or others would be contingent on the application context.

In Chapter 5, we carried out the experiments related to online vocal percussion classification. This was done for both amateur vocal percussion and beatbox and consisted of two related studies that were performed independently from one another. In the first one, we searched for an appropriate set of phonemes for online amateur vocal percussion classification using three criteria: the frequency of use of the phonemes, the spectral similarity of the phonetic sounds to reference drum sounds, and the classification separability of these phonetic sounds. The study concluded that the /p/, /k/, /t/, /ts/, and /tʃ/ phonemes were the most appropriate to carry out online amateur vocal percussion classification with. In the second study, various heuristic and data-driven algorithms were applied to online vocal percussion classification, using the five phonemes derived from the first study as input for amateur vocal percussion classification and the sounds contained in the BTX dataset for beatbox classification. The best-performing algorithm for both amateur vocal percussion and beatbox was a 1D-CNN with the Mel spectrum as input. We also found that the amateur vocal percussion phonemes selected in the first study accounted for the highest F1-score of .950 at 20.3 ms of time delay. We also got relatively low online beatbox classification accuracies, which may imply the need for more separable sounds and warrants a more in-depth look at this type of vocal percussion for online VPT.

In Chapter 6, we covered the offline case of vocal percussion classification. Like the last chapter, this also consisted of two independent but related studies. In the first study, we compared several heuristic and end-to-end data-driven algorithms for five tasks: user-based classification of amateur vocal percussion (AVP UB), user-agnostic classification of amateur vocal percussion (AVP UA), user-agnostic classification of amateur vocal percussion using the syllables in the AVP-F set (AVP UA Syll), user-agnostic classification of amateur vocal percussion using the phonemes derived from the first study of Chapter 5 (AVP UA Phon), and user-agnostic classification of beatbox sounds (BTX UA). We saw that, for amateur vocal percussion, the AVP

UA Syll and the AVP UA Phon strategies significantly outperformed The AVP UB and AVP UA strategies, which means that having a fixed set of sounds to classify generally makes algorithms yield better classification accuracies than having sets of sounds that are derived from personal choices. The performances for beatbox sounds (BTX UA) were similar to those derived from the online case, reinforcing the need for a deeper look into beatbox sound classification to obtain better accuracies. Also, the best-performing classifier for the AVP UB strategy (a CNN) needed around 3 minutes to train on a CPU, which could be slow enough to make it impractical in real-world scenarios. This realisation led to the second study, where we explored embedding learning as an alternative strategy to shorten this training time and improve the generalisation power of user-based classification approaches by minimising the chances of overfitting. Specifically, we supervised several CNNs on four types of label sets (instrument-level, syllable-level, phoneme-level, and sound-level) and used the 32 features in their penultimate layer as input to a KNN classifier. Results suggested that the embeddings that were extracted from the CNN model supervised with syllable-level labels were the most appropriate for user-based approaches, as they were the best-performing feature sets in terms of both accuracy and stability to training conditions. The embedding learning strategy also performed similarly to the best-performing end-to-end model for the AVP UB task, making embedding learning a fast and non-overfitting alternative for user-based vocal percussion classification. We also computed several saliency maps from these embedding learning models that showed how the spectrogram regions in the 1-to-2 kHz frequency band and those with an absence of energy seem to have special importance for the algorithms when classifying drum instruments in an end-to-end manner.

In Chapter 7, we carried out several experiments in DSRV using data-driven feature learning algorithms. In the first part of the chapter, we conducted several analyses on the MDV dataset to both search for timbral relationships between drum samples and vocal imitations and inform algorithms on how to build input representations for training. We saw that features like the derivative after the maximum of the sound envelope and the spectral centroid linked the acoustic space of drum sounds with that of vocal imitations significantly well, meaning that users imitated them skillfully. We also noted that the log attack time descriptor did not perform well in this respect. We also discussed how these two observations could imply that both time and frequency resolutions are important when describing the sounds in the spectral domain. Looking at the average spectrograms of drum sounds and vocalisations in the MDV dataset, we also found that an adequate truncation length in the time axis when calculating the spectrogram representations could be around 1.5 seconds. Finally, we confirmed that there are significant user differences in vocal imitation styles that could justify the use of user-based algorithms. In the second part of the chapter, we explored data-driven feature learning algorithms for DSRV using the previous insights as guidance when computing input spectrograms. In particular, we studied the suitability of several embeddings learnt via different types of deep convolutional autoencoder models for the task of DSRV. To evaluate how well these embeddings could link vocal imitations with their reference drum sounds, we used four different metrics, two of them acoustics-based (MRR, MSS), and two of them perception-based (Accuracy, and AIC) via listeners' similarity ratings of drum-imitation pairs. Deep convolutional autoencoders conditioned on both sound- and drum-type labels performed the best in all four metrics. We also saw how the features learnt by the best-performing model were better reproduced by imitators than the set of heuristic features.

8.2 Challenges and Perspectives

With the study contained in this thesis, we believe we advanced the state of the art in QVP to a certain degree and demonstrated that data-driven models like deep neural networks are powerful tools to reach high performances. Regularisation and data augmentation techniques like pitch shifting and time stretching were also proven to help algorithms generalise better, especially in regimes with low amounts of training data.

Results obtained in this thesis suggest that some of the tasks where we carried out our experiments could be considered, to a reasonable extent, solved. We believe this is the case for offline percussive onset detection, for instance, as the associated accuracy score is near perfect and the time deviation from the ground truth annotations is minimal. We were not overly surprised by this result, as monophonic percussive transients are usually well-recognised by the human eye in both the time and the frequency domains.

Likewise, we obtained great results on the task of user-agnostic vocal percussion classification with fixed sounds, achieving near-perfect accuracy scores for both offline and online classification. In this respect, we have argued that the selection of these fixed vocal percussion sounds is a crucial step to optimise accuracy, as one would have to ensure that the sounds are separable enough so that the classifiers are able to reach the maximum accuracy possible. Adopting the sounds relative to the phonemes and syllables /p/, /k/, /t/, /ts/, and /tʃ/ as the set to perform classification on, we reached high accuracy scores while ensuring that the sounds remain natural for the users to vocalise.

While the results above seem promising for QVP, we have also seen that many tasks are certainly not solved yet, which can inspire future advances in the field. In particular, we see four main unsolved problems: beatbox sound classification, user-based amateur vocal percussion transcription, online (real-time) percussive onset detection, and DSRV in general. In the paragraphs below, we offer several reasons why these problems are inherently hard to tackle and ideas about potential strategies to attack them in an effective way.

We see several idiosyncrasies that make beatbox sound classification an inherently difficult task to solve. Some of these particularities are (i) the high density of sound events in time due to its fast pace, (ii) the user-to-user timbral variance of within-class beatbox sounds due to stylistic differences, (iii) the inclusion of ill-defined labels like "miscellaneous" or "unsure of classification", and (iv) the relatively high number of labels. The first point of difficulty can be easily tackled by choosing a little enough hop size when calculating representations. If the second point negatively affects classification results significantly, the algorithms could benefit from a user-based classification approach. As beatbox sounds are generally similar enough to each other across users so to pursue user-agnostic classification strategies, algorithms could reach better accuracy scores if the models can be further fine-tuned to individual users. The third and fourth points of difficulty would require an informed consensus about the labelling of beatbox sounds and the classification process itself. For instance, the third issue could be easily avoided by excluding the problematic labels (e.g. "unsure of classification") when calculating classification accuracy. Also, the fourth issue could be solved by grouping labels into higher-level classes if the application context permits it (e.g. grouping “p-like snare” and “k-like snare” into a single “snare” class).

Regarding user-based amateur vocal percussion transcription, there are several challenges to overcome that relate to (i) the different vocal percussion styles among participants, (ii) the qualitative difference between train and test data, (iii) the selection of vocal percussion sounds that are hard to separate, and (iv) sound vocalisation inconsistencies. The first and second points of difficulty are inherent to the task and, thus, their negative effects on final performances would not disappear completely. However, there exist effective ways to mitigate these negative effects like, for instance, the recording of the train set in a “fixed phrase” manner using a vocal percussion pattern that is complex enough to resemble an improvisatory performance. The third and fourth challenges could be tackled, for example, by ensuring the selection of highly separable phonemes through prompted recommendations in the case of the former and asking users to record more examples of a sound class that the algorithm is finding hard to classify in the case of the latter.

With respect to online percussive onset detection, we see several challenges that include (i) avoiding the prediction of vowel coda phonemes in syllables, (ii) avoiding the prediction of breath sounds, (iii) avoiding the prediction of percussive ambient noises, and (iv) predicting joint onsets with no silence in-between. As the algorithms operate in a real-time regime, these challenges could sometimes be hard to overcome. For instance, breath sounds and some ambient noises have a noisy spectral profile that is similar to those from percussive onsets. Hence, while offline algorithms are able to take a look at a bigger part of the sound to decide if it is a relevant onset or not, online algorithms may be more prone to this kind of errors given their short analysis buffer. In the case of vowel coda phonemes, their usually harmonic structure could help algorithms discern between them and a relevant vocal percussion onset, so that case is not likely to be as problematic as the earlier ones. Lastly, predicting joint onsets accurately in time could largely benefit from non-algorithmic fixes like, e.g., better diction from the users.

Finally, the main challenges associated with DSRV that we see are (i) the low amount of publicly available drum-imitation sound pairs, (ii) the difficulty of imitating same-category drum sounds in a consistent manner due to the usually subtle timbral differences between them, and (iii) the different imitation styles of users. The first issue can be simply tackled by recording a dataset with drum-imitation pairs that is big enough to apply more refined data-driven algorithms like those based on metric learning (e.g. siamese networks). That way, the learnt embeddings could potentially be more informative and the retrieval accuracy could be higher. The second challenge is inherent to the task and suggests that hearing and imitation skills might be decisive when assessing which users are likely to benefit most from DSRV systems, raising the question of whether DSRV systems are potentially useful tools for the general public or just for a few people. Finally, and similarly to the case of beatbox sound classification, the third point of difficulty is likely to benefit from user-based algorithms that fine-tune predictions via techniques like active learning. That way, the DSRV system could be able to refine its own predictions by strategically prompting the users the drum samples that it finds challenging to match so that they imitate them vocally.

All in all, we believe that the challenges associated with QVP that we have observed in this thesis could be overcome to a reasonable extent by trying out new strategies that include the recording of more data, the selection of the learning strategy, and the interaction with the user among others. If successful, they would facilitate the inclusion of QVP systems in music production environments as reliable tools to assist artists in the creative process.

Appendix A

Author's publications

1. **A. Delgado**, S. McDonald, N. Xu and M. Sandler, "A New Dataset for Amateur Vocal Percussion Analysis," *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound*, Nottingham, United Kingdom, 2019.

2. **A. Delgado**, S. McDonald, N. Xu, C. Saitis and M. Sandler, "Learning Models for Query by Vocal Percussion: A Comparative Study," *Proceedings of the 46th International Computer Music Conference, ICMC*, Online, 2021.

3. **A. Delgado**, C. Saitis and M. Sandler, "Spectral and Temporal Timbral Cues of Vocal Imitations of Drum Sounds," *Proceedings of the 2nd International Conference on Timbre*, Online, 2020.

4. **A. Delgado**, C. Saitis and M. Sandler, "Phoneme Mappings for Online Vocal Percussion Transcription," *151st Convention of the Audio Engineering Society*, Las Vegas, NV, and Online, 2021.

5. **A. Delgado**, E. Demirel, V. Subramanian, C. Saitis and M. Sandler, "Deep Embeddings for Robust User-Based Amateur Vocal Percussion Classification," *Proceedings of the 19th Sound and Music Computing Conference*, Saint-Étienne, France, 2022.

6. **A. Delgado**, C. Saitis, E. Benetos, and M. Sandler, "Deep Conditional Representation Learning for Drum Sample Retrieval by Vocalisation," *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Rhodes Island, Grece, 2023 (submitted).

Appendix B

BFD-VRT: A Dataset to Test Robustness of Drum Transcription Algorithms

About

This study was done in collaboration with Fred Bruford, Vinod Subramanian, SKoT McDonald, and Mark Sandler in 2020. We include it as an appendix section because the dataset that it references was never published due to issues related to intelectual property. The issues were mostly due to the sudden purchase of the dataset by another company and to a change of priorities regarding the mentioned BFD Player application, which has not been released yet neither by the current nor the previous owner of the dataset as in August 2022. Despite this, we think that the work could be of interest to research in Automatic Drum Transcription (ADT) among other fields.

Abstract

We present BFD Vintage Recording Techniques (BFD-VRT), a new dataset to test the robustness of automatic drum transcription and source separation algorithms to different production environments. BFD-VRT has a core library consisting of 960 professionally recorded multi-channel drum samples, including 60 articulations from 14 drum instruments, 16 velocity layers, 27 microphone channels, and 2 processed channels. We show how this content can be used with the BFD Player application to generate custom drum datasets. This way, researchers are able to import symbolic drum patterns, adjust recording and post-processing parameters as desired, and synthesise datasets on which to test their algorithms. We provide Python scripts and a Lua API to facilitate this process. In the final evaluation section, we compare the robustness of five open-source automatic drum transcription algorithms to different production techniques by running them on fifteen of these generated datasets. The BFD-VRT dataset and the code for this paper are freely available.

B.1 Introduction

The study of drum sounds and its applications in Music Information Retrieval (MIR) has been recently led by two main disciplines: Automatic Drum Transcription (ADT) and Drum Source Separation (DSS) [1]. The former attempts to localise and classify drum events in audio recordings [2] while the latter seeks to isolate the original drum signals from audio mixes [3].

The most popular algorithms for both ADT and DSS are based on end-to-end deep learning techniques [4][5][6]. These neural network models learn representations of drum performances directly from their spectrogram or audio waveform without the need for extracting engineered features as an intermediate step. Results obtained from these data-driven approaches are directly dependent on the quantity and the variety of the data available. In general terms, the larger and more representative the training/validation dataset is, the better the generalisation capabilities of the final classifier are expected to be.

Researchers in ADT and DSS may not always have access to such datasets. This is particularly true in scenarios where the drum audio files are processed using different techniques. There are many possible ways for a music producer to process drum kit recordings, such as applying gain balancing, panning, dynamics processing, equalisation, and adding reverberation and distortion. These processing styles are often genre-dependent. Most ADT and DSS systems in literature are trained and evaluated on balanced mixes from signals captured by different microphones, usually not accounting for any other pre- and post-processing routines that may alter the timbre of the instruments and affect their place in the mix. Neglecting these issues can ultimately cause the algorithms to fail in their predictions when faced with unseen real-world contexts. On this note, quantifying how robust the models are to these contexts is still a challenge in music transcription as a whole.

We release the BFD Vintage Recording Techniques (BFD-VRT) dataset as a new evaluation framework to test ADT and DSS algorithms on realistic production scenarios. It contains a collection of professionally produced drum samples that allows ADT and DSS researchers to create drum patterns under the production conditions of their choice. This can be done by using the dataset jointly with the BFD Player application[1], a free ”player” edition of the BFD virtual drum kit instrument that synthesises realistic sounding drum performances from symbolic drum patterns and the BFD-VRT dataset. An easy-to-use scripting environment for generating audio data is also provided.

In this paper:

- We discuss the main limitations of public drum datasets and review past attempts to overcome them.
- We provide a detailed overview of the BFD-VRT dataset, with information on its samples, mixing presets, metadata, and rendering process.
- We list some alternative use-cases for the BFD-VRT dataset in MIR apart from ADT and DSS.

[1]https://www.fxpansion.com/products/bfd3

- We use the BFD-VRT dataset to compare the robustness of five open-source ADT algorithms.

B.2 Limitations of Public Drum Datasets

The most popular drum datasets for ADT and DSS are ENST-Drums [7] and IDMT-SMT-Drums [8]. These are two sets of multi-channel drum recordings including single hits, common patterns, and sometimes accompaniment recordings (ENST-Drums) and synthesized drum sounds (IDMT-SMT-Drums). While both datasets provide clean drum recordings with precise annotations, they may sometimes impose important limitations on the generalisation capabilities of ADT and DSS models. In a recent review of ADT [2], authors identify significant shortcomings of current public drum datasets and discuss how they could affect the performance of end-to-end models that have been trained on them. Some of these limitations refer to the insufficient data diversity within them while others refer to their inappropriately small size for some taks like source separation.

The *lack of data diversity* is arguably the most important limitation. Popular drum datasets only contain enough representative data of a few musical genres, hit articulations, playing styles, and production conditions. Not accounting for the large variety of drum performances that exist in the real world can cause data samples to be too similar to each other (often redundant) and make algorithms unable to perform well outside the training data. Some attempts to overcome this problem have involved the incorporation of playing techniques to the training phase. Studies on timbral variations of snare drums [9], cymbals [10], and rack and floor toms [11] have been carried out in the past with views to improve the generalizability of ADT systems. While tested algorithms achieved good performances for solo drum kit recordings, it has been shown that their accuracy drops significantly when background music is present in the audio files [12].

The *insufficient size* of the datasets is also seen as a big problem, especially in DSS [13]. This is mostly due to the large amount of training parameters that deep neural networks have, which are in need of a large amount of data to make the right inferences. If the amount of training samples is considerably lower than the number of trainable parameters in the model, some undesirable phenomena like data overfitting might emerge at the evaluation stage. To deal with this lack of data in drum sound libaries, a popular technique that has been tried recently is data augmentation [14] [15]. Results using this strategy showcase how the generation of new audio files by applying post-processing effects to original dataset samples can make algorithms generalize better.

The issues discussed here raise concerns about how well ADT and DSS models trained on current public drum datasets would actually perform in unseen real-world scenarios. We believe the BFD-VRT dataset would help researchers in ADT and DSS to better quantify the robustness of their algorithms in face of these limitations.

B.3 Dataset

The BFD-VRT dataset consists of a detailed sampling of a drum kit, with one-shot multi-channel samples for strikes on individual kit instruments recorded for numerous articulations and velocities. Using the free BFD Player plugin, this sample set may be sequenced by symbolic data in MIDI or BFD Groove format to render stereo audio loops. Within BFD Player it is possible to control mixing and effects processing for the rendering, as well as articulation, velocity, and other parameters. Using a simple scripting environment provided with the dataset, the process of rendering stereo audio files of drum loops can be automated. These features mean that large datasets of realistic, studio-quality drum loops with matching ground truth data may be easily generated from symbolic data, under controlled rendering conditions. It is this control of rendering conditions that enables generation of datasets for robustness testing.

B.3.1 Recording

The samples contained within the BFD-VRT dataset were recorded by in-house FXpansion recording engineers in a professional standard drum recording studio, with a professional drummer. The microphones used in the recording, along with kit instruments and mic placements are shown in **Table 3**. Aside from direct microphones, three mono room mics are used, along with 4

Quantity	**Value**
Mean Sample length	2.11s
Max Sample length	10s
Total number of kit pieces	14
Total articulations	60
Velocity layers per articulation	16
Total number of samples	960
Total number of channels per sample	28
Dataset size (BFDLAC)	2862MB
Dataset size (WAV)	7120MB

Table 2-A: Key statistics for all samples in dataset

overhead pairs, a sub mic, 2 ambient stereo pairs and a processed reverb pair. The overhead pairs consist of an ORTF pair, Glyn Johns method spaced pair, a regular spaced pair, and a coincident pair of fig-8 ribbon microphones forming a Blumlein array. There is a total of 30 microphone channels and 2 digital reverb channels, for a total of 32 channels. Of these, 28 or 29 will present in any one kit piece. All kit pieces' direct mics apart from the cymbals provide ”bleed” signals for all hits. The drum recording microphone techniques used are described in [16].

B.3.2 Samples

Information concerning the samples contained in dataset is shown in **Table 1**. Samples are available in two file formats, multichannel WAV and BFDLAC, BFD's native lossless data compression format for multi-channel data [17]. Samples names are labeled according to their velocity, with low numbers quietest and high numbers loudest (e.g. ”master01.bfdlac” would be the quietest velocity sample for a given instrument articulation). Samples are grouped in folders for each articulation separately. The number of different articulations for each instrument is found

Instrument	**Number of Articulations**
Kick	1
Snare	6
Toms	3
China	2
Hi-hat	12
Other Cymbals	3

Table 2-B: Total number of articulations for each kit piece

Source	**Brand and Size**	**Microphone Placing**	**Microphone(s)**	**Delay**
Kick	Ludwig 22”x14”	In	AKG D112	3.8
		Out	Shure Beta52	3.4
Snare	Ludwig 14” x 5” 400	Top	Shure SM57	0.3
		Bottom	SE Electronics SE3A	0.1
		Top 2 (Condensor)	SE Electronics SE3A	0.5
Hi-hat	Zildjian 14” Quick Beat	Top	Neumann KM184	3.1
Floor Tom Low	Ludwig 18” x 16”	Top	Sennheiser MD421	3.0
Floor Tom	Ludwig 16” x 16”	Top	Sennheiser MD421	2.0
Mid Tom	Ludwig 13” x 19”	Top	Sennheiser MD421	2.1
High Tom	Ludwig 12” x 8”	Top	Sennheiser MD421	1.4
Splash	Zildjian 12” K	Top	SE Electronics SE3A	
China	Zildjian 19” K	Top	SE Electronics SE3A	
Crash	Zildjian 16” A Custom	Top	SE Electronics SE3A	
Ride	Zildjian 22” A Custom	Top	SE Electronics SE3A	
Sub mic	-	Front kit	JBL speaker	6.7
Ambient	-	Reverb	2x DSP / Processed	12.4
Room Mono	-	Far room	Oktava MK-220	14.3
	-	Side kit	Oktava MK-220	7.3
	-	Front kit	Neumann U87 Fet	3.8
Room Stereo	-	Far Room	2x Ribbon	12.0
	-	Room	2x Ribbon	11.5
	-	Ribbon		
	-	Blumlein Array	2x Ribbon	16.0
	-	Glyn Johns spaced pair	2x Neumann TLM103	5.3
	-	ORTF Pair	2x AKG C451B	6.1
	-	Regular spaced pair	2x AKG C414 XLS	6.5

Table 2-C: List of all sound sources in the BFD-VRT dataset. Microphone group delay times are given for a snare centre hit. For snare hits, there are no bleed signals supplied from the cymbal microphones. The relative *delay* (in milliseconds) to the onset in each channel is caused by the distance in 3D space of all microphones relative to the snare drum’s primary direct *Top* mic.

in **Table 2**. The BFD-VRT dataset sounds have their delay tails capped at 10 seconds for compactness.

B.3.3 Metadata

Detailed metadata for individual kit pieces are provided within the folder structure alongside the samples. For each kit piece, there is an XML file (entitled BFDInfo) providing detailed information about the kit piece, such as the manufacturer, dimensions, year constructed, type of beater or drum stick used, construction materials, audio channel structure and a photo of the kit piece. All metadata was documented by FXpansion engineers and developers at the time of recording of the dataset and is known to be accurate.

B.3.4 Groove Files

In addition to support for MIDI files, BFD's native Groove files may be used to sequence the drum samples for rendering audio drum patterns. BFD's Grooves are symbolic drum patterns recorded by human drummers. They are mostly unquantized and can trigger all velocities and articulations. These are in an XML-based format, grouped in ”bundles” of around 20-40 similar grooves of the same style and genre. Metadata is included in addition to the drum patterns themselves, such as the original tempo they were recorded at (Grooves can be rendered at any tempo), time signature, and articulations used. A set of 142 Grooves that were used in the evaluation are provided with the sample dataset. These are described in **Section 5.1**.

B.3.5 Mixing and Effects

The BFD-VRT dataset provides 15 mixing presets for rendering drums. These presets can be customised via mixer and engineering macro knobs within BFD Player, giving access to a wide range of mixing possibilities, including channel balancing, EQ and dynamics processing, reverb, distortion. The fifteen mixing presets provided are described in **Table 4**. The Macro Knobs in BFD Player provide high-level sound design, linking to many underlying engineering and effects parameters provided by the full BFD software.

Preset Name	**Description**
Blue Funky	Compression on all direct mics, EQ and drive on kick.
Blumlein	Blumlein array overheads and direct mics.
Direct Microphones Only	No overheads or ambient microphones. No effects.
Far Room Microphones	All direct mics and far room mics. No overheads. No effects.
Glyn Johns Technique	Direct kick and snare mics and Glyn Johns overheads spaced pair. No effects.
Mainly Mono Mics	Direct kick snare and hi-hat. Three mono room mics. No effects.
Modern Heavy	All direct mics and far room mics. Heavy compression, distortion and EQ.
One Mic	Mono feed from single room mic in front of drummer. No effects.
ORTF	ORTF stereo overheads and all direct mics. No effects.
Pick N' Mix	Direct kick snare and hi-hat mics. Mix of all overheads. Heavy compression and drive.
Pumped Up Monos	All direct mics, spaced pair overheads and three mono room mics. Compression.
Ribbon Time	Reverb on all direct mics. EQ and drive on room mics.
Slow and Heavy	Direct drums and hi-hat mics. Glyn Johns overheads. EQ, reverb, compression and distortion.
Spaced Pair	All direct mics and regular spaced pair overheads. No effects.
Sub Microphone	All direct mics and spaced pair with added sub mic on kick. No effects.
Vintage Crushed	Direct mics on drums and hi-hat. 2 room mics and mixed overheads. Heavy compression and distortion.

Table 2-D: Mixing presets in the BFD-VRT dataset

B.3.6 Scripting and Automation

A scripting environment for BFD Player provides functionality for rendering large numbers of audio files from symbolic data, with matching ground truth data for dataset generation. A set of Python functions call commands from within a Lua API that communicates with BFD from a command line environment. A full guide to using this environment can be seen in [link will be added after review].

Key functions of this scripting language include loading and saving presets, loading Groove or MIDI files, muting channels, adjusting mixer or effects parameters, setting the tempo, rendering audio, and saving Groove or MIDI files as ground truth data. With this interface, it is simple to automate the generation of audio drum patterns from symbolic data under various controlled conditions. This way, datasets can be generated for the same symbolic patterns but with different mixing and processing techniques, with or without certain instruments, humanization (random variations in onset timings or velocities), or other parameters. An example use case is provided in the next section, where the same set of symbolic drum patterns are rendered as audio for a number of different mixing styles with the aim of testing the robustness of ADT algorithms. Example code is provided [link will be added after review] for generating the datasets used in this paper.

For instance, the following is an example of a simple python script calling Lua commands within BFD. It loads the preset *Blue Funky* with a groove *waltz.mid*, sets the tempo to *180*, and renders it to the wav file *top10.wav*.

```
lua2BFD("loadPreset(bfd, 'Blue Funky')")
lua2BFD("loadGroove(bfd, 'waltz.mid')")
lua2BFD("setTempo(bfd, 190)")
lua2BFD("exportGroove(bfd, 'top10.wav')")
```

B.4 Other Research Topics

The BFD-VRT dataset will contribute to research topics beyond drum transcription and drum source separation. Given that the BFD-VRT dataset provides new options for generating drum grooves, it opens new avenues for research and analysis. In this section, we will present a non-exhaustive list of topics for research that this dataset will contribute to.

B.4.1 Analyzing drum articulation

Playing technique datasets [9, 11] are usually comprised of isolated drum sounds. The drawback is that it is not reflective of how different playing techniques occur in the real world.

The BFD-VRT dataset allows the incorporation of different playing techniques into the Grooves to create realistic drum patterns and allows different mixing conditions to further emulate the diversity of drum sounds in the real world. This would serve as a bridge between evaluating systems on isolated drum sounds and real drum recordings.

B.4.2 Generalisability

A common problem with machine learning algorithms is the ability to estimate how well they perform on unseen data in the real world.

A simple way to evaluate the generalisability of the system would be to partition the dataset based on the mixing presets so that there is no overlap of the presets between the training and test data. Then evaluating the performance on the test data gives a more realistic understanding of how the system would behave in unseen scenarios.

Another approach for generalisability is through domain adaptation [18]. Creating features that are invariant to the mixing conditions would increase the likelihood of good performance in unseen mixing conditions. In music, this idea has been explored in the context of transposition invariance for audio-to-score alignment [19]. A similar approach of using gated autoencoders might be a good starting point for mixing invariance in drum transcription.

B.4.3 Automatic Mixing

Typically automatic mixing of music focuses on level, panning, EQ, compression, and reverb. As De et al. [20] point out there is a constant need for trustworthy labeled data to perform automatic mixing. In automatic drum mixing work [21] it is noted that machine learning approaches are gaining popularity and that suitable data is needed. In addition, there is a need for more mixing parameters.

The BFD-VRT dataset provides an easy setup to generate data for different types of mixing parameters. The dataset would be suitable for machine learning approaches too.

B.4.4 Explainability of Drum Transcription

Debugging any deep learning system is tricky because they are black boxes. What can be changed is the parameters of the input in order to verify certain assumptions about the system.

For example, if the model architecture used is a convolutional recurrent neural network and there are concerns that the model is overfitting to pop genres. BFD-VRT can be used to generate uncommon rhythms, which the model can evaluate on to give a sense of whether the model overfits to pop.

The ability to control so many parameters of the BFD-VRT with precision allows for analysis where parameters of the input can be gradually changed until the system breaks. Ultimately improving the ability to analyze these drum transcription systems.

B.5 Evaluation

In this section, we test the robustness of five open-source pretrained ADT algorithms to various recording conditions and post-processing effects. This is intended to show the potential of the BFD-VRT dataset to evaluate ADT models, as well as comparing the generalisability capacity of the tested algorithms.

B.5.1 Data

To evaluate these models on different drum production scenarios, we synthesise fifteen datasets based on the fifteen presets included in the BFD-VRT dataset. These presets encompass different realistic settings of both recording techniques and post-processing effects. **Table 4** describes these presets.

The datasets are rendered using the set of drum loops featured in [22], which consist of 142 loops found within the BFD Groove library. They are drawn from a diverse range of styles, with an equal split from each of these 8 genre groups: Rock, Metal, Jazz, Dance/Hiphop, Blues/Country, Latin/Reggae, Funk, and Pop. All loops are 2 bars long, in 4/4 time, and rendered at 120bpm. Some of them are swung or contain triplet rhythms. As they are recorded by drummers, they are unquantized and contain variable velocity for each hit.

B.5.2 Models

We prepare five models for evaluation: NMF [23], ADTLib [24], CNN, CRNN and BRNN [15]. These are all open-source algorithms that have been trained to predict three instruments: bass drum, snare drum, and hi-hat.

The NMF model is described in [23]. Audio spectrograms of isolated drum sounds are used to pre-train a dictionary matrix that is then used in a Partially Fixed Non-Negative Matrix Factorisation (PFNMF) algorithm. The PFNMF model analyses music signals and updates the original dictionary matrix accordingly. The final activation matrix derived by this process contains the predicted onsets and labels. Input spectrograms are calculated using a frame hop size of 11.6 ms.

The ADTLib model is based on the method described in [25]. It consists of a convolutional neural network that extracts features from the whole magnitude spectrogram of the input audio file, three soft attention recurrent neural networks (one for each class) to model how these features change in time, and another recurrent network to perform peak-picking on the output activation function. The input spectrogram is calculated using a frame hop size of 11.6 ms.

The CNN, CRNN, and BRNN models were originally proposed in [15]. The CNN and CRNN algorithms share the same four-layer convolutional architecture in the network's backend while differing in their last block of layers. The last block in the CNN consists of two dense layers, while the one in the CRNN consists of three bi-directional recurrent layers. The BRNN is uniquely composed of three bi-directional recurrent layers, without any feature extractor operating in its backend. All three models take the log magnitude spectrogram of the audio file and its first derivative as input representations. These are calculated every 10 ms.

B.5.3 Methodology

We run the five models in their recommended configurations on the fifteen generated datasets and we use the F1-score metric F_1 to evaluate transcription performances. Specifically, we calculate the sum F1-score [26] per dataset. Minimal deviations from the annotated onsets in the predictions are taken into account by setting a tolerance window of 50 ms. In addition to the F1-score for each of the models, we also measure their performance stability. To do so, we first calculate the standard deviation σ_{F_1} of a model's F1-scores across all 15 datasets for each

Model	Precision	Recall	F1-score
CNN	0.934	0.579	0.715
ADTLib	0.662	0.548	0.600
BRNN	0.997	0.530	0.692
NMF	0.823	0.543	0.655
CRNN	0.991	0.626	0.768

Table 2-E: Model performances over instruments and mixing conditions.

instrument and then average the three resulting standard deviations.

B.5.4 Results and Discussion

Figure 1 and **Table 5** display the results from the evaluation process described above.

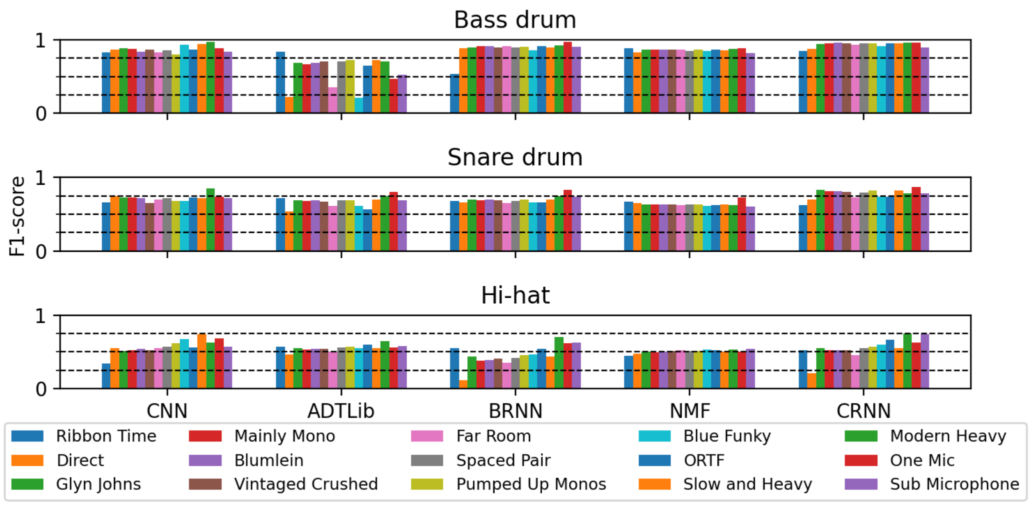


Figure 2.1: Histograms illustrating results of each model for the generated datasets.

In terms of overall performance results, the CRNN model gave the highest F1-score ($F_1 = .768$), followed by the CNN ($F_1 = .715$), the BRNN ($F_1 = .692$), the NMF ($F_1 = .655$), and the ADTLib ($F_1 = .600$). The fact that the NMF's performance was comparable with the rest of the models is particularly interesting, as this algorithm was trained on only 18 samples per class at most. We also note that precision scores are significantly higher than recall scores, meaning that the algorithms tend to miss a relatively high number of drum events while classifying most detected ones correctly. Models based on activation functions, however, may adapt their pick-picking threshold (detection sensitivity) and strike an optimal balance between precision and recall given the context.

Regarding performance stability, the NMF was the most stable ($\sigma_{F_1} = .020$), followed by the CNN ($\sigma_{F_1} = .060$), the CRNN ($\sigma_{F_1} = .063$), the BRNN ($\sigma_{F_1} = .066$) and the ADTLib ($\sigma_{F_1} = .091$). This would make sense, as the NMF is a linear model that was trained with few data samples, making it less prone to overfit to the training dataset than highly complex non-linear models like neural networks.

Several other observations can be drawn when looking at the accuracies on individual datasets. The models performed significantly better on the ”Slow and Heavy” ($F_1 = .759$) and ”Modern Heavy” ($F_1 = .751$) datasets, and significantly worse on the ”Ribbon Time” dataset ($F_1 = .563$), which also accounted for the least stable performances ($\sigma_{F_1} = .149$). This could indicate that post-processing effects have the potential to help with the detection and classification of drum instruments (e.g. equalisation can reduce instrument overlap in the frequency domain) and that the addition of reverb to direct microphones could confuse the algorithms.

We can also see that the presence of room and ambient mics is a very important condition for some models to detect certain instruments. For instance, the BRNN and CRNN models fail to detect the hi-hat properly on the ”Direct Microphones Only” mix. The addition of the far room microphone seems to improve performance in these situations, bringing it a little closer to transcription accuracies with overhead microphones. This shortcoming could also be tackled by the correct choice of hyper-parameters when attempting transcription.

B.6 Conclusions

This paper has covered the release of the BFD-VRT dataset, designed to assess the robustness of ADT and DSS algorithms to recording conditions and post-processing engines. Its generative character lets researchers create their own datasets from symbolic drum patterns while manipulating production parameters like the type of microphones used, volume levels, and sound effects. We illustrated the dataset’s potential by evaluating the performance of five open-source ADT models using it. This allowed us to make some inferences about how accurate the algorithms are expected to be in specific production contexts. The BFD-VRT dataset and the accompanying code for this paper are freely available at [link will be added after review].

References

[1] D. Fitzgerald, "Automatic drum transcription and source separation," *Dublin Institute of Technology, Diss*, 2004.

[2] C.-W. Wu, C. Dittmar, C. Southall, R. Vogl, G. Widmer, J. Hockman, M. Müller, and A. Lerch, "A review of automatic drum transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 9, pp. 1457–1483, 2018.

[3] C. Dittmar, P. López-Serrano, and M. Müller, "Unifying local and global methods for harmonic-percussive source separation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* Ieee, 2018, pp. 176–180.

[4] K. Choi and K. Cho, "Deep unsupervised drum transcription," *arXiv preprint arXiv:1906.03697*, 2019.

[5] C. Jacques and A. Roebel, "Automatic drum transcription with convolutional neural networks," 2018.

[6] C. Lordelo, E. Benetos, S. Dixon, and S. Ahlbäck, "Investigating kernel shapes and skip connections for deep learning-based harmonic-percussive separation," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA).* Ieee, 2019, pp. 40–44.

[7] O. Gillet and G. Richard, "Enst-drums: an extensive audio-visual database for drum signals processing."

[8] C. Dittmar and D. Gärtner, "Real-time transcription and separation of drum recordings based on nmf decomposition." in *DAFx*, 2014, pp. 187–194.

[9] A. R. Tindale, A. Kapur, G. Tzanetakis, and I. Fujinaga, "Retrieval of percussion gestures using timbre classification techniques." in *Ismir*, 2004.

[10] V. M. Souza, G. E. Batista, and N. E. Souza-Filho, "Automatic classification of drum sounds with indefinite pitch," in *2015 International Joint Conference on Neural Networks (IJCNN).* Ieee, 2015, pp. 1–8.

[11] M. Prockup, E. M. Schmidt, J. J. Scott, and Y. E. Kim, "Toward understanding expressive percussion through content based analysis." in *Ismir*. Citeseer, 2013, pp. 143–148.

[12] C.-W. Wu and A. Lerch, "On drum playing technique detection in polyphonic mixtures." in *Ismir*, 2016, pp. 218–224.

[13] C. Dittmar, "Source separation and restoration of drum sounds in music recordings," 2018.

[14] C. Southall, R. Stables, and J. Hockman, "Player vs transcriber: A game approach to data manipulation for automatic drum transcription." in *Ismir*, 2018, pp. 58–65.

[15] R. Vogl, M. Dorfer, G. Widmer, and P. Knees, "Drum transcription via joint beat and drum modeling using convolutional recurrent neural networks." in *Ismir*, 2017, pp. 150–157.

[16] B. Owsinski and D. Moody, *The Drum Recording Handbook*, 2nd ed. Hal Leonard, 2016.

[17] S. McDonald, "BFDLAC: A fast lossless audio compression algorithm for drum sounds," in *Acmc*, 2015.

[18] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, no. 1-2, pp. 151–175, 2010.

[19] S. Lattner, M. Grachten, and G. Widmer, "Learning transposition-invariant interval features from symbolic music and audio," *arXiv preprint arXiv:1806.08236*, 2018.

[20] B. De Man, J. Reiss, and R. Stables, "Ten years of automatic mixing," 2017.

[21] D. Moffat and M. Sandler, "Machine learning multitrack gain mixing of drums," in *Audio Engineering Society Convention 147*. Audio Engineering Society, 2019.

[22] F. Bruford, M. Barthet, S. McDonald, and M. Sandler, "Modelling musical similarity for drum patterns: A perceptual evaluation," in *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound on ZZZ*, 2019, pp. 131–138.

[23] C.-W. Wu and A. Lerch, "Drum transcription using partially fixed non-negative matrix factorization with template adaptation." in *Ismir*, 2015, pp. 257–263.

[24] C. Southall, N. Jillings, R. Stables, and J. Hockman, "Adtweb: An open-source browser based automatic drum transcription system," 2017.

[25] C. Southall, R. Stables, and J. Hockman, "Automatic drum transcription for polyphonic recordings using soft attention mechanisms and convolutional neural networks." in *Ismir*, 2017, pp. 606–612.

[26] C.-W. Wu, C. Dittmar, C. Southall, R. Vogl, G. Widmer, J. Hockman, M. Müller, and A. Lerch, "A Review of Automatic Drum Transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 9, pp. 1457–1483, Sep. 2018, conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing.