Automated Rhythmic Transformation of Drum Recordings

Maciej Tomczak

29 November 2023

A thesis presented for the degree of

Doctor of Philosophy

Sound and Music Analysis Group

Digital Media Technology Lab

Faculty of Computing, Engineering and the Built Environment

Birmingham City University

Abstract

Within the creative industries, music information retrieval techniques are now being applied in a variety of music creation and production applications. Audio artists incorporate techniques from music informatics and machine learning (e.g., beat and metre detection) for generative content creation and manipulation systems within the music production setting. Here musicians, desiring a certain sound or aesthetic influenced by the style of artists they admire, may change or replace the rhythmic pattern and sound characteristics (i.e., timbre) of drums in their recordings with those from an idealised recording (e.g., in processes of redrumming and mashup creation). Automated transformation systems for rhythm and timbre can be powerful tools for music producers, allowing them to quickly and easily adjust the different elements of a drum recording to fit the overall style of a song. The aim of this thesis is to develop systems for automated transformation of rhythmic patterns of drum recordings using a subset of techniques from deep learning called deep generative models (DGM) for neural audio synthesis. DGMs such as autoencoders and generative adversarial networks have been shown to be effective for transforming musical signals in a variety of genres as well as for learning the underlying structure of datasets for generation of new audio examples. To this end, modular deep learning-based systems are presented in this thesis with evaluations which measure the extent of the rhythmic modifications generated by different modes of transformation, which include audio style transfer, drum translation and latent space manipulation. The evaluation results underscore both the strengths and constraints of DGMs for transformation of rhythmic patterns as well as neural synthesis of drum sounds within a variety of musical genres. New audio style transfer (AST) functions were specifically designed for mashup-oriented drum recording transformation. The designed loss objectives lowered the computational demands of the AST algorithm and offered rhythmic transformation capabilities which adhere to a larger rhythmic structure of the input to generate music that is both creative and realistic. To extend the transformation possibilities of DGMs, systems based on adversarial autoencoders (AAE) were proposed for drum translation and continuous rhythmic transformation of bar-length patterns. The evaluations which investigated the lower dimensional representations of the latent space of the proposed system based on AAEs with a Gaussian mixture prior (AAE-GM) highlighted the importance of the structure of the disentangled latent distributions of AAE-GM. Furthermore, the proposed system demonstrated improved performance, as evidenced by higher reconstruction metrics, when compared to traditional autoencoder models. This implies that the system can more accurately recreate complex drum sounds, ensuring that the produced rhythmic transformation maintains richness of the source material. For music producers, this means heightened fidelity in drum synthesis and the potential for more expressive and varied drum tracks, enhancing the creativity in music production. This work also enhances neural drum synthesis by introducing a new, diverse dataset of kick, snare, and hi-hat drum samples, along with multiple drum loop datasets for model training and evaluation. Overall, the work in this thesis increased the profile of the field and hopefully will attract more attention and resources to the area, which will help drive future research and development of neural rhythmic transformation systems.

Acknowledgements

I would like to thank my supervisors Jason Hockman and Ryan Stables for their support, guidance and encouragement with a special thanks to Jason for his expert knowledge and invaluable feedback on my research throughout my PhD journey. A huge thanks to the members of the DMT Lab (Cham Athwal, Carl Southall, Jake Drysdale, Islah Ali-MacLachlan, Alan Dołhasz, Nick Jillings, Sean Enderby, Matthew Cheshire, Sam Smith, Spyros Stasis, Xiangyu Zhu, Xueyang Wang and Ian Williams) for their valuable insights and the camaraderie that made my time at the university so enjoyable. I would also like to thank my co-authors (Carl Southall, Jake Drysdale, Matthew Cheshire, Sean Enderby, Islah Ali-MacLachlan and Masataka Goto) for the productive discussions about drums and music that helped shape my thinking. I would also like to extend my thanks to Masataka Goto for offering me the internship opportunity in the Media Interaction Group (MIG) at AIST Japan and his guidance and encouragement throughout the duration of my internship. An extra thanks to the members and students of the MIG (Masahiro Hamasaki, Megumi Sato, Jun Kato, Tomoyasu Nakano, Kosetsu Tsukuda, Yuki Koyama, Kento Watanabe, Fabrizio Pedersoli, Edward Lin, and Yin-Jyun Luo) who made me feel welcome in Japan and supported me throughout and after my internship.

Lastly, enormous thanks to my family for their constant support and encouragement in all small and large parts of my life. Thank you!

Contents

List of Figures

List of Tables

Acronyms

AE	autoencoder
AAE	adversarial autoencoder
AAE-ISO	AAE with isotropic Gaussian prior
AAE-GM	AAE with Gaussian mixture prior
ART	automated rhythmic transformation
AST	audio style transfer
ADT	automatic drum transcription
BPM	beats per minute
BCE	binary cross-entropy
BIC	Bayesian information criterion
CNN	convolutional neural network
CS	cosine similarity
CQT	constant-Q transform
DAW	digital audio workstation
DGM	deep generative model
DFT	discrete Fourier transform
DNN	deep neural network
DT	drum translation
GAN	generative adversarial network
LSD	log-spectral distance
K-NN	K-nearest neighbors
K-means	K-means clustering algorithm

MIDI	musical instrument digital interface
MIR	music information retrieval
MMD	maximum mean discrepancy
MSE	mean-square error
MIREX	music information retrieval evaluation exchange
NN	neural network
NDS	neural drum synthesis
NAS	neural audio synthesis
PCA	principal component analysis
RNN	recurrent neural network
RMSE	root-mean-square error
RCS	rhythmic cosine similarity
SCS	spectral cosine similarity
STFT	short time Fourier transform
VAE	variational autoencoder
WGAN	Wasserstein GAN
WAE	Wasserstein autoencoder
WAE-MMD	WAE with MMD regularisation
WGAN-GP	Wasserstein GAN with gradient penalty
X-means	X-means clustering algorithm

Mathematical Notation

x	A scalar
$\boldsymbol{x}$	A vector
X	A matrix
$\|x\|$	Absolute of x
M_{i*}	The i-th row of matrix M
M_{*j}	The j-th column of matrix M
M^{-1}	Inverse of matrix M
M^T	Transpose of matrix M
$\max_k, \min_k$	Extrema with respect to an integer value
$\sup_x, \inf_x$	Extrema with respect to a real value
$\max_{i=0}^{I} X_i$	Maximum value of X_i for $0 \leqslant i \leqslant I$
$[0, 1]$	Range with min value of 0 and max value of 1
$x \in X$	Set membership, x is an element of X
$\forall x \in X$	For all elements x in X
$X \in \mathbb{R}^{m,n}$	X is an m by n matrix containing real numbers
$mean(x)$	Mean value of $(\frac{1}{i}\sum_{i=1}^{I} x_i)$
$x \leftarrow x + 1$	x becomes $x + 1$
$x \equiv y$	x is equivalent to y
$x \cdot y$	Inner product of x and y
$x \odot y$	Element-wise multiplication (Hadamard product) of x and y

Chapter 1

Introduction

Automated rhythmic transformations of musical audio are creative computational approaches for the manipulation of musical sounds of varying length (e.g., long, short) and accentuation (e.g., loud, soft) that are grouped into patterns. Taking inspiration from capabilities offered in digital audio workstations and plugins, along with emerging research in the music and multimedia communities, automated rhythmic transformations have become entrenched within modern music production workflows. Recent advances in powerful deep learning (DL) algorithms have given rise to new modalities of neural audio synthesis and effects processing procedures, which in turn have afforded new musical supportive systems for pitch, timbre and rhythm manipulation for music arrangement and sound design. Although various advances in deep learning have been proposed, the majority of these have focused on generation, interaction and visualisation of pitched instruments, and relatively few have explored generation of percussion instruments and transformations of the underlying rhythmic patterns.

In this thesis the term rhythm is used in two ways: 1) in reference to the process of analysis of musical time via computational methods for *rhythmic description*, and 2) in context of a bar-length pattern of events referred to as a *rhythmic pattern*. In the title of this thesis *Automated Rhythmic Transformation of Drum Recordings* both meanings are implied, whereas in other instances the usage is distinct. A distinction is also made between the use of words *rhythm* and *rhythmic* in a similar manner as a distinction between the identification of musical constructs that exist in our minds and extraction of what would be phenomenally present in the audio signal (Gouyon, 2003). For instance, a recurring rhythmic pattern of sound events can be phenomenally present in the acoustic signal; however, not all possible patterns are perceived as different rhythms, thus ascertaining that acoustic realisation of rhythm and its perception are two separate phenomena. In addition, for automated transformations to be possible, content processing of music audio signals is commonly employed using signal processing and deep learning methods. In this context the word content represents any piece of information related to the audio source that is in any way meaningful (i.e., carries semantic information) to the user (Amatriain et al., 2003). The term *transformation* should not be understood in the same way as an effect. For a transformation the emphasis should be put on the change that a particular sound experiences, rather

than on the result. Thus, not every sound can undergo a certain transformation but an effect can be applied on any source.

The history of rhythmic transformation of drum recordings is closely connected with the development of sampling, the practice of using a recorded sound or sequence of sounds in a new composition. The French composer Pierre Schaeffer began experimenting with sound manipulation techniques in the 1940s using turntables and tape recorders. Schaeffer's use and manipulation of percussion sounds through tape editing and other techniques is considered to be an important precursor to the use of sampling in electronic music. An important composition which incorporated sampled percussion sounds can be heard in his piece named Étude aux Tourniquets (Toy Tops and Percussion Instruments) (Schrader, 1982). Central to the sonic creation of electronic music is the music producer's selection process, or *sourcing*, which commonly involves percussion recordings. Since the introduction of the 'affordable' digital sampler in the late 1980s the most frequently sampled recordings became the percussion solos from 1960s and 1980s Jazz and Funk and R&B performances—referred to as breakbeats—which soon became prevalent in popular music such as Hip Hop (Chang, 2007) and electronic music genres (Hockman, 2014; Reynolds, 2012). Following sourcing of percussion recordings, the subsequent transformation process often involves segmentation of drum events and rearranging them along with further modification through additional effects and layering. To date, computational analysis of percussion has focused on tasks such as, automatic drum transcription (Wu et al., 2018), timbre description (Aucouturier et al., 2005) and rhythm and metre analysis (Klapuri and Davy, 2007; Böck, 2016).

Modern deep generative models for neural audio synthesis are well-suited for automated rhythmic transformation of drum recordings due to their ability to learn complex patterns and relationships in music signals (Carney et al., 2021; Dhariwal et al., 2020). Recent advances in deep learning can facilitate the generation of new drum sounds and rhythms by exploiting learned representations of existing recordings. Deep learning-based models can learn a compact representations of the training data, and then use this representation to generate new audio examples. This approach can be used to manipulate percussion recordings in various ways, such as changing the tempo, pitch, timbre of the drums as well as creating entirely new breakbeats by interpolating between different rhythmic styles.

The research presented in this thesis explores the capabilities of the deep learning models for leveraging the rhythmic patterns of percussion recordings learned in the analysis phase to provide interpolation across the latent space with the objective of transformation of original recording with a given pattern to another pattern. The implications of different possibilities are presented throughout this thesis, with research outcomes consisting of the thesis itself, open source rhythmic transformation implementations and datasets to promote future research in neural drum synthesis for automated rhythmic transformation.

1.1 Motivation

Within the creative industries, music information retrieval (MIR) techniques are now being applied in a variety of music creation and production applications. In addition, musicians and DJs are able to incorporate techniques from music informatics and deep learning towards the development of systems for generative content creation and manipulation. Examples of research in this field include content-aware audio effects (Amatriain et al., 2003; Wilmering et al., 2013), musical expert agents (Knees et al., 2015) and automatic remixing (Davies et al., 2014a). In each case the decisions are made for the user based on algorithmic interpretation of the data to perform audio generation and modification (e.g., source separation, timbre morphing, rhythmic transformation). The automated transformations of musical audio therefore allow users to focus more on the creative rather than technical aspects of the creation process and encourage greater experimentation. In contrast, the manual manipulation of audio can be time consuming and require a high-level knowledge of digital audio workstations. Both automated and manual transformations operate on the audio signal through the manipulation of perceptual attributes such as loudness, pitch, timbre, position (i.e., spatial hearing) and duration (i.e., time). While deep learning systems exist for neural generative synthesis of audio (Dhariwal et al., 2020; Engel et al., 2020), meaningful control over the intermediate latent space (e.g., between rhythms of source and target recordings) remains an open research issue for rhythmic and multi-timbral percussion recordings, limiting the synthesis and transformative possibilities of such systems.

Artificially intelligent technologies have had a significant impact on the music industry. Streaming services like Spotify and Pandora use deep learning to recommend music to listeners, and deep learning-based music production tools have become increasingly popular in recent years allowing musicians to create, manipulate and generate music in new ways (Bittner et al., 2017; Tingle et al., 2010). There is a wide range of AI-based music production tools available to music producers, each targeting different stages of the music creation process with different levels of complexity and scope. Some examples includes sequencing tools like Magenta MelodyRNN and Melody Transformer,[1] which can generate MIDI sequences for use in music production software; effects software like Magenta DDSP Tone Transfer,[2] which can manipulate the sound of an existing recording to create new variations; neural audio synthesis tools like Bytedance Mawf,[3] which can generate new sounds and timbres; music composition tools like AIVA,[4] which can generate entire songs without any human input; as well as AI-based mixing and mastering tools like iZotope Ozone[5] and MixGenius Landr,[6] which can automatically balance and enhance audio recordings. These tools are not only used by individual music producers but also by professionals in the music industry, and are increasingly becoming a standard in music production.

[1] https://magenta.tensorflow.org/music-transformer
[2] https://sites.research.google/tonetransfer
[3] https://mawf.io/
[4] https://www.aiva.ai/
[5] https://www.izotope.com/en/products/ozone.html
[6] https://www.landr.com/

While commercial deep learning-based tools specifically for breakbeats have not yet been developed, there are deep learning-based tools available for working with individual sample percussion recordings. These tools include sample organisation and sequencing tools like XLN XO[7] and Algonaut Atlas,[8] which provide a more efficient and intuitive way to navigate and organise sound samples. Additionally, neural drum synthesis tools that use generative adversarial networks (GANs), such as Sony CSL DrumGAN,[9] which can generate new drum sounds by interpolating between learned representations of existing sounds. These tools can help musicians to expand the sonic possibilities of their recording collections.

1.2 Aim and Objectives

The aim of this thesis is to develop systems for automated transformation of rhythmic patterns of percussion instruments using deep generative models. The dissertation has three main objectives:

- To formulate and describe modes of transformation of percussion instruments through the use of deep generative models. While the topic of neural audio synthesis has been more widely discussed, an account of neural drum synthesis approaches for the transformation of rhythmic patterns of percussion instruments does not yet exist.
- To explore and propose systematic methodologies for rhythmic transformation using deep generative models, ensuring control over the rhythmic patterns and timbral characteristics of drum recordings.
- To develop and optimise modular deep learning-based systems for drum synthesis and rhythmic transformation with detailed description of architectures and model parameters for reproducibility.

These are achieved through the following contributions.

1.3 Contributions

- Formulation, description and naming of modes of transformation of drum instruments using deep generative models (Chapter 3)
- Development and optimisation of modular deep learning based systems for rhythmic transformation (Chapters 4–6)
- An audio style transfer model that incorporates rhythmic constraints for rhythmic transformation of drums from multiple recordings in a form of a mashup (Chapter 4)
- Creation of new loss functions suited for mashup-oriented drum transformations (Chapter 4)
- Development of drum translation system for timbral and rhythmic transformation drum recordings (Chapter 5)

[7] https://www.xlnaudio.com/products/xo
[8] https://algonaut.audio/
[9] https://cslmusicteam.sony.fr/prototypes/drumgan/

- Development of a rhythmic transformation system for continuous transformation of drum recordings, steered by a latent space representation conditioned on rhythmic pattern styles (Chapter 6)
- Implementation of comprehensive evaluation methodologies, encompassing similarity metrics, onset detection and automatic drum transcription to rigorously assess and validate the capabilities and efficacy of the proposed systems (Chapters 4–6).
- Development of new datasets for the study of rhythmic patterns (Chapters 4, 5)

Open Source Implementations and Datasets

- **Audio style transfer with rhythmic constraints**: An open source command line transformation system.
  `https://github.com/maciek-tomczak/audio-style-transfer-with-rhythmic-constraints`
- **Dataset of drum samples:** Dataset of high-quality individual kick, snare and hi-hat samples used in training of the proposed rhythmic transformation system in Chapter 5.
  Web implementation of dataset visualisation available at `https://tdsdne.vercel.app/`
- **Dataset of drum loops (Sonified MIDI):** Dataset of sonified MIDI drum loops curated in preliminary work for Chapter 4.
  `https://maciek-tomczak.github.io/rppw2017/`
- **Dataset of drum loops (kicks and snares):** Dataset containing kick and snare drums compiled and used in evaluations of the proposed systems in Chapters 4 and 5.
  `https://maciek-tomczak.github.io/dafx2018/`
- **Dataset of drum loops (kicks, snares and hi-hats):** Dataset containing kicks, snares and hi-hat instruments compiled and used in evaluations of the proposed system in Chapter 5.
  `https://maciek-tomczak.github.io/dafx2019/`
- **Dataset of drum loops (`HJDB`, `DALI`, `HMX`, and `JFRB`):** Dataset of source-separated drum loops curated using existing and publicly available datasets for the training and evaluation of the proposed system in Chapter 6.
  `https://maciek-tomczak.github.io/acm2020/`

1.4 Structure

The remainder of this thesis is structured into six chapters as follows: Chapters 2 and 3 provide background information and literature reviews, and Chapters 4–6 provide the main research projects. Chapter 7 reiterates the main findings from Chapters 1–6 and presents suggestions for future work in this area. The following provides a more detailed explanation of each of the seven chapters.

In **Chapter 2**, the task of automated rhythmic transformation is described in context of musical rhythm, instrumentation and cultural origins surrounding different transformation types applied to drum recordings. The relevant literature to the task is reviewed to understand the current state of the field. This is essential in understanding and placing the contributions of this thesis. First, the background of research into rhythmic description is presented. It begins with a brief overview of the modern Western drum kit and ends with discussion of different subtasks and algorithms for the automatic description of the metrical structure of music signals. Finally, the review of the existing methods for automated rhythmic transformation of drum recordings is presented, with a particular emphasis on advanced signal-processing and deep learning techniques. This exploration of the literature lays the groundwork for the thesis and paves the way for the investigations in the following chapters.

Chapter 3 presents and overview of the deep learning techniques utilised in this thesis. The chapter starts with a discussion of the approaches for the training process of deep neural networks, following with a presentation of different modes of transformation specialised on the neural synthesis and rhythmic-timbral modification of drum recordings using a range of deep learning techniques. The modes of transformation are presented in the context of deep generative models for neural audio synthesis. These preliminaries are crucial for comprehending the performance and capabilities of the systems deployed in this thesis, offering the essential context for the subsequent chapters.

Chapter 4 introduces a system for rhythmically constrained audio style transfer, akin to automatic remixing and mashup creation of drum recordings. The system facilitates automatic remixing and mashup creation. In this mode of transformation, the rhythmic and timbral features of two or more input signals are modified such that the result adheres to the larger metrical structure of the chosen input. New loss functions for mashup and augmented mashup transformations are proposed and evaluated using various similarity and onset detection performance metrics. The final section of the chapter highlights the system's effectiveness in introducing precise, user-defined rhythmic modifications—as validated by onset detection results—and provides audio examples that demonstrate its distinctive capability to modify rhythmic patterns, surpassing existing techniques while preserving rhythmic coherence.

Chapter 5 presents the development of a drum translation system, aiming at both timbral and rhythmic transformation of drum recordings offered by generative audio synthesis with WaveNet autoencoders. The system serves as a method for redrumming which facilitates a new musical composition technique, enabling artists to blend and morph drum sounds within their works. The chapter presents the methodology behind the translation approach, which allows for the input of variable-length drum recordings to be adapted, mimicking the performance of different drum instruments. The chapter details the architecture and parameters that allow a user to exert control over generated rhythmic patterns. A novel visual web-based method is then introduced for the removal of outliers within a newly curated dataset of drum samples. The translation results are assessed, noting the improvements and the capacity

for the system to translate percussion instruments in simple and complex scenarios. Lastly, the chapter sets the stage for Chapter 6, which merges the drum synthesis and transformation systems into a continuous framework.

Chapter 6 presents a new method that merges drum synthesis with rhythmic pattern manipulation, using Gaussian mixture adversarial autoencoders (`AAE-GM`) conditioned on real-world percussion recordings. This chapter provides insights into a continuous transformation system, which synthesises individual drum sounds and enables manipulation of rhythmic patterns within bar-length segments. The method offers a framework for interactive music applications where drum tracks adapt to user input and musical context, as well as for generative music systems that enable neural drum synthesis and the alteration of rhythmic patterns. The chapter begins by introducing the system architecture and elaborating on how rhythmic patterns are represented and used to condition the `AAE-GM` during its training phase. The latter part of the chapter is dedicated to evaluating the new system against established baseline systems, with a thorough analysis of its performance in creating and transforming drum patterns. This evaluation is supported by a dataset comprising over 500,000 bars from a diverse selection of audio tracks annotated for this purpose. The chapter proceeds to explore the multi-dimensional latent space that facilitates the continuous manipulation of bar-length patterns.

The thesis is concluded in **Chapter 7** with a summary of findings across Chapters 4 to 6, and suggestions for future work in this area.

1.5 Publications

The following is a list of all published papers that are directly associated with this thesis:

- **Tomczak M.**, M. Goto, and J. Hockman, Drum Synthesis and Rhythmic Transformation with Adversarial Autoencoders. ACM International Conference on Multimedia (ACM-MM), Seattle, WA, USA, 2020.
- **Tomczak M.**, J. Drysdale and J. Hockman, Drum translation for timbral and rhythmic transformation. In Proceedings of the International Conference on Digital Audio Effects (DAFx), Birmingham, United Kingdom, 2019.
- **Tomczak M.**, C. Southall and J. Hockman, Audio style transfer with rhythmic constraints. In Proceedings of the International Conference on Digital Audio Effects (DAFx), Aveiro, Portugal, 2018.
- **Tomczak M.**, C. Southall and J. Hockman, Rhythm modelling using convolutional neural networks. In Rhythm Production and Perception Workshop (RRPW), Birmingham, United Kingdom, 2017.

The following papers are associated with neural drum synthesis and audio effects which use deep generative systems for drum processing that do not directly contribute to this thesis. While these

collaborative works primarily delve into specific techniques of synthesis of drum samples, this thesis articulates a framework centred on the automated transformation of rhythmic patterns of drum recordings. This positions the research distinctly within the broader domain of manipulating rhythmic patterns of drum recordings using deep generative models for neural drum synthesis.

- Cheshire M., J. Drysdale, S. Enderby, **M. Tomczak** and J. Hockman, Deep Audio FX for Snare Drum Recording Transformations. In Journal of the Audio Engineering Society (JAES), Special Issue: New Trends in Audio Effects, 2022.
- Drysdale, J., **M. Tomczak** and J. Hockman. Style-based Drum Synthesis with GAN Inversion. In Proceedings of the International Society of Music Information Retrieval Conference (ISMIR), 2021.
- Drysdale J., **M. Tomczak** and J. Hockman, Adversarial Synthesis of Drum Sounds. In Proceedings of the International Conference on Digital Audio Effects (DAFx), Vienna, Austria, 2020-21.

Chapter 2

Review of Rhythmic Description and Transformation Literature

Rhythmic transformation of drum recordings is inherently connected with the history of sound recording and sampling technology progressing from the end of 19th century to the digital era of the present day. In the late 1940s, French composer Pierre Schaeffer, by means of discs, tape recorders, and several kinds of analogue devices, pioneered the manipulation and sampling of recorded sounds. He was motivated by the possibilities that this technique introduced for creation of new artistic works. An example of Schaeffer's compositional approach to rhythmic transformations of drum recordings can be heard in the piece named Étude aux Tourniquets (Toy Tops and Percussion Instruments) part of his Cinq Études de Bruits (Five Studies of Noises) (Schrader, 1982). Several such techniques applied to pre-recorded sounds using analogue tape were described by Serra (1989) in the context of transformations in electronic music. More recently, sampling became one leading practice for many DJ-orientated electronic music genres that evolved with the development of affordable home-based studio equipment. Electronic music genres are characterised by a high degree of manipulation of drum recordings using techniques such as drum resequencing, time-stretching and pitch-shifting (Collins, 2001; Hockman, 2007; Hockman, 2014).

Creative manipulations of drum recordings developed along with the early use of hardware samplers and evolved into sophisticated digital audio effects and support applications with rhythmic transcription functionalities (e.g., automatic beat and rhythmic pattern detection). Early digital audio effects relied on digital signal processing (DSP) algorithms (Zölzer, 2011), but in the recent years deep learning (DL) techniques have emerged as prominent creative tools that have introduced musicians to new modes of music creation (Knees et al., 2016, 2015). These techniques are capable of sound transformation as well as generation, and within this dissertation, the title refers to a special use case of deep learning methods applied to musical signals that target rhythmic transformation of drum recordings. In this chapter, an overview of components related to rhythmic description and transformation is presented to provide an understanding of the history and current trends within the field of music informatics. Section 2.1

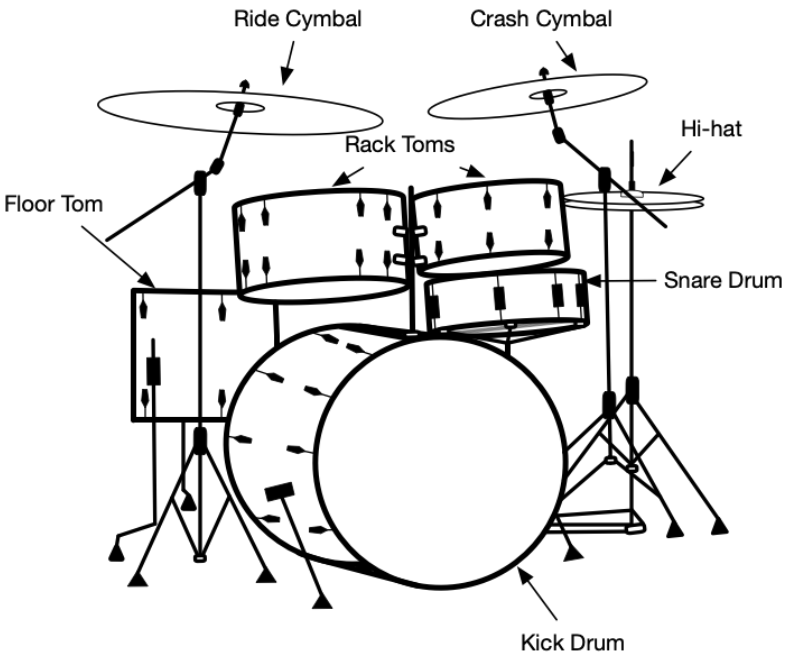

Figure 2.1: The modern Western drum kit equipped with a kick drum, snare drum, hi-hats, rack toms, floor tom, crash cymbal and ride cymbal.

presents an overview of characteristics of drums and discusses features of styles related to percussion instruments. Section 2.2 considers literature on rhythm and covers computational methods for rhythmic description. Previous approaches to automated rhythmic transformation are discussed in Section 2.3.

2.1 Characteristics of Drums

Within Western popular and traditional music styles such as jazz, rock, pop and dance music, the drum kit is an important component of the rhythm section that sets the basic tempo as well as the rhythmic structure for a musical performance. This structure is realised by the means of rhythmic patterns, often referred to as drum patterns, which contribute to the desired sound for each music genre. The kick and snare drums—although not exclusively—provide the rhythm in popular music through interlocking patterns and can be clear indicators to beat locations. These patterns are of chief interest to music producers (Hewitt, 2009; Snoman, 2012), but are also the subject of research in several fields related to rhythm and timbre such as rhythmic similarity (Paulus and Klapuri, 2002), rhythmic style modelling (Marín, 2018) and automatic drum transcription (Southall, 2019). In order to achieve automated systems for rhythmic transformation proposed in this thesis an understanding of drum instruments together with different styles of playing and production techniques is required.

2.1.1 Instruments of the Drum Kit

The drum kit consists of multiple instruments played by a single musician through the use of their hands and feet as well as external tools such as drum sticks, brushes and pedals. Figure 2.1 shows the modern Western drum kit. Percussion instruments can be classed as membraphones (i.e., drums) and idiophones

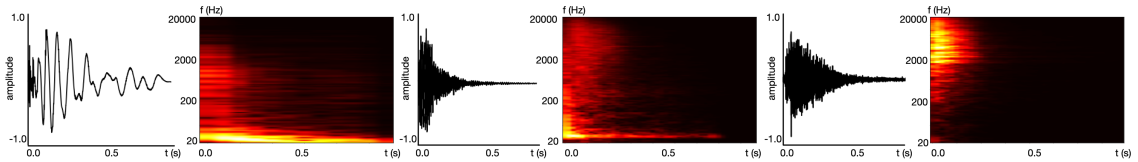

Figure 2.2: Waveforms and spectrograms for a kick (left), snare (middle), and a closed hi-hat (right).

(e.g., bells, cymbals). The standard drum kit includes membraphones such as the kick drum, snare drum, and toms, as well as idiophones such as hi-hats and cymbals. Popular additions to the drum kit are instruments like tambourines, cow bells, and wood- or plastic-blocks. An overview of the primary instruments from the drum kit is provided below.

Kick Drum: The kick drum (or bass drum) is the largest instrument in the typical drum kit and is played using a foot pedal with an attached beater. It produces a low pitch sound and is often used to emphasise the beat locations. Typical frequency ranges of a kick drum are 50–145Hz (fundamentals) and 1kHz–6kHz (harmonics) (Major, 2014; Rossing et al., 2014). The left part of Figure 2.2 shows the audio waveform and a spectrogram (i.e., time-frequency representation) of a kick drum.

Snare Drum: The snare drum is a shallow drum with a set of wires (i.e., snare wires) held under its lower side, which vibrate in sympathy when the drum is struck. It is a versatile instrument capable of producing a range of timbres with different playing techniques. The snare can be struck with variable velocities in different locations across the batter head producing different timbres due to different modes of the membrane being excited (Rossing, 2001; Rossing et al., 2014). For example, snare drums can be hit lightly producing quiet sound, termed a ghost note, or can be used to accentuate the sound with a standard drum stroke that applies a greater force (e.g., ghost notes (♩) and snare accents > in Figure 2.4). In both cases the snare wires augment the produced sound in chaotic ways introducing inharmonic wire noise to the vibrational modes of the drum (Fletcher and Rossing, 1998). Typical frequency ranges of a snare drum are 100–200Hz (fundamentals) and 1–20kHz (harmonics) (Owsinski, 2017; Owsinski and Moody, 2009). The middle part of Figure 2.2 presents an audio waveform and a spectrogram of a snare drum.

Hi-hat: The hi-hat is built with two cymbals mounted on a stand, where the top cymbal is facing downwards and the bottom is facing upwards. The hi-hat can be hit with a drum stick or can be controlled with a pedal. Different combinations of open, closed or in-between (i.e., washy) hi-hats facilitate various playing techniques and produce different sounds when hit with a stick or when operated separately through the foot pedal. Typical frequency ranges of a hi-hat are 300–580Hz (fundamentals) and 1–15kHz (harmonics) (Major, 2014). The right part of Figure 2.2 presents an audio waveform and spectrogram of a closed hi-hat.

Other drums and cymbals: Other drums of the drum kit include rack and floor toms that are commonly used in drum solos and sections that diverge from the main rhythmic pattern (i.e., drum fills)

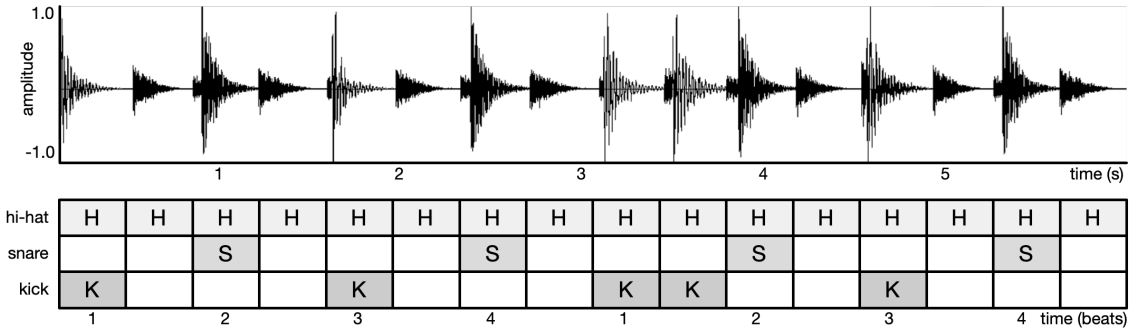

Figure 2.3: Time-unit box representation of a standard eighth note drum pattern. Audio signal of a drum recording (top) and an eighth note quantised representation of patterns made by kick, snare and hi-hat (bottom).

characteristic of rhythm & blues (R&B), rock and funk music (Stewart, 2000). The other cymbals used within the drum kit include: crash, ride, splash and china cymbals. They are typically used to emphasise particular rhythmic sections, or in the case of the ride cymbal, build momentum in the music piece and give the drum pattern a sense of continuity and motion (Hewitt, 2009).

2.1.2 Features of Drum Styles

Described below are several of the main rhythmic devices (i.e., swing and syncopation) that characterise different qualities of rhythmic patterns in various styles of music as outlined by Pressing (2002), while more detailed definitions of rhythm and metre are presented in Section 2.2. Music styles often rely on interlocked and repeating rhythmic patterns created by the kick drum, snare drum and cymbals—usually the hi-hat or ride cymbal. A standard eighth note rhythmic pattern made of three instruments is illustrated in Figure 2.3. The hi-hat plays a constant eighth note pattern and both kick and snare drums are played in alternating patterns. There are two aspects to rhythmic patterns that characterise their role in different styles of music (Desain and Honing, 1989). The first refers to how pattern events can be aligned to a quantised time grid. Notes represented on a discrete grid can be described by integer multiples or divisions of a quarter note and can reflect how a pattern would be notated on a score as in Figure 2.4. The second involves systematic timing deviations between events and their corresponding positions in the metrical structure. These deviations can be represented when performers use tempo variation as means of expression or as event shifts at a constant tempo (Bilmes, 1993; Desain and Honing, 1991).

Jazz, funk and Latin music is characterised by a pattern of deviations called swing in which consecutive eighth notes are performed as long-short patterns (Friberg and Sundström, 2002). In these patterns the duration of the first eighth note is extended from the original eighth note ratio of 1:1. Conventionally, swing in jazz corresponds to notation of a beat that is divided in the ratio of 2:1, where the first eighth note can be represented by two triplet eighth notes. Friberg and Sundström (1999) have shown that this ratio decreases at fast tempi, while in practice this ratio varies and depends on the drummer as shown in analysis on jazz and Cuban recordings by Alén (1995) and Freeman and Lacey (2002). Syncopation refers to the stressing of normally unstressed beats in a rhythmic pattern via expressively timed events;

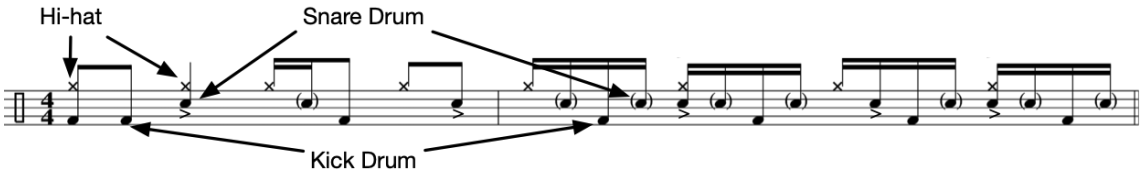

Figure 2.4: Drum notation of a rhythmic pattern represented on a music staff. The hi-hat pattern is a sequence of regular quarter notes interlocked with kick and snare drum backbeat patterns. The snare drum backbeat is extended with the fourth offbeats (>) instead of the fourth onbeats and a sixteenth note ghost note ((♩)) pattern. The kick drums are predominantly placed on the third offbeats. Audio arrangement of this drum notation is available here `https://funklet.com/mother-popcorn/`.

however, it can also refer to the interruption of a repeating pattern (e.g., switching a snare hit with a hi-hat, thus disturbing the expected pattern). Butler (2006) has described two types of syncopation catered to instrumental drum performances. In his first, general type the accentuations occur on beats 2 and 4 (commonly referred to as backbeat in Western music (Frane, 2017)). The second type, extends a model by Temperley (1999) to include forward as well as backward displacement of syncopated events that define drum patterns specific to funk music. In addition, delayed events can eventually stretch a rhythmic pattern and create surprise in form of syncopation as discussed in Robertson (2009) with an example recording from John Bonham of Led Zeppelin. The notion of displacement of events in relation to the beat is referred to by drummers as playing behind the beat, in front of the beat or in the pocket (Robertson, 2009). Gouyon et al. (2003) observed that rock and funk drummers play quarter notes slightly behind the beat. Such deviations occur in many musical styles and rely on the positions in the metrical structure (Gouyon, 2007). An example can be seen in the top part of Figure 2.3, where hi-hat events appear slightly before snare drums (e.g., around the 1sec and 4sec). Moreover, various authors have investigated the theoretical and practical knowledge required for composition of rhythmic patterns in educational scenarios (Adamo, 2010; Hewitt, 2009; Snoman, 2012). To explain characteristic features of rhythmic patterns, notations are often used (as in Figures 2.3 and 2.4).

2.2 Rhythmic Description

The previous section has demonstrated the variety of sounds that can be produced by the drum kit as well as the qualities of rhythmic patterns utilised in different styles of music. Automatic rhythmic description methods introduced in this section refer to detection of events in the audio signal (e.g., notes and drum events), synchronisation points (e.g., beats and downbeats) and rhythmic patterns. These methods are chief for rhythmically-informed music transformations (Gouyon et al., 2003; Hockman et al., 2008; Ravelli et al., 2007). This section begins with definitions of terminology related to rhythm and metre (Section 2.2.1) to provide a basis for review of computational methods for analysis and description of rhythmic information from audio signals utilised in automated transformation of drum recordings. Section 2.2.2 covers onset detection used in a process of locating events in an audio signal. Sections 2.2.3 and 2.2.4 review methods for beat and downbeat detection that aim to extract

perceptual synchronisation points in the rhythmic structure of music. Section 2.2.5 presents an overview of automatic methods for the detection of drums in audio recordings.

2.2.1 Rhythm and Metre

One commonly responds to music through repeated movements such as tapping to the beat or swaying with the pulse of music. Underlying such motions is a system of perception of time that allows humans to tap to the perceived beat—an inferred pulse of a succession of events—while cueing subdivisions of this beat with their hands (Fraisse, 1984). Metre is a counting mechanism defined by the presence of a regular pattern of accented and unaccented beats. Rhythm defines the grouping of events into patterns based on the length, pitch and accentuation of these events in sequence. For example, one may emphasise syncopation of certain events by accentuating them with other body movements that when synchronised together can represent an overall rhythmic structure of music in a process referred to as entrainment (Witek et al., 2014). The recognition of events and an inference of beats underlying the music—that do not need to correspond to any particular sound events (Rosenthal, 1992)—facilitates the perception of rhythm and metre. The concept of rhythmic, or metrical, structure in rhythm relates to grouping of beats together. Grouping describes how a series of sounds are perceived to be clustered or grouped together. Research on rhythm grouping principles shares similarities with work on development of Gestalt principles of perceptual organisation (Tenney and Polansky, 1980). The formal study of grouping and metrical structures by Lerdahl and Jackendoff (1983) introduced the concept of alternating sequence of strong and weak beats in a hierarchy. The set of strong beats forms the metrical level above the beat level. This grouping of beats creates bars (also known as measures) with the first beat in a bar referred to as a downbeat. Lerdahl and Jackendoff (1983) used the term tactus to define the most influential beat level that divides the bar and corresponds to the main quarter note beat described by a time signature. In music notation, time signature serves a similar purpose to metre defining two numbers, one stacked above the other (e.g., represented in common binary $\frac{2}{4}$, ternary $\frac{3}{4}$, and quaternary $\frac{4}{4}$ subdivisions). The lower provides the metrical duration of a beat (e.g., quarter note, eighth note), and the upper provides the number of such beat durations in a bar. The metrical levels below the beat are also characterised by strong and weak events, where the strong events correspond with the beats. Bilmes (1993) termed the lowest metrical level in a musical performance as *tatum* in honour of the jazz pianist Art Tatum. This level forms a metrical grid on which most events will occur. The metrical levels can also develop above the bar level to form hyper-measures that can correspond to structural segments in music such as the verse and chorus (Allan, 2004).

Research related to our ability to identify rhythmic structures in music has been conducted in fields of music theory, psychology and cognitive science. The work in these fields ranges from study of structural hierarchies in symbolic music notation (e.g., Cooper and Meyer, 1963; Lerdahl and Jackendoff, 1983) to psychological models of listener attention and timekeeping (e.g., Jones and Boltz, 1989; London, 2012; Temperley, 2004). The intuitive ability of humans to synchronise and remain in synchrony was studied in sensorimotor research using metronomic and musical stimuli (e.g., Fraisse, 1982; Repp, 2005). This work

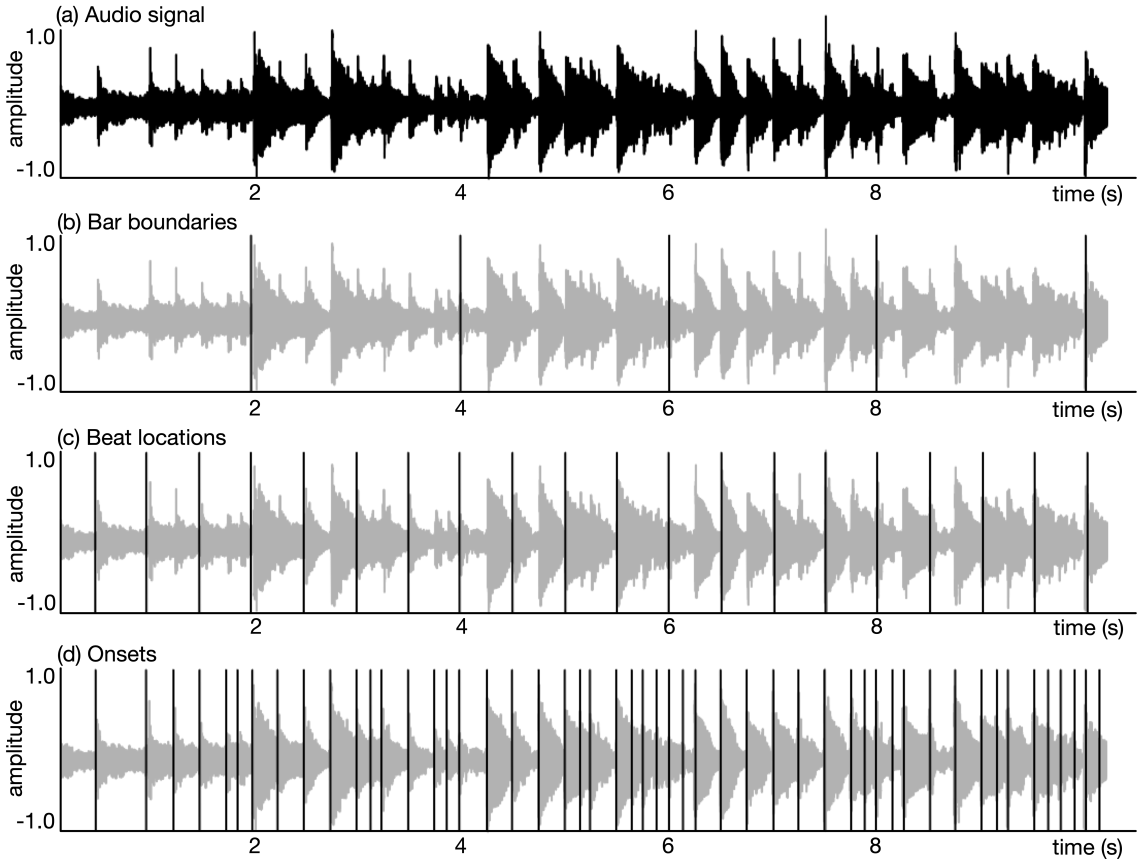

Figure 2.5: Metrical structure locations (black vertical lines) for three levels depicting: (a) audio signal, (b) bar boundaries, (c) beat locations, (d) onsets representing time markers at positions of musical events.

contributed to the understanding of limits of human rhythmic perception, memory and reproduction, suggesting optimal temporal ranges and boundaries for our percept of rhythm, beat and metre.

Computational approaches to rhythm and metre analysis of audio signals are a necessary component of automated rhythmic transformation systems and rely on transcription of a symbolic list of music events. Within the field of music information retrieval (MIR), automatic rhythmic description systems seek to extract rhythmic structure information from a musical recording through localisation of bar boundaries and characterisation of acoustic events within these boundaries (Gouyon, 2003; Termens, 2004). Rhythmic structure can be automatically detected in audio signals through identification of note start times (onsets), the inferred pulse of the music (beats), and the bar boundaries (downbeats) in subtasks of onset detection (Bello et al., 2005), beat tracking (Goto and Muraoka, 1994) and downbeat detection (Goto, 2001), respectively. These boundaries are illustrated in Figure 2.5 and the corresponding computational approaches for their detection are described in the following sections. Computational approaches for rhythmic description can be also used in the detection of rhythmic patterns (Krebs et al., 2014) and classification of music signals based on their rhythmic similarities (Paulus and Klapuri, 2002) as well as in the task of automatic drum transcription (Wu et al., 2018).

2.2.2 Onset Detection

Within the field of MIR, high-level rhythm analysis tasks such as the detection of beats and downbeats rely on the detection of periodicities in the occurrences of music events. The location of beginnings of events in an audio signal is assisted by the task of onset detection. Onsets are markers in time that represent the beginning of events in music signals. In music, onsets are often characterised by transients, which are associated with very brief periods of time, for instance, the moment a stick comes in contact with the drum membrane (Bello et al., 2005). Transients are short-lasting sounds of high amplitude that occur at the beginning of a waveform, where the signal changes abruptly (Crocker, 1998). Transients are often associated with attack regions—also referred to as attack transients—of drum recordings and are characterised by energy in a wide-band frequency range. Figure 2.6 illustrates a waveform of a drum recording together with an onset location marking the beginning of an idealised envelope with the demarcated attack and decay regions. While the attack transients have been used in percussive onset detection approaches by Duxbury et al. (2002) and Masri (1996), alternative methods may be necessary for the detection of more difficult sounds (e.g., snare ghost notes). This section presents a review of onset detection methods often utilised as a first processing step for many music tasks (e.g., beat and downbeat detection) and music applications related to automated rhythmic transformation.

The existing methods for onset detection rely on three computational stages: (1) creation of an audio input representation, (2) creation of an onset detection function, and (3) onset selection. Several audio representations have been utilised to facilitate the detection of desired signal attributes (e.g., transients). Bello et al. (2005) categorised preprocessing techniques applied to input audio representations into transient identification and modification techniques or sub-band decomposition. The former focused on extraction or enhancement of transient signal regions with methods such as spectral modelling synthesis in (McAulay and Quatieri, 1986; Verma et al., 1997), and the latter included methods that used different filtering (i.e., merging several frequency bins into a single one) techniques. The use of filtering methods was motivated by the aim to reduce the interference between instruments in different ranges of the spectrum and can be subdivided by the filterbank type used, such as Mel scale bands, Bark scale bands, and constant-Q bands. These are perceptually motivated scales that aim to overcome the linear resolution of the discrete Fourier transform, where all frequency bins are evenly spaced in each spectral frame. Here, the bins of a spectral frame refer to the frequency axis of a time-frequency representations (i.e., spectrogram) of an audio signal such as the short-time Fourier transform (STFT) and the constant-Q transform. The constant-Q transform ensures an identical number of frequency bins per musical octave and was used in approaches by Böck et al. (2012b) and Lacoste and Eck (2006). Mel and Bark scales as well as logarithmic filtered magnitude spectrograms were assessed by Böck et al. (2012a), Eyben et al. (2010), and Schlüter and Böck (2014) for onset detection in a wide range of musical instruments.

An onset detection function (ODF)—also termed novelty function (Foote, 2000) or rhythmic envelope (Roma, 2008)—of an audio signal emphasises attack transients of events as illustrated in Figure 2.7.

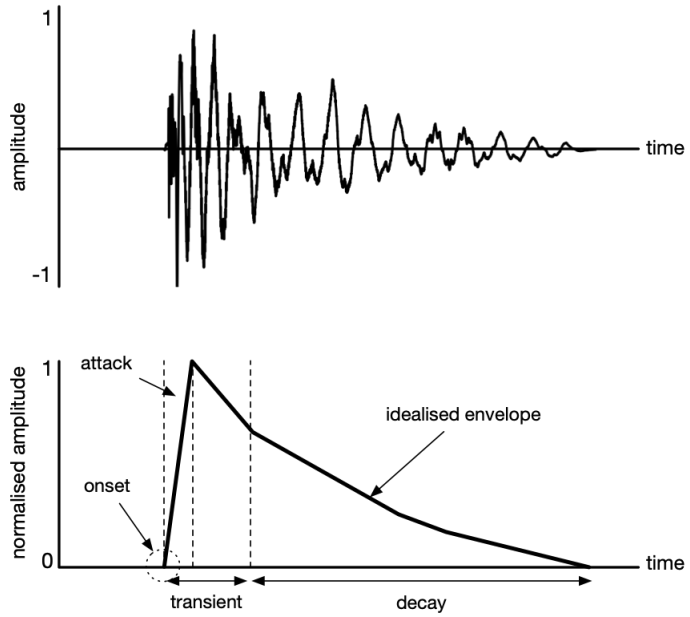

Figure 2.6: Audio waveform (top) and idealised envelope (bottom) consisting of attack and decay sections separated by dashed line. The onset marks the beginning of the attack section.

The created ODF is represented with a feature at a lower sampling rate than the original audio signal. ODF computation approaches can be categorised using design considerations borrowed from the field of machine learning into: (1) unsupervised methods, (2) supervised non-deep learning, and (3) supervised deep learning methods.

Early onset detection methods for music were unsupervised and relied exclusively on the time domain signals (Gordon, 1985; Schloss, 1985). More recent approaches incorporated time-frequency representations (TFR) of audio signals for computation of different onset detection functions. Multiple TRF-based methods were reviewed by Bello et al. (2005) and Dixon (2006) and include high frequency content (Masri and Bateman, 1996), spectral difference (Duxbury, 2005), phase deviation (Bello and Sandler, 2003) and complex spectral difference (Bello et al., 2004). Some variants, such as SuperFlux by Böck and Widmer (2013b) were proposed to mitigate the negative effect of loudness variations of steady tones in non-percussive instruments by utilising phase information from STFT and combining phase information of several frequency bands into one using local group delay weighting algorithm (Böck and Widmer, 2013a). Unsupervised approaches have advantages such as not requiring system training on large amounts of data and computational efficiency.

Supervised non-deep learning methods rely on statistical models and probabilistic inference for detection of onset change points within a signal (Bello et al., 2005). Jehan (1997) presented a Gaussian autoregressive (AR) model for creation of an ODF that indicated presence or absence (i.e., changes in polarity) of an onset in a music signal. Davy and Godsill (2002) created an ODF by combining a method based on spectral difference with support vector classification (SVC) to improve sharpness of peaks located at event locations.

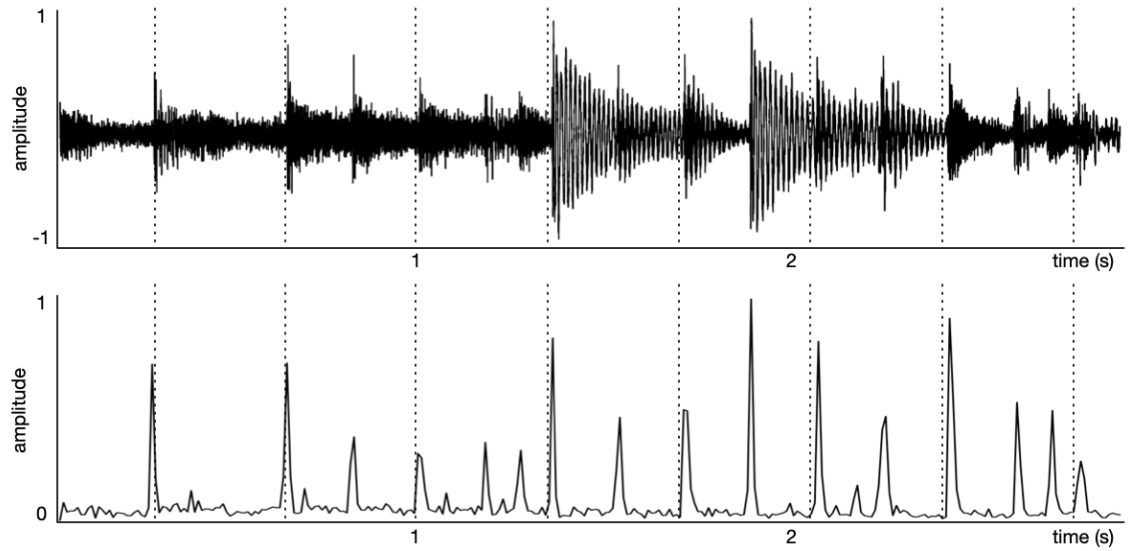

Figure 2.7: Spectral difference onset detection function (i.e., rhythmic envelope) visualisation for a short audio recording. Vertical dotted lines indicate beat locations. Audio signal (top) was used to extract rhythmic envelope (bottom) using spectral difference onset detection function.

An alternative approach for ODF creation relies on supervised deep learning models for classification of STFT frames. Lacoste and Eck (2006) proposed a feed-forward neural network to perform a decision for spectral frames with onsets and without onsets. A similar approach was proposed by Eyben et al. (2010) who applied a recurrent neural network (RNN) with a capability to utilise spectral frames from either side of an onset location. This model used a bidirectional recurrent network (BRNN) that takes into account both past and future information in detection of onsets. In subsequent works, state-of-the-art approaches have been established by means of using RNNs (Böck, 2016; Böck et al., 2012b) and convolutional neural networks (CNN) (Schlüter and Böck, 2013). Deep learning models were recognised as a significant advancement in onset detection, as evidenced by evaluations in the music information retrieval evaluation exchange (MIREX) campaign (MIREX, 2018a). These models have since been extended to include other architectures, such as temporal convolutional networks (TCN) (Fonseca et al., 2021), and encoder-decoder transformer models (Hawthorne et al., 2021) for the task of onset detection.

In the last stage, peaks in the onset detection function are automatically selected and associated with with discrete time locations in a peak picking-process. Peak picking can be achieved with techniques such as local maxima and adaptive threshold calculation as well as probabilistic determination. In simple local maxima calculation, an onset is selected if the current location in an ODF exceeds both a value of the neighbouring positions and a value of a preset threshold. To enhance the selection of peaks from different ODF types, a dynamic threshold based on a weighted median filter calculation may be used (Bello et al., 2004). Toh et al. (2008) presented a statistical method based on two Gaussian mixture models for classification of singing voice audio frames into onset or no-onset classes. Additionally, Pons et al. (2017) proposed a hidden Markov inference approach for onset selection in scenarios where musical score is provided.

2.2.3 Beat Detection

Beat tracking algorithms attempt to estimate a set of beat times from an audio recording, which would match those given by a trained human musician and correspond with the metrical level of the tactus. Automatic detection of beats precisely where a listener would tap to music is a nontrivial task, especially with varying tempi. To achieve this both the beat period and the beat phase must be identified. The beat period represents the duration between beat events and can be used in the computation of tempo measured in beats per minute (BPM), while beat phase refers to the temporal locations of beats. Accurate beat detection provides a range of possibilities for different automated rhythmic transformation applications, such as time-stretching of audio loops (Hockman et al., 2008), self-adapting digital audio effects (Reiss and Brandtsegg, 2018) and beat synchronised DJ mixing (Davies et al., 2014a).

Most methods for beat detection in audio signals consist of three stages: (1) feature extraction, (2) periodicity analysis, and (3) phase detection. Typically these methods use spectral features, onset detection functions, and onset times. A multiple-agent based induction, autocorrelation, comb filterbank, and histogram are standard techniques by which periodicity estimation has been performed. Some methods output beat period and phase information after the periodicity analysis (Dixon, 2001, 2006), whereas other require an additional phase detection stage to determine the exact locations of beat pulses (Scheirer, 1998).

The first beat tracking system (BTS) for audio was developed by Goto and Muraoka (1994) and used a series of inter-onset intervals (IOI) processed by a multiple agent architecture which assessed the validity of several hypotheses of beat locations. The system learned characteristic frequencies of kick and snare drums to determine beat period, beat phase and beat type (i.e., strong or weak) using 28 agents for selection of the most likely hypothesis at the current location. Multiple agent approach has also been proposed for the study of expressive timing by the authors in Dixon (2001). Scheirer (1998) proposed the use of a comb filter bank for beat period estimation that was capable of simultaneous extraction of tempo and beat phase. Autocorrelation methods for periodicity analysis have been used by several authors (Davies, 2007; Ellis, 2007). This approach is computationally less complex than comb filter bank methods, but requires an additional computation stage for beat phase estimation. Periodicity analysis has also been performed using an IOI histogram by combining note lengths in bins that represent discrete time intervals associated with possible periods (Dixon, 2007).

More recent beat detection methods have incorporated deep learning for the determination of beat period without the need for additional processing required to locate beat locations from onsets. These techniques typically use supervised classification methods trained on individual spectral frames to output a beat activation function. In the final stage, a peak-picking algorithm is often used to improve the accuracy of the system (Böck, 2016). The workflow presented in Böck and Schedl (2011) specifically employed bidirectional recurrent neural networks (RNN) for tempo and beat detection. This methodology achieved exemplary performance, as evidenced by its high ranking in the MIREX evaluation of beat tracking systems (MIREX, 2019). Additionally, temporal convolutional networks have been introduced

as an alternative to RNN systems and have demonstrated state-of-the-art performance in beat detection evaluations (Böck and Davies, 2020). Despite the advancements in beat detection methodologies, there still exists a variety of challenges for automated rhythmic transformation systems, particularly in adapting to diverse musical genres, complex rhythms, and varying tempi as well as the detection of rhythmic pattern boundaries which are typically demarcated by downbeats.

2.2.4 Downbeat Detection

The downbeat is the first beat of the bar which often demarcates beginnings of rhythmic patterns as well as chord changes and harmonic cues in music. Downbeat detection algorithms seek to find these positions within musical recordings and are crucial for many commercial music applications and high-level tasks related to automatic music structure analysis. In recent years, multiple computational approaches to downbeat detection have been proposed in the literature.

Goto (2001) implemented a computational downbeat detection model for audio by combining knowledge about onset times, chord changes and drum patterns. Davies and Plumbley (2006) presented a similar method that uses spectral difference of inter-beat intervals for finding downbeats from found beat positions. Klapuri et al. (2006) performed a joint analysis of the beat, sub-beat and metre periods at different hierarchical beat levels using a first-order Markov model. Peeters and Papadopoulos (2011) estimate beat position templates with hidden Markov models (HMM), and then use reverse Viterbi decoding to associate these positions with the downbeats. Hockman (2014) introduced a genre-specific system trained with a dataset of drum recordings using support vector regression for detection of most likely downbeat positions designed for electronic music genres. The dataset consisted of percussion recordings from HJDB genres with complex rhythmic patterns performed at fast tempi. Krebs et al. (2014) jointly model downbeats, tempo, and typical rhythmic patterns of ballroom dance music with a dynamic Bayesian network. This work was later improved in Krebs et al. (2015) by introducing a more effective state space learned by the model. More recently, classification methods utilising deep neural networks have achieved best results for downbeat tracking. While the downbeat detection competition in MIREX (2016) determined that convolutional neural networks (CNN) achieve the highest results (Böck et al., 2016b; Durand et al., 2016), subsequent research extended this approach to incorporate temporal convolutional networks (Böck and Davies, 2020; Böck et al., 2019).

2.2.5 Automatic Drum Transcription

Automatic drum transcription (ADT) systems aim to generate a symbolic representation akin to that shown in Figure 2.4 and have typically put predominant focus on the detection of kick drums, snare drums and hi-hats (Southall, 2019). Wu et al. (2018) proposed several ADT-related sub-tasks that identify drums in simple context such as classification of isolated drum recordings to more challenging contexts that strive to detect drums within polyphonic music with multiple percussion and non-percussion instruments.

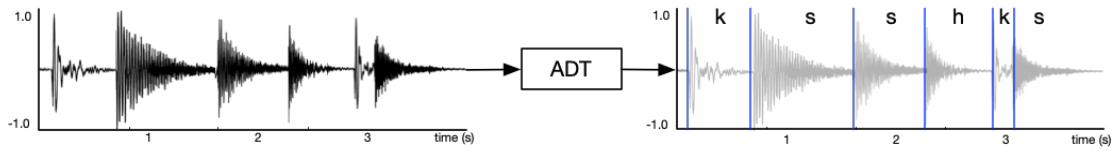

Figure 2.8: Example of percussion detection analysis. Input signal (left) is processed with an automatic drum transcription system that outputs onset times and drum types (right) with labels: kick drum (k), snare drum (s), hi-hat (h). Note that each drum segment can contain multiple overlapped drums (e.g., kick drum and hi-hat playing together).

Existing ADT approaches can be categorised into: (1) segmentation-based, (2) classification-based, (3) language model-based and (4) activation-based methods.[10]

Schloss (1985) proposed an ADT system that categorised between various conga sounds using amplitude thresholding segmentation and binary classification. The system by Schloss (1985) and other segmentation-based systems typically consist of an event segmentation and classification tasks, which utilise feature representations such as the onset detection function described in Section 2.2.2. Tzanetakis et al. (2005) proposed a system that converted percussive audio input into a new feature representation that used preset frequency ranges for each instrument with band-pass filtering (ranges 30–280 Hz for kick and 2.7–5.5 kHz for hi-hat). The classification-based methods rely on a two-stage process of event segmentation and subsequent classification of the detected events. Classification-based approaches typically use features such as spectral features (e.g., centroid, flux, flatness), temporal features (e.g., zero crossing rate, RMS, rhythmic envelopes), and Mel-frequency cepstral coefficients (Bello et al., 2006; Gillet and Richard, 2008; Gouyon et al., 2000; Herrera et al., 2002; Pampalk et al., 2008). Multiple methods in the literature have also used supervised classifiers such as k-nearest neighbours (Herrera et al., 2002), and support vector machines (Gillet and Richard, 2008). Language model-based methods predict future drum events based on the past events. Previously proposed language model-based approaches typically utilised the Viterbi algorithm (Nakano et al., 2005) and hidden Markov models (Paulus and Klapuri, 2009). The activation-based methods create a type of onset detection function for drum events referred to as the activation function. ADT approaches based on activation functions can be classified into two groups. In the first group, systems apply matrix factorisation algorithms to an audio spectrogram and decompose it into multiple drum specific activation functions. Non-negative matrix factorisation is typically used (Battenberg et al., 2012; Paulus and Virtanen, 2005; Wu and Lerch, 2015). The second group uses deep neural networks to create an activation function for each of the investigated drums. In recent years, the proposed systems have relied on feedforward neural networks (Southall et al., 2016), recurrent neural networks (Southall et al., 2016), convolutional neural networks (Southall et al., 2017; Vogl et al., 2017), and convolutional recurrent neural networks (Southall et al., 2018). The current state-of-the-art approaches to ADT as presented in the MIREX campaign rely on deep neural networks (MIREX, 2018b).

[10] Termed design groups in (Wu et al., 2018).

2.2.6 Rhythmic Pattern Detection

Automatic methods for rhythmic pattern detection in MIR seek to detect repetitive patterns typically covering temporal scopes of one or a couple of bars, and are closely related to the notion of rhythmic similarity (Dixon et al., 2004; Paulus and Klapuri, 2002; Toussaint, 2004). While the concept of similarity is broadly studied outside of MIR (Gärdenfors, 2004), it is also one of the fundamental problems in computational music theory as it relates to human perception of rhythm (see Section 2.2.1). In the context of music applications for automated rhythmic transformation, rhythmic similarity serves the purpose of establishing quantitative relationships between drum patterns and can be used to gain insight into the affinity between patterns of different styles. In recent years, multiple computational approaches to automatic rhythmic pattern detection have been proposed; they require prior knowledge of metrical structure including subsequent detection of beats and downbeats, and are often based on binary (see Figure 2.3) and rhythmic envelope (see Figure 2.7) representations of rhythmic patterns.

Monophonic rhythmic patterns are commonly represented as a binary sequence of ones and zeros, where a zero indicates a rest and a one indicates a beat or an event onset (Thul and Toussaint, 2008; Toussaint, 2004). The time-unit box representation in Figure 2.3 illustrates three monophonic patterns for different instruments of the drum kit (i.e., kick, snare and hi-hat) as sequences of filled and empty boxes representing events and rests, respectively. When combined, the three patterns create a typical eighth note drum pattern core to a variety of genres present in popular music. Alternatively, rhythmic envelopes illustrated in Figure 2.7 have been utilised in detection and classification of rhythmic patterns (Dixon et al., 2004; Ellis and Arroyo, 2004; Pikrakis, 2013).

Gabrielsson (1973a,b) presented similarity ratings of rhythmic patterns from an early drum machine based on several subjective studies. Gabrielsson utilised multidimensional scaling (MDS) for mapping subject similarity ratings to rhythmic pattern dimensions, termed rhythm spaces in his study, resulting in two or three axes attributed to characteristics such as metrical pulse, rhythmic complexity and typical patterns. Bilmes (1993) implemented a similarity-based rhythmic pattern clustering technique using a modified k-means algorithm that did not require a predefined number of pattern clusters. The final clusters represented groups of bar-length patterns with drum events quantised to a sixteenth note grid, which were maximally similar according to the chosen distance metric. Additionally, several approaches have been proposed for the determination of typical rhythmic patterns in music classification tasks (Dixon et al., 2004; Paulus and Klapuri, 2002; Peeters, 2005). Typically, after the detection of beats and downbeats, these approaches extract some types of rhythmic features (e.g., rhythmic envelopes) from a set of patterns to determine an average rhythmic pattern for the entire musical piece or a set of pieces that belong to the same genre. Paulus and Klapuri (2002) proposed a method that utilised a dynamic time warping (DTW) algorithm for the calculation of similarity between two rhythmic patterns using a rhythmic feature based on the spectral centroid weighted with the log-energy of the signal. Dixon et al. (2004) proposed to use rhythmic envelopes extracted from the signal using an root mean squared filter followed by the k-means clustering to determine the prominent rhythmic

patterns. Peeters (2005) combined features extracted using the discrete Fourier transform and an autocorrelation function to produce spectral rhythmic patterns for different musical genres. Ellis and Arroyo (2004) implemented a system for the detection of typical patterns in drum recordings based on principal component analysis (PCA) for music classification and generation tasks. In their system, the typical patterns were represented by the means of the top eigenvector bases—termed eigenrhythms—that explained 90% of the variance. More recently, Krebs et al. (2013) utilised dynamic Bayesian networks for rhythmic pattern modelling of different ball room dances represented as separate distributions in a Gaussian mixture model (GMM). The authors proposed to use a rhythmic envelope feature based on the spectral flux algorithm, which was previously presented in larger evaluations of onset detection algorithms in Böck et al. (2012b). More recently, Foroughmand and Peeters (2019) incorporated convolutional neural networks for the modelling of rhythmic pattern classes and tempo estimates. Rhythmic pattern detection has seen a variety of methods that aim to understand and quantify rhythmic patterns and their similarities across various musical genres and styles. This pursuit remains central to enhancing automated rhythmic transformation systems, striving to better align computational approaches with human understanding of rhythm.

2.3 Existing Methods for Automated Rhythmic Transformation

The previous sections have presented characteristics of drum instruments (Section 2.1) as well as computational models for rhythmic description (Section 2.2). This section reviews automated rhythmic transformation systems in the literature. These methods can be achieved through a combination of techniques from digital signal processing (DSP) and deep learning (DL) fields, and belong to a broader category of computational transformations of musical content. Section 2.3.1 presents an overview of a general content-based transformation framework that was first introduced in digital audio effects. This general framework can be expanded into two methods for automated rhythmic transformation and is discussed in the subsequent sections. Section 2.3.2 presents resequencing-based methods that represent one of the earliest approaches for rhythmic transformation of audio signals. Section 2.3.3 covers hybrid-based methods, where the rhythmic transformation is executed as a blend of two or more audio recordings or by generating a new audio output from a learned representation of a deep generative model.

2.3.1 Content-based Transformations

Content-based audio effects were distinguished from other types of digital audio effects by Amatriain et al. (2003) to address transformations that manipulate higher-level information within an audio signal as opposed to effects that used only simple representation of the sound (i.e., audio samples). Content-based transformation systems can be seen as part of a larger set of support systems to guide users when they lack inspiration, technical knowledge, musical capability as it relates to melody, harmony, rhythm, structure or style (Knees et al., 2015). Content-based effects can be classified according to

Figure 2.9: Diagram of a general analysis/synthesis transformation system. Dashed line represents an additional input option (e.g., for a recording of another style). Additional inputs are characteristic of automated rhythmic transformation systems that facilitate automatic remixing or mashup creation.

the perceptual attribute that is primarily altered by the used method. Amatriain et al. (2003) proposed six axes of content-based transformation: loudness, duration, pitch, position, quality, and timbre. This categorisation was condensed to five axes in Zölzer (2011) to loudness, time, pitch, spatial hearing, and timbre. Ideally, content-based transformation would affect only one dimension without changing others. This is not always possible and depends on the musical material and transformation technique used. For example, the rhythmic content of an audio signal can refer to rhythmic attributes discussed in Sections 2.1 and 2.2 such as tempo, beats, swing ratios, and rhythmic patterns. In this scenario, changing one rhythmic dimension can also have an effect on others. Amatriain et al. (2003) outlined a three-stage process—with analysis, transformation, and synthesis stages—for a content-based transformation, where one or more audio inputs is affected. This general process, visualised in Figure 2.9, borrows from design of digital audio effects that target perceptual attributes of sound (Verfaille et al., 2006). In this framework, the analysis stage refers to creation of a sound representation such as STFT from audio samples and application of some changes (e.g., time-stretching) to it in the transformation stage. The synthesis stage refers to conversion of the transformed sound representation back into audio samples.

Other transformation scenarios can include various configurations of user and metadata inputs at different stages of this process (Amatriain et al., 2003). An important characteristic of content-based rhythmic transformation systems presented in the subsequent sections is the option to add an additional musical input (see Figure 2.9) that facilitates the blending of rhythmic and other perceptual qualities in remix and mashup creation that have been explored in a variety of systems. This characteristic can be motivated by the need for manipulation of the source musical material by characteristics of sound inspired by an already existing body of work from other musicians (Knees et al., 2016). Often musicians, desiring a certain sound or aesthetic influenced by the style of artists they admire, replace the rhythmic pattern of drums in their recordings (i.e., source) with that from an idealised recording (i.e., target). Multiple automated methods satisfy this transformation scenario (Gouyon, 2003; Hockman et al., 2008; Ravelli et al., 2007). However, the work presented in Chapters 4, 5, and 6 of this dissertation extends this transformation scenario to the generation of new drum sounds within the paradigm of user-controlled rhythmic transformation.

2.3.2 Resequencing Methods

Figure 2.10 showcases drum segment resequencing from a drum recording comprised of two patterns. Alternatively, the degree that each segment is altered can vary to accommodate changes in microtiming

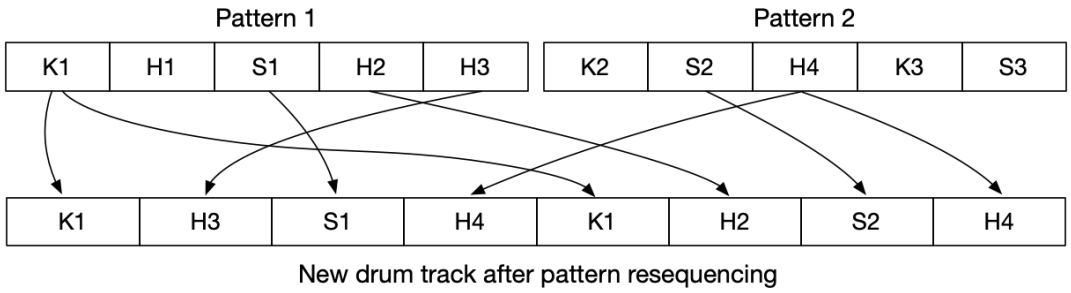

Figure 2.10: Resequencing example of a drum pattern. The pattern is divided into bars with individual hits, such as kick drums (K), hi-hats (H), and snare drums (S). Some of these hits (as represented by the boxes at the top) are repurposed to form a new percussion arrangement shown below. Figure adapted from Hockman and Davies (2015).

of events (Gouyon et al., 2003; Hockman et al., 2008). Resequencing methods rearrange a sequence of audio segments within a drum recording using an extracted knowledge of rhythmic patterns. The new arrangement does not need to contain all segments of the original recording as shown in a system for breakbeat resequencing by Hockman and Davies (2015).[11]

One of the earliest automated rhythmic transformation systems was proposed by Bilmes (1993) and focused on manipulation of sequenced prerecorded material. It extracted note onset times from digitised multitrack recordings of Cuban percussion instruments and recreated their performance by inserting drum samples with new calculated microtiming deviations. Bilmes' work emphasised the importance of tatum analysis for representing expressivity in percussive recordings. Gouyon et al. (2003) used a phase-vocoder approach, to modify rhythmic performance of polyphonic audio signals based on an input-defined swing ratio. Their work created a metrically-informed swing modification of the signal but was limited by the lack of information about affected drum types and more precise rhythmic pattern representation. In addition, Gouyon et al. (2003) outlined the importance of accurate rhythmic analysis that can otherwise contribute to unpleasant errors such as phasiness in time-stretching and a wrapping effect in rhythmic pattern matching. Janer et al. (2006) extended the system from Gouyon et al. (2003) by implementing transformation of tempo, swing, meter, and accent in a real-time audio plugin. This method performed well on drum recordings with regular tempo but suffered from octave errors (e.g., BPM incorrectly detected as half of the true rate) that are a common problem in beat detection algorithms (Janer et al., 2006). Yoshii et al. (2007) implement a real-time system capable of drum pattern rearrangement, as well as loudness and timbre transformation of selected audio segments. Each segment in the original drum arrangement could be automatically replaced with drum segments from other recordings. Ravelli et al. (2007) presented a system that performed adaptation of source drum loops, based on rhythmic information extracted from a target loop. In their approach, drum events of different types could be either time-stretched to suitable target events in the model, or reordered. Figure 2.11 illustrates these time-stretching and resequencing transformations of drum recordings, where a source recording is progressively edited to match the rhythmic pattern and drum characteristics (i.e., timbres) of the target

[11]Breakbeats, or breaks, refer to sampled recordings of solo percussionists generally from Funk or Jazz music of the 1960s to the 1980s. A notable example is the *Amen* break, originally sampled from The Winstons' Amen, Brother (1969), that has since become one of the most sampled song in music history (Reynolds, 2012).

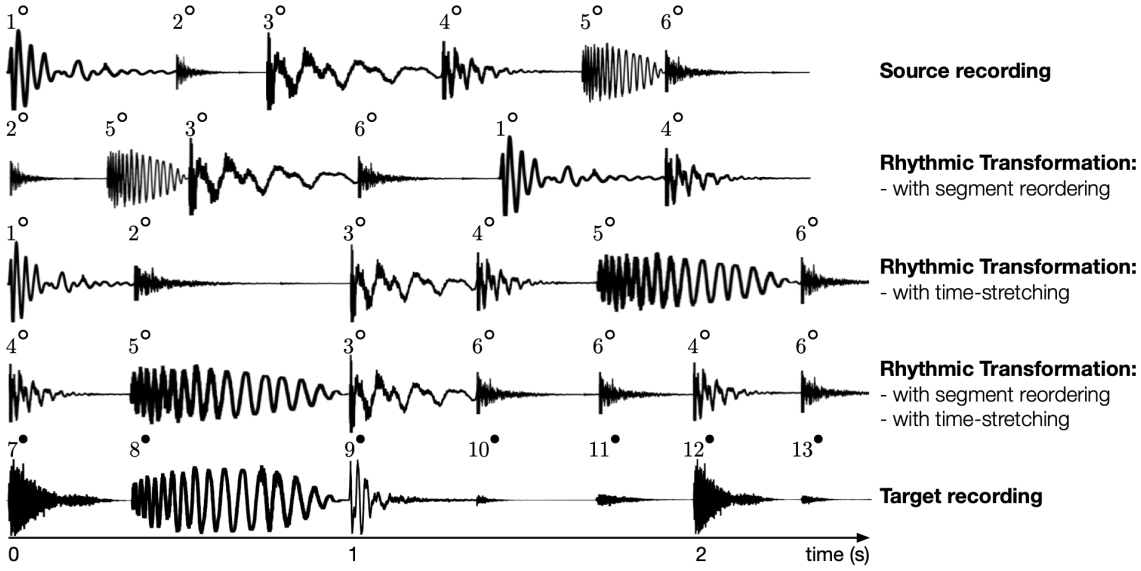

Figure 2.11: Rhythmic transformation using signal processing techniques. Source recording segments and its transformations using time-stretching and segment reordering with the aim of matching the rhythmic pattern and drum types of the source to those of the target. The ○ and • symbols denote drum timbre characteristics (e.g., from different drum kits) of source and target recordings, respectively.

recording. An extension of their work was proposed by Hockman et al. (2008) with an aim to analyse downbeats and intra-measure infrastructure of recordings prior to rhythmic transformation. Similarly, time-stretching was performed on a sequence of onset times corresponding to the rhythmic pattern of the source recording to match the pattern onsets of the model recording. In this transformation, if the ratio between the new and original tempi (scalar ratio) is increased or reduced by too great a factor, then transient regions of the pattern onsets may become smeared or artificial, resulting in a perceptual loss of audio quality (Hockman et al., 2008). Cocharro et al. (2014) implement a system for manipulation of syncopation in drum loops, where the input audio is analysed for drum events and quantised to a binary pattern prior to a time-stretching transformation illustrated in Figure 2.11. More recently, Sawada et al. (2019) proposed a system that automatically extracted typical drum patterns from source and target recordings utilising harmonic-percussive source separation (HPSS) and automatic drum transcription. A new audio track was then synthesised with automatically rearranged bar-length patterns by mixing the harmonic source-separated part of the source recording with preset drum samples triggered by onset events of the target detected patterns.

The advantage of resequencing methods is that they do not require computationally intensive models and can be often implemented as real-time systems. However, as they rely on signal processing techniques for sound segmentation, pattern matching (i.e., segment alignment between different metrical levels such as beats or note onsets), and time-stretching (e.g., using a phase vocoder) to satisfy the target transformation, the errors in early stages cascade to the later stages of the transformation. Also, initial analysis is responsible for artefacts such as transient smearing caused by incorrect demarcation of temporally-relevant event positions or spectral artefacts from source separation.

Outside of resequencing methods operating directly on audio signals, automated rhythmic transformation of drums were explored in symbolic music domain (e.g., MIDI). Lattner and Grachten (2019) proposed to use convolutional gated autoencoders for conditional generation and manipulation of drum patterns given snare and bass tracks in different music styles. In Gillick et al. (2019), the authors propose a generative model based on variational autoencoders for user-controlled drum performance generation in popular music genres performed by different professional drummers. Valenti et al. (2020) propose to use adversarial regularisation as a flexible mean to imbue recurrent VAEs with information about different popular music styles. Briot and Pachet (2017) and Ji et al. (2020) provide extensive overviews of systems for symbolic music generation and transformation with deep generative models.

2.3.3 Hybrid Methods

Hybrid methods refer to transformation systems that mix content characteristics of two or more recordings by combining them together or by generating an audio output anew. This transformation can be a result of a combination of two songs in the form of a musical mashup (Shiga, 2007) but can also refer to generation of new audio material based on an internal representation created within a deep neural network. Additionally, audio inputs can be resequenced to match the rhythmic timing of another target audio input and subsequently combined using a cross-synthesis (i.e., a technique of impressing the spectral envelope of one sound on the flattened spectrum of another), as proposed by Hockman (2007). Another cross-synthesis method by Collins and Sturm (2011), comes from dictionary-based approaches, where audio signals are atomically decomposed using scale-time-frequency dictionaries, offering some control over sound transformations. This atomic decomposition allows for creative manipulations, including the ability to influence one signal's decomposition with another, for unique cross-synthesised sounds. Such hybrid methods span from cross-synthesis, automatic mashups, and remixing, to neural audio synthesis approaches such as audio style transfer.

Several web applications exist for users that wish to listen to or create mashups by hand, such as a system developed by Tokui (2008) with an interactive user interface, and a system by Griffin et al. (2010) capable of beat-synchronous time-stretching of an input song to the target tempo. Ishizaki et al. (2009) demonstrated that mashup transformations that are not beat-synchronised correlate with user discomfort. To mitigate this, Davies et al. (2013) proposed a mashability parameter that considered rhythmic and harmonic similarity scores for one or more audio recordings. These scores were based on beat-synchronous feature representations that aimed at segmenting the input audio at harmonically consistent locations that coincided with downbeats. The automated multi-song music mashup system from Davies et al. (2013, 2014a) has been extended to a real-time audio effect in Davies et al. (2014b). In this system, time-stretching and pitch-shifting algorithms were used to transform songs at different tempi and musical keys within preset ranges limiting sonic artefacts in the output.

Upon this past research, the contemporary development of audio style transfer (AST) emerged, leveraging deep generative models (DGM) to transform high-level musical characteristics. Pioneered

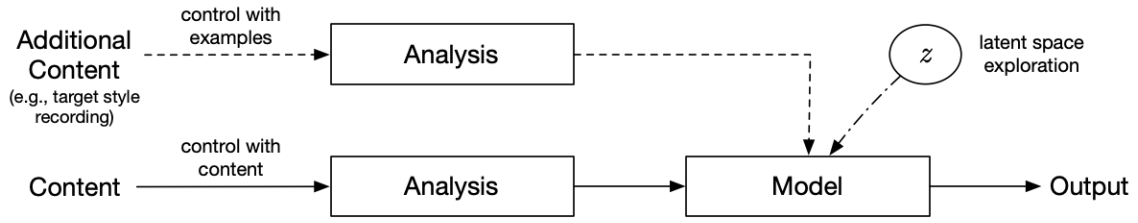

Figure 2.12: Diagram of a transformation system using deep learning model with distinction of three variants for content manipulation: (1) with input content, (2) with additional content, and (3) with a latent space of a trained model. Bold lines represent the typical transformation signal flow. Dashed lines represent additional inputs and dotted-dashed line represents an independent control over the transformation output with parameter space of the model.

by Foote et al. (2016) and Ulyanov and Lebedev (2016), it adapted image-based algorithms like the one from Gatys et al. (2015) to the audio domain. In essence, AST blends timbral characteristics of two recordings—labelled as "style" and "content". A visual representation of a standard AST model is detailed in Figure 2.12.

More recently, audio style transfer (AST) has emerged as a method to manipulate the high-level musical attributes of audio signals using deep generative models (DGM). AST was first attempted by Foote et al. (2016) and Ulyanov and Lebedev (2016), which directly extended an algorithm proposed for images by Gatys et al. (2015). In AST, timbral characteristics of two recordings—one termed style and another termed content—are mixed together in a form of a mashup that allows the transfer of style characteristics from one to the other. Figure 2.12 shows a diagram for a typical AST, where the control over the output can be achieved with additional audio inputs (e.g., style recording) as well as parameters of a trained deep learning model. A new audio signal can be generated and transformed through an inference process applied to a learned feature space of a trained model in form of latent space exploration.

For percussion instruments, audio style transfer can resemble a drum replacement process referred to as *redrumming*, in which musicians desiring a certain sound influenced by the style of artists they admire, replace their drum sounds with those from an idealised recording. Typically redrumming describes functionality of hardware or virtual drum machines and relies on substitution of drum hits in a track with single (i.e., one-shot) drum sounds from a personal collection. Here, the following methods for AST make use of a Gram matrix feature representation (i.e., inner product between neural feature maps) for musical style that has been suggested to represent timbre (Grinstein et al., 2017; Verma and Smith, 2018). Additionally, approaches to AST can be divided into two categories: (1) time-frequency domain (i.e., spectrogram) based, where log-magnitudes of a short-time Fourier transform (STFT) are used as inputs to a CNN that performs the style transformation followed by a process of phase reconstruction; and (2) time-domain (i.e., raw audio) based, where the audio samples are directly optimised, removing the need for additional phase reconstruction. Grinstein et al. (2017) introduced a spectral filtering method based on a sound texture model to improve the transformation of timbre from style directly onto a new audio initialised as content sound using multiple pre-trained neural networks. Similarly, Wyse (2017) explored the effects of pre-trained weights from a network trained on an audio classification

dataset for AST. The presented system appears to generate a more integrated transformation of content and style with the included pre-trained network. In Verma and Smith (2018) the authors provide an additional loss term that constrains the temporal envelope of the newly generated spectrogram to match that of the style recording. The motivation for the additional loss function was to better portray the temporal dynamics of the style recording and diminish the impact of the content recording. Audio style transfer was also used in the attempt to change the style of prosodic speech by Perez et al. (2017). The authors report success in transferring low-level textural features of the content but difficulty in transferring the high-level prosody such as emotion or accent of the style voice recording. Mital (2017) combines information from multiple discrete Fourier transform parts and presents them as different concatenated batches (i.e., layers) of a convolutional filter. Concatenated real, imaginary, and magnitude features produced the best results. Barry and Kim (2018) implemented a parallel architecture with deep specialised networks to improve the key invariance and increase the temporal memory of the network through the use of constant-Q transform, and dilated convolutions (see Section 3.1.2), respectively. The above AST methods utilise two specialised *content* and *style* loss functions that require a retraining or a reoptimisation phase to manipulate the output transformation. More recently, other DGMs have replaced the need to retrain a network after each transformation by utilising a training strategy that involves conditioning on the target music characteristic (e.g., rhythmic patterns and drum timbres).

DGMs such as variational autoencoders (VAE) (Kingma and Welling, 2013) and generative adversarial networks (GAN) (Goodfellow et al., 2014) have seen increasing success in various fields through targeting the task of learning and manipulation of disentangled feature representations. Disentangled representations denote techniques that break down each input data feature into narrowly defined variables to be encoded into separate dimensions. Such features can be linked to different examples of musical content such as timbre, rhythmic patterns and musical style.

Dieleman et al. (2018) adapted the WaveNet architecture by Oord et al. (2016a) for raw audio generation of piano performances at timescales across tens of seconds. Engel et al. (2017) proposed a WaveNet autoencoder that learns codes that meaningfully represent the space of musical instruments with the ability to model long temporal dependencies. This work was later extended by Engel et al. (2017) to the NSynth system, a neural synthesiser capable of generating new sound embeddings learned from a large dataset of musical notes. Dieleman et al. (2018) adapted the WaveNet architecture for the unconditional generation of piano music that exhibits stylistic consistency at longer timescales across tens of seconds. In Manzelli et al. (2018), the authors combined audio and symbolic models and use a long short-term memory recurrent neural network (RNN) to learn melodic structures of different styles of music, which are used as conditioning input to a WaveNet-based instrument melody generator. Other AR models include RNN-based architectures such as: VRNN (Chung et al., 2015), SampleRNN (Mehri et al., 2016) and WaveRNN (Kalchbrenner et al., 2018). Alternatively, the WaveNet architecture has been used in the context of musical timbre transfer. Huang et al. (2018) adapted an image-based style transfer method by Zhu et al. (2017) for translation of an image from one domain to another using a conditional WaveNet synthesiser within the TimbreTron model. Kim et al. (2019) proposed a

music synthesis system with timbre control that learns to generate spectrograms from symbolic music representations and instrument embeddings, and generates raw audio with a WaveNet vocoder.

Additionally, AST has been proposed using generative adversarial networks. Veire et al. (2019) incorporated the CycleGAN network architecture by the authors in Zhu et al. (2017) to perform the style transformation between different electronic music genres. The authors presented a method for generation of a new output that combined phase information of the input recording with the transformation output to improve quality as well as a method that relied fully on the Griffin-Lim reconstruction algorithm. Mor et al. (2018, 2019) introduced a system for timbre and style translations that combined the WaveNet autoencoder with unsupervised adversarial training. Dhariwal et al. (2020) presented a Jukebox model based on vector-quantised variational autoencoder by Razavi et al. (2019) for generation of musical performances in the raw audio domain conditioned on styles learned from different artists and genres. Wu et al. (2022) extended the Jukebox model to synthesise and add a drum part to a drumfree recording.

Standard audio style transfer methods have highlighted certain inherent limitations. For instance, they often rely on Gram matrix feature representations. Despite their capabilities in transferring low-level features, these representations might not fully capture the nuances of comprehensive musical styles and have not been explored for controlled rhythmic transformation. While DGMs have delved into various feature representations, the exploration of rhythmic transformation has largely been underrepresented in neural audio synthesis research. Additionally, a prevalent dependency on continual re-optimisation or re-training to effectuate output changes posed computational and efficiency challenges. Transformation of rhythmic patterns specialised on percussive instruments remains an unexplored area. This thesis bridges these gaps by introducing deep generative models for automated rhythmic transformation. Building on previous work predominantly focused on pitched instruments, this research explores novel approaches specifically tailored for transformation of rhythmic patterns of percussion recordings.

2.4 Chapter Summary

This chapter delved into the rhythmic styles that characterise playing techniques of the modern Western drum kit. In doing so, it described the interconnected nature of rhythmic styles and their significance in percussion-based music. This exploration also highlighted the challenges and nuances inherent in the rhythmic analysis of drum recordings. Central to this chapter was a detailed review of the methods for automated rhythmic transformation, contextualised within broader content-based transformation systems. These transformation methods predominantly fall into two categories: resequencing-based and hybrid methods. The former harnesses traditional signal processing techniques, with particular emphasis on segment reordering and time-stretching—both essential in modern music production. Hybrid methods, on the other hand, signify a progressive shift towards the use of deep generative modelling, positioning themselves at the forefront of neural audio synthesis. Especially noteworthy is their innovative application in redrumming, leveraging the audio style transfer approach. Building on the foundation of hybrid methods, the chapter discussed two defining feature representation domains: time-frequency domain

and time-domain. Each domain offers unique methodologies and holds specific implications for the larger context of rhythmic analysis and transformation. Both are explored in this thesis. As this chapter concluded, it laid the foundation for subsequent chapters, notably indicating the integration of deep learning techniques which are presented in Chapter 3. Such techniques are fundamental to the advanced rhythmic transformation approaches proposed in the upcoming Chapters 4–6.

Chapter 3

Deep Learning and Deep Generative Models for Audio Synthesis

The previous chapter presented an overview of automated techniques for description and transformation of rhythm in music signals, outlining advantages and disadvantages of previous rhythmic transformation methods. To provide an understanding of the fundamental concepts and deep generative models (DGM) referred to throughout the thesis, a compact introduction to deep learning will be provided in this chapter together with the proposed transformation modes of DGMs. These fundamentals are necessary not only to understand the performance and capabilities of the implemented systems that are used for the analysis and processing of the existing data, but also for generation of new audio examples.

The following sections are summarised as follows: Deep learning concepts are given with regard to convolutional neural networks in Section 3.1, and an overview of techniques and challenges present during the training process of deep neural networks is discussed in Section 3.1.3. Section 3.2 presents deep generative models and Section 3.3 introduces their transformation modes utilised in the following chapters.

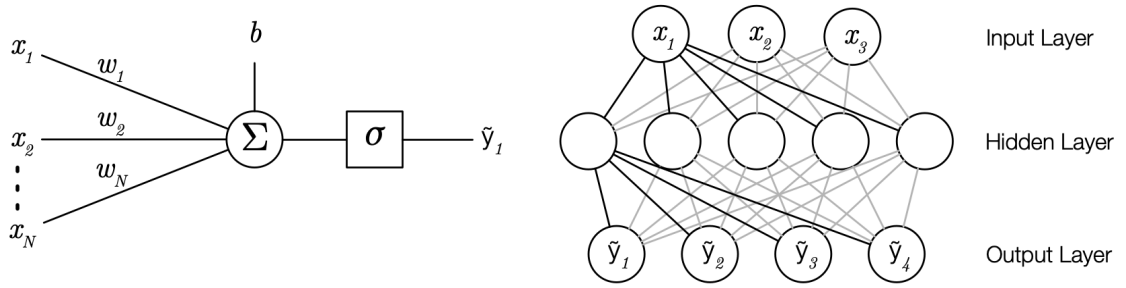

Figure 3.1: The left diagram portrays an example of a neuron for inputs x, corresponding weights w, a bias b, and the activation σ applied to the weighted sum of the inputs. The right diagram portrays a hypothetical multilayer perceptron network with a fully-connected (i.e., dense) input, hidden and output layers.

3.1 Deep Learning

Deep learning is a term used to refer to a sub-family of machine learning algorithms that enables the modelling of complex relationships from data through the use of large neural network architectures. Such algorithms learn without being explicitly programmed and utilise hierarchical architectures structured into multiple layers. Deep neural networks (DNN) operate using supervised, semi-supervised and unsupervised data-driven optimisation approaches. In supervised learning, the user provides a system with labelled data from which to learn. In unsupervised learning the system groups data into classes without user input. Semi-supervised methods leverage subsets of labelled and unlabelled data, especially where only a small proportion of labelled examples exist (Kingma et al., 2014). In the recent years, these methods have become increasingly more popular, as computation capacity and data has become more accessible, allowing many practical applications in the areas of audio signal processing and music information retrieval. Additionally, deep learning has also allowed researchers to extend the typical data analysis and processing tasks to include the generation of new audio data using deep generative networks. Example transformations made possible through the use of deep generative models include style transfer by Gatys et al. (2015) and neural translation transformations by Isola et al. (2017).

In a typical deep learning model, the input observation is first converted into a numerical representation such as amplitude values of a raw audio signal or an audio spectrogram. These input features are then processed by the model to satisfy a specifically designed objective (see Section 3.1.3). Common learning objectives consider tasks for discrimination of labelled inputs (i.e., classification), or estimation of a mapping for a continuous output (i.e., regression), whereas generative models learn to replicate the statistics of individual categories within the observed data (see Section 3.2).

3.1.1 Multilayer Perceptrons

Figure 3.1 shows a multilayer perceptron (MLP) which is a basic module used in deep neural networks. MLPs are a type of a feed-forward neural network that consists of a sequence of layers with neurons, known as fully-connected layers or dense layers. Fully-connected layers can be used to map the input to another space to perform the target task (e.g., classification), where the last layer outputs the result

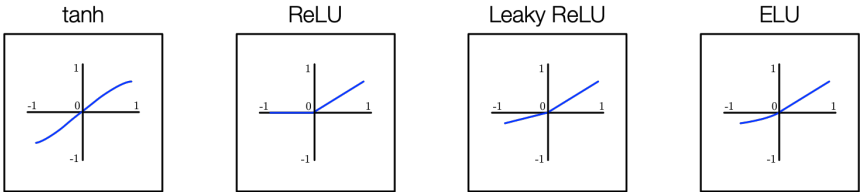

Figure 3.2: The diagram shows four activation functions.

computed by subsequent hidden layers. This supervised neural network uses input features $x \in R^{d_{in}}$ to calculate the target output $y_l \in R^{d_{out}}$, where d represents the number of used features. The output vector $\tilde{y}_l \in R^{d_{out}}$ of a single layer, also referred to as a hidden state or neuron activation, is defined by an affine transformation computed with a non-linearity function σ per each layer l as follows:

$$\tilde{y} = \sigma(W_l^\top x_l) + b_l, \tag{3.1}$$

where x, $\tilde{y}$, and bias $b_l \in R^{d_{out}}$ are vectors, and $W_l \in R^{d_{in} \times d_{out}}$ is a weights matrix.

The nonlinearity functions σ, also referred to as activations, enable neural networks (NN) to learn solutions for nontrivial problems. Activation functions influence the modelling capabilities of NNs and are an important hyperparameter in the architecture design of different networks. Multiple non-linearity functions have been used for training of NNs based on the target application. A widely used activation is the hyperbolic tangent function (tanh), where the gradient of the line is greater towards the origin at 0 within the output ranges of $[-1, 1]$. The output of tanh activation is calculated as follows:

$$tanh(v) = \frac{1 - e^{-2v}}{1 + e^{-2v}}. \tag{3.2}$$

The sigmoid function normalises any real value into another value in ranges $[0, 1]$ and can be used to map predictions to probabilities in binary classification tasks. The sigmoid activation is computed as follows:

$$sigmoid(v) = \frac{1}{1 + e^{-v}}. \tag{3.3}$$

Softmax activation considers the hidden state of the whole layer such that values sum to 1 ($\sum_{j=1}^{J} \tilde{y}_j$). The function maps values to the range of $[0, 1]$ and can be used in multi-label (e.g., one-hot) classification problems, where each neuron represents the probability of an individual class. The softmax activation is calculated as follows:

$$softmax(v) = \frac{e^v}{\sum e^v}. \tag{3.4}$$

Nonlinearities such as the rectified-linear unit (ReLU) activation function act as a hard limiter for negative values setting them to 0 as follows:

$$relu(v) = \begin{cases} v & \text{for } v > 0, \\ 0 & \text{for } v \leqslant 0. \end{cases} \tag{3.5}$$

A popular variation of ReLU is the exponential linear unit (ELU) activation:

$$elu(v) = \begin{cases} v & \text{if } > 0, \\ \alpha(e^v - 1) & \text{if } v \leqslant 0, \end{cases} \tag{3.6}$$

where, the output of the function smoothly approaches $-\alpha$ in contrast to output of ReLU which does not produce negative results. Other variations of ReLU include leaky ReLU (LReLU) (Maas et al., 2013) shown in Figure 3.2 and scaled ELU (SELU) (Klambauer et al., 2017). The LReLU prevents weights of a NN from being held at 0 by allowing negative values, while the SELU activation uses an additional scaling parameter to introduce self-normalising properties of outputs at each layer.

3.1.2 Convolutional Neural Networks

Neural networks described in Section 3.1.1 relied on fully-connected layers, akin to the neural connections present in the human brain, in which every neuron in one layer shares connections with every other neuron in the subsequent layer. For deep neural networks this can rapidly increase memory requirements as well as training time. Convolutional neural networks (CNN) were designed to follow visual mechanisms of living organisms, thus allowing NNs a capability to utilise local structures of the input during training. While fully-connected networks discard the ordering of the data, given that it is ordered in a consistent manner, CNNs emphasise the structure of the input by learning spatial relations present within it. These relations can be learned from images, where individual pixels often represent only a single object in the context of adjacent pixels, and from audio spectrograms, where musical instrument sources are distributed on the horizontal axis (i.e., time) and vertical axis (i.e., frequency).

A convolution operation typically uses same-size window patches around sample points to aggregate raw information, creating a more abstract representation for the next hidden layer. Let, the input values of x_i be represented as a matrix X_i with weights W_{ij}, where i and j indicate input and output dimensions (i.e., channels), respectively, for the n-th shared $M \times M$ weight template for a certain feature. The 2D convolution operation between the shared weights and the input layer results in a full convolutional unit H, which is calculated as follows:

$$H_{cd} = \sigma(\sum_{n=0}^{N} \sum_{i=0}^{M} \sum_{j=0}^{M} W_{ij,n} X_{(c+i)(d+j),n} + B), \tag{3.7}$$

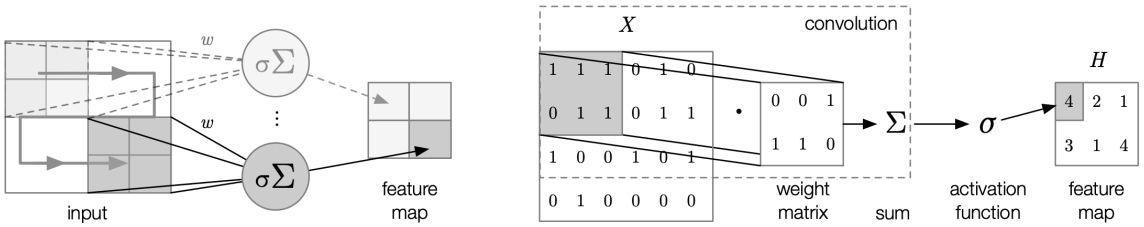

Figure 3.3: An example of a convolution operation, where multiple filters shifted over the input map its features onto feature map representations within the network (left). The convolution operation is shown using a simple example with only a single input and output channels (right), where an activation σ outputs a value onto a feature map after summing the product of an element-wise multiplication between the input matrix and the shared weight matrix.

where N represents the number of original channels or the feature maps of the previous layer, X_{cd} and H_{cd} denote the input and output activation at location (c, d), respectively, and B is a shared bias. For simplification, the notation for layer l from Equation (3.1) is omitted. The output H_j can also be referred to as a feature map and shares weight and bias matrices from the map of the subsequent input layer. The left side of Figure 3.3 shows operation procedure of a convolution layer that takes a full matrix as an input channel and produces a single output channel by shifting a local receptive field (i.e., a small window on the input neurons) across the full input. The right side shows calculation of a feature map, where the convolution operation can be seen as an element-wise multiplication of a filter—also referred to as kernel—with its corresponding local window (i.e., receptive field) followed by summing. The step size of the filter is controlled by a stride parameter that also affects the size of the output channel. The output size of H in Equation (3.7) becomes smaller than the input after the convolution operation, which can be a desired outcome but can be offset through the use of padding (e.g., zero padding). The parameter reduction in the network also contributes to reduced memory requirements and faster training speeds.

The spatial size of feature maps can be downsampled with pooling layers that have a similar operation mode as convolutional layers but instead of element-wise multiplication and summation, use operations such as max and average. For example, max pooling and average pooling layers calculate maximum and average values of the input matrices, respectively. Pooling operations, similarly to convolutions, reduce the number of parameters in a CNN and can also control overfitting (see Section 3.1.4) in deep learning models to some extent.

Figure 3.4 shows a standard CNN architecture created as a combination of convolutional, pooling and fully-connected layers. Multiple types of CNN layers have been used for various applications and data, and the details about the most often used types are presented below.

1×1 Convolution

1×1 convolutions are used for channel-wise pooling before more computationally-expensive convolutions that incorporate larger filters (e.g., 3×3 and 5×5). The 1×1 convolution operation convolves the input with filters of size 1×1, often using zero padding and stride length of 1. This kind of dimensionality

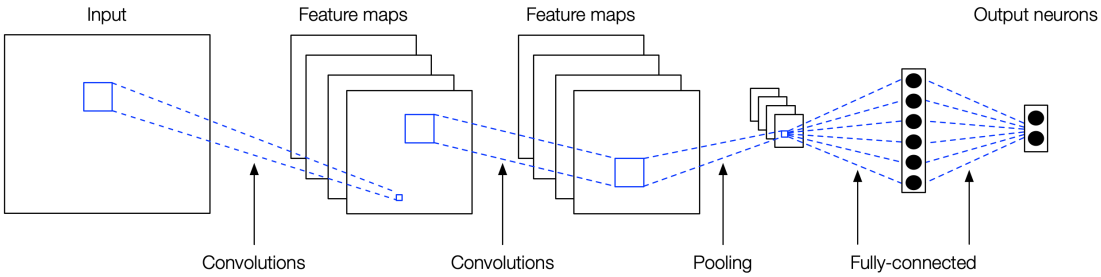

Figure 3.4: A convolutional neural network architecture consisting of two convolutional layers and a subsampling (i.e., pooling) layer with two fully-connected layers. The input consists of a single channel and convolutional layers that output four feature maps each. The last fully-connected layer consists of two neurons and represents network output.

reduction can be used to reduce the number of feature maps while preserving the salient features of the input. Additionally, a one-to-one projection can be created from the input feature maps in order to pool features across channels or to increase the number of feature maps, similarly to a standard pooling layer (e.g., max pooling).

Dilated Convolutions

Dilated convolutional layers use an additional dilation parameter, referred to as dilation rate, which defines the spacing of values in a filter (Yu and Koltun, 2016). It is a special case of a standard convolutional layer, where filters cover a larger area of the input, without increasing the filter size. Dilations can be useful for detecting large scale structures that span across larger areas of the input. The left side of Figure 3.5 demonstrates how a 3×3 filter with dilation rate of 2 can have the same receptive field as a 5×5 filter, while utilising nine values. Hence, a dilation rate of 1 represents a standard convolution operation and only when dilation rate $\geqslant$2, the spacing in the filters is introduced.

Transposed Convolutions

The transposed convolution layer (Zeiler and Fergus, 2014), sometimes referred to as deconvolution, is an inversion of the standard convolutional layer. Transposed convolutions are often used in symmetrical deep learning architectures, where the reconstruction of the original input is desired (e.g., autoencoders, generative adversarial networks). The transposed convolution layer performs an upsampling of the input through zero padding, controlled by the parameters for filter size and stride, and then application of the standard convolution operation. The filters in transposed convolutions are learnable and therefore can be used to approximate examples that resemble the original data. The right side of Figure 3.5 shows a transposed convolution operation that performs upsampling of a zero-padded input.

3.1.3 Training Methods

Training of a neural network refers to the process of updating tunable parameters Θ with regard to the target output y when provided with input data x. These parameters are updated through minimisation of an error that is measured with a loss function L, which calculates the difference between target

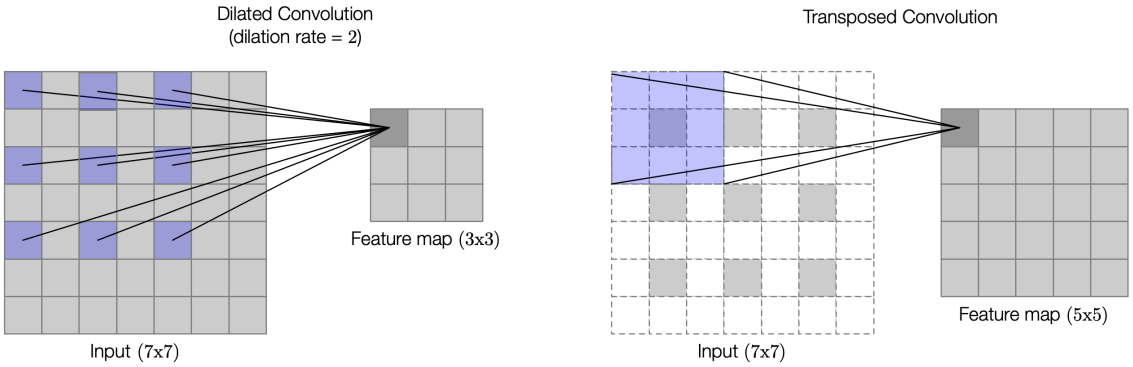

Figure 3.5: In a dilated convolution on the left, the receptive field is larger than in a standard convolution. This increase is achieved through the insertion of spaces, where the spacing is controlled by the dilation rate. In a transposed convolution on the right, zeros (i.e., white units) are inserted between input values and a filter moves over the input (i.e., grey units) to produce output feature map.

values y and the actual output values $\tilde{y}$. The goal of the network is to minimise the single value output computed by the loss function. Mean squared error (MSE) and cross entropy (CE) are two common loss functions used in a variety of regression and classification tasks. MSE is a regression loss that represents the sum of squared differences between y and $\tilde{y}$, and is calculated using:

$$L_{MSE}(\Theta, x, y) = \frac{1}{N_{out}} \sum_{n=1}^{N_{out}} (y_n - \tilde{y}_n)^2, \tag{3.8}$$

where N_{out} represents the number of output neurons and y_n represents the n-th element of the output vector y. MSE is always positive, where larger differences represent a larger relative error. For classification tasks that concern multiple classes, a softmax output layer (see Equation (3.4)) can be used together with a categorical cross-entropy loss:

$$L_{CE}(\Theta, x, y) = -\sum_{n=1}^{N_{out}} y_n log(\tilde{y}_n). \tag{3.9}$$

While MSE penalises larger errors more than small errors, the CE loss penalises larger errors to a greater extent through the use of the natural log function. Binary cross entropy (BCE) loss is often used in combination with the sigmoid output layer (see Equation (3.3)) in binary classification tasks (i.e., logistic regression), and is calculated as follows:

$$L_{BCE}(\Theta, x, y) = -\frac{1}{N_{out}} \sum_{n=1}^{N_{out}} [y_n log(\tilde{y}_n) + (1 - y_n) log(1 - \tilde{y}_n)]. \tag{3.10}$$

The neural network parameters Θ ($\Theta = [W, B]$) are updated using the gradient descent optimisation algorithm. A gradient $\mathcal{G}$ is calculated given the loss and the network parameters as follows:

$$\mathcal{G} = \nabla_{\Theta} L(\Theta, x, y). \tag{3.11}$$

The layered structure of neural networks introduced in Section 3.1.2, facilitates the process of parameter adaptation that computes backward propagation of errors, referred to as backpropagation (Rumelhart et al., 1986). Backpropagation computes partial derivatives by recursively applying the chain rule through all layers in the backward direction of the network. It is used to calculate the gradient of an error measure with an automatic differentiation applied to all derivatives from every activation function. This gradient is then used for updating each individual parameter of the network in incremental updates termed iterations. Gradient descent is weighted with the learning rate η that controls the degree of the update. The updated parameters are calculated by subtracting the gradients from the network parameters as follows:

$$\Theta \Leftarrow \Theta - \eta \cdot \mathcal{G}. \tag{3.12}$$

There exist several ways for the calculation of the gradients with regard to the training data pairs. A parameter update can be computed from all training example pairs (i.e., batch gradient descent), a subset of examples (i.e., mini-batch gradient descent), or a single training example (i.e., stochastic gradient descent). An iteration over the whole training dataset is referred to as an epoch. For example, in a batch gradient descent an epoch consists of one update, whereas mini-batch gradient descent often completes an epoch after many iterations, often hundreds or thousands.

The standard gradient descent algorithm has some limitations when applied to complex problems where it can get stuck in a local minima, instead of the global minima, of the entire domain of the loss function. Another issue can be caused by the learning slowing down the training progress due to either too small or too large updates, thus preventing convergence. There exist several methods that have been proposed to improve the convergence of gradient descent such as momentum and other methods with adaptive learning rate adaptation. Stochastic gradient descent with momentum V (Sutton, 1986) incorporates short-term memory of the past gradients to increase the learning rate and accelerate the updates as follows:

$$V_i = \lambda V_{i-1} + \eta \cdot \mathcal{G}, \tag{3.13}$$

$$\Theta = \Theta - V_i, \tag{3.14}$$

where λ controls the weighting of the previous iteration. Thus, the gradients are increased by adding momentum at each iteration i to overcome strong local minima. Other methods that accelerate convergence with learning rate adaptation include: Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012), RMSprop (Tieleman, Hinton, et al., 2012), and adaptive moment estimation (Adam) (Kingma and Ba, 2014). As an example, the Adam method uses the history of the past gradients with a learning rate adaptation technique that enables the calculation of increased updates on sparse parameters and smaller updates on abundant parameters. The first and second moment estimates s_{i-1} and r_{i-1}, respectively, are calculated as follows:

$$s_i = \rho_1 s_{i-1} + (1 - \rho_1) g_{i-1}, \tag{3.15}$$

$$r_i = \rho_2 r_{i-1} + (1 - \rho_2) g_{i-1} \odot g_{i-1}, \tag{3.16}$$

where ρ_1 and ρ_2 denote the exponential decay rates, and $\odot$ is element-wise product between gradients g. The first and second moment estimates are initialised as zeros. To prevent the decay rates from becoming biased to zero their decaying averages are corrected as follows:

$$\hat{s}_i = \frac{s}{1 - \rho_1}, \tag{3.17}$$

$$\hat{r}_i = \frac{r}{1 - \rho_2}. \tag{3.18}$$

Then the parameter update is computed as:

$$\Theta \Leftarrow \Theta - \eta \frac{\hat{s}_i}{\sqrt{\hat{r}_i} + \delta}, \tag{3.19}$$

where δ is a small constant added for numerical stabilisation (e.g., $\delta = 10^{-8}$). The decay rates ρ_1 and ρ_2 are typically set to values 0.9 and 0.999, respectively. Alternative neural network training methods have been proposed such as limited-memory BFGS algorithm (Liu and Nocedal, 1989), genetic algorithms (Moriarty and Mikkulainen, 1996), and augmented Lagrangian methods (Taylor et al., 2016). The systems implemented in this thesis, unless stated otherwise, are trained with the Adam optimiser.

3.1.4 Regularisation

An important challenge in training of deep learning networks is the prevention of overfitting. Methods that counteract this problem are referred to as regularisers. Overfitting occurs when a trained model fits too closely or exactly to the training data. An overfitted model accurately represents the training data but is not effective at generalising to unseen data. On the contrary, if a model does not fit the training data and does not generalise well to unseen data, then it is underfitted. Overffiting and underfitting can be monitored by splitting the total data into training, validation and testing sets. The validation set is used to monitor the performance of the model on isolated data but is not used to upgrade gradients $\mathcal{G}$ during training. When the validation loss begins to increase then the model is overfitting and the training is stopped. This process is illustrated in Figure 3.6, where the optimal capacity of the model lies between the underfitting and overfitting zones.

A number of methods can be implemented to prevent complex models from overfitting during training. A straightforward regularisation approach is to lower the complexity of the network architecture by reducing the number of trainable parameters. This is time consuming and often not computationally feasible, hence a common approach is to utilise models with more capacity than necessary, and counteract overfitting with other regularisation techniques presented below.

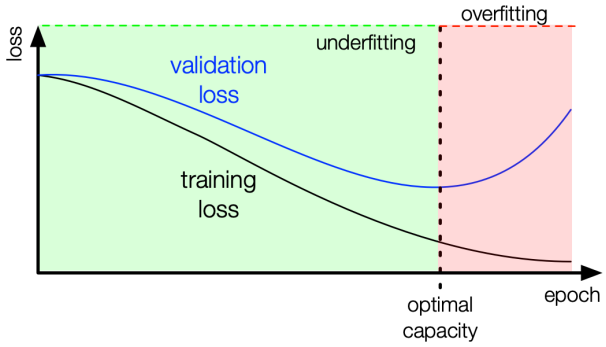

Figure 3.6: Optimal capacity of a network illustrated with training and validation losses. When the validation loss begins to increase then the model is overfitting and the training is stopped. The underfitting and overfitting zones are represented with green and red, respectively.

ℓ_1 and ℓ_2 norm Regularisation

ℓ_1 and ℓ_2 norms are regularisation techniques that prevent overfitting by modifying the loss function to penalise large absolute weight values in the network as follows:

$$\ell_1 = \ell_1 + \lambda \cdot \sum |w|, \tag{3.20}$$

$$\ell_2 = \ell_2 + \lambda \cdot \sum w^2, \tag{3.21}$$

where λ controls the weighting of each term. The ℓ_1 norm forces the parameters towards 0, while the ℓ_2 norm forces the weights w to be small, but not 0. Both norms limit the modelling capabilities of the network and thus combat overfitting.

Batch Normalisation

Data normalisation is a common technique that prevents imbalances of different features during training. Input features and features in the subsequent hidden layers are often normalised; however, normalisation of all features does not guarantee that data within individual mini-batches preserves the normalisation criteria. Batch normalisation (Ioffe and Szegedy, 2015), is a technique that normalises activations between each layer of the network during mini-batch training and results in a mean of 0 and a variance of 1, while adding two learnable scaling and shifting parameters. Batch normalisation accelerates training and has some regularisation properties. First, the mean $\mu_{\mathcal{F}}$ and the variance $\sigma^2_{\mathcal{F}}$ of features $\mathcal{F} = \{x_1 ... x_m\}$ is calculated over the mini-batch of size m as follows:

$$\mu_{\mathcal{F}} = \frac{1}{m} \sum_{i=1}^{m} x_i, \tag{3.22}$$

$$\sigma^2_{\mathcal{F}} = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_{\mathcal{F}})^2. \tag{3.23}$$

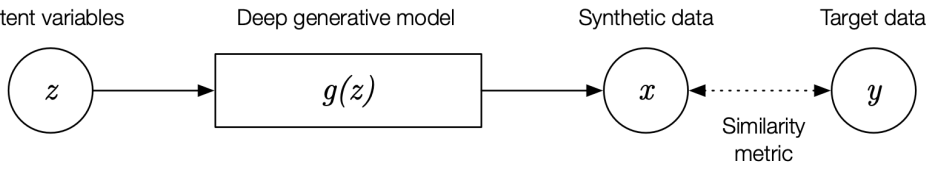

Figure 3.7: A deep generative model g is trained to map samples from a simple distribution z to the more complex distribution $g(z)$ based on a comparison with the target data distribution y. An objective function is used during training to quantify the discrepancy between the generated (i.e., synthetic) x and the target examples y.

The batch normalising transform is implemented by subtracting the mean and then dividing by the variance of each feature in the current mini-batch before scaling and shifting with the learnable parameters $\lambda_{\mathcal{F}}$ and $\beta_{\mathcal{F}}$:

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{F}}}{\sqrt{\sigma^2_{\mathcal{F}} + \epsilon}}, \tag{3.24}$$

$$y_i = \lambda_{\mathcal{F}} \hat{x}_i + \beta_{\mathcal{F}}, \tag{3.25}$$

where ϵ is a small constant to avoid dividing by 0 and y_i is the mini-batch normalised output. During inference, the process of running unseen data through the trained model, the mean and variance calculated from the full training set is used. This leads to deterministic predictions output from the model, which is a useful property.

Dropout

For each training iteration, dropout disables the chosen connections between two layers in a network by setting them to 0. The percentage of the disabled neurons is controlled by an additional user-defined hyperparameter. Dropout improves the generalisability and prevents overfitting by forcing the system to identify additional paths and relationships throughout the network (Srivastava et al., 2014).

Data Augmentation

Data augmentation counteracts overfitting and improves the generalisability of a deep network by increasing the training set with a larger variety of examples. Increasing the size of the training set is especially helpful for small datasets with under-represented data points. In the field of image recognition, the augmentation techniques often change properties such as brightness, hue, contrast and rotation of the images in the training set. Augmentation techniques used for music signals are often inspired by popular transformations used in music production such as pitch-shifting and time-stretching (see Section 2.3.2). Other techniques include volume changes, addition of noise and sound overlapping.

3.2 Deep Generative Models

Deep generative models (DGMs) are DNNs trained to approximate complex and high-dimensional probability distributions. A trained DGM can be used to estimate the likelihood of an observation

and to generate new examples from the underlying distribution. Many deep generative models are based on differentiable generator networks. This class of models includes networks such as variational autoencoders (VAEs), which combine the generator network with an inference network (see Section 3.2.3); generative adversarial networks (GANs), which combine the generator with a discriminator network (see Section 3.2.4); and methods that train generators in isolation (see Section 3.2.1).

Generative models approximate a probability distribution of the target data p_d, provided with a finite set of samples from this distribution. During training the most plausible network parameters are found with regard to a distance between the model distribution and the true data distribution. A generative model $p_\theta(X)$ is trained to closely match an empirical data distribution $p_d(X)$ using training examples X. This optimisation task can be computed as follows:

$$\min_{\theta \in \mathcal{M}} \mathcal{D}(p_d, p_\theta), \tag{3.26}$$

where θ are parameters of the network, $\mathcal{M}$ denotes the chosen generative model type (e.g., VAE), and $\mathcal{D}$ denotes a divergence function that measures the difference between the two distributions. Measuring differences between probability distributions is challenging and differs from the approach used in typical loss functions such as MSE from Equation (3.8), used in supervised learning. In the supervised learning setting, the input values and the output values of the network are typically defined on real-valued vectors with paired targets. In the case of MSE loss, the elements of the network output are assumed to follow independent normal distributions. In generative networks, there may be no pairing between the inputs and outputs, since the training objective is to follow a specific distribution, where either one or both, X and θ can be random. These two types of modelling can be referred to as discriminative and generative; the former learns the distribution of y conditioned on individual samples of x in the form of $p(y|x)$; the latter maps the distribution of x to the distribution of y in the form of $p(x, y)$. Generative models can also be referred to as latent variable models, where an assumption is made that a set of latent (i.e., hidden) variables z from a simple distribution can be used to generate samples x from a complex data distribution. Multiple divergence functions have been used for various applications and additional details about the most often used functions are presented below.

Many popular divergences are special cases of f-divergence for a particular function f. One popular instance of f-divergences is the Kullback-Leibler (KL) divergence used in variational inference. Given two continuous probability distributions p_x and p_y that possess, respectively, absolutely continuous density functions $p_x(z)$ and $p_y(z)$, the f-divergence is defined as:

$$D_f(p_x||p_y) = \mathbb{E}_{z \sim p_y}\left[f\left(\frac{p_x(z)}{p_y(z)}\right)\right], \tag{3.27}$$

where f is the generator function satisfying $f(1) = 0$. Intuitively, f-divergence computes the average of the odds ratio given by p_x/p_y weighted by the chosen function f. Nowozin et al. (2016) present several choices for f. For example, when $f(t) = t \log t$, the Equation (3.27) represents the KL-divergence.

The integral probability metrics (IPMs) are another class of difference measures on probabilities that include: total variation distance, Wasserstein distance, and maximum mean discrepancy. Given two real-valued bounded distributions p_x and p_y, the IPM is defined using the supremum (least upper bound abbr. sup) over a function class F as follows:

$$IPM(F, p_x, p_y) = \sup_{f \in F} |\mathbb{E}_{x \sim p_x}[f(x)] - \mathbb{E}_{y \sim p_y}[f(y)]|. \tag{3.28}$$

Intuitively, the statistics extracted by f can be used to differentiate the two distributions. Various distance metrics can be obtained through the appropriate choice of F given that it is also computationally tractable (Sriperumbudur et al., 2009). Three popular function classes of F are F_{tv}, $F_{\mathcal{W}}$ and F_K:

$$F_{tv} = \{f : ||f||_\infty \leqslant I\}, \tag{3.29}$$

$$F_{\mathcal{W}} = \{f : ||f||_L \leqslant I\}, \tag{3.30}$$

$$F_K = \{f : ||f||_{\mathcal{H}} \leqslant I\}, \tag{3.31}$$

where the constant I (by default $I = 1$) can be any number used to scale each metric by a preset amount. F_{tv} results in computation of the total variation distance with bounds between $[-1, 1]$. $F_{\mathcal{W}}$ is a K-Lipschitz continuous function used in computation of the Wasserstein distance. F_K results in computation of maximum mean discrepancy functions that reside within the unit sphere of a reproducing kernel Hilbert space (RKHS) defined by a well-behaved kernel function where calculations can be carried out. Many statistical distances are not proper metrics (e.g., f-divergences); however, IPMs do not require the computation of p_x or p_y, but only an expectation, which can be approximated only through the use of samples. Although the often intractable detection of supremum over a class of functions results in IPMs being more computationally expensive, one computationally tractable example is maximum mean discrepancy.

Maximum mean discrepancy (MMD) is defined when the function class F in Equation (3.28) is the unit sphere in RKHS $\mathcal{H}$ with kernel $k(\mathcal{X}, \mathcal{X}) \to \mathcal{R}$, where $\mathcal{X}$ represents a nonempty compact set. Since $\mathcal{H}$ is an RKHS, there exists a feature mapping, $\phi(x) : \mathcal{X} \to \mathcal{H}$ for each positive definite kernel function such that $k(x, y) := \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$ without the need to compute ϕ explicitly. The kernel function can be tractably computed and applied to any learning algorithm using a kernel trick if it can be entirely expressed in terms of a dot product $\langle x, y \rangle$ (Muandet et al., 2016). The positive definiteness property of the kernel function ensures that the dot product will exist within a high-dimensional feature space $\mathcal{H}$. The standard form of this mapping is represented as $\phi(x) = k(x, \cdot)$ (Steinwart and Christmann, 2008). The notation $k(x, \cdot)$ indicates the kernel has one argument fixed at x and the second is free.

The idea of feature maps is extended to the space of probability distributions through the definition of each distribution p as a mean function such that the expectation of the feature map $\phi(p)$ is defined by a kernel mean embedding $\mu_p := \mathbb{E}_{x\sim p}k(x,\cdot)$.

An important property of MMD is attaining the supremum through the calculation of the witness function f^* that maximises the mean discrepancy in distributions, defined by Gretton et al. (2012) as:

$$f^*(\cdot) \propto \mathbb{E}_{x\sim p_x}k(x,\cdot) - \mathbb{E}_{y\sim p_y}k(y,\cdot). \tag{3.32}$$

Intuitively, the function f^* witnesses the MMD by assigning high values on samples from p_x and low values on samples from p_y. f^* is near zero for samples from regions where p_x and p_y have similar densities. The squared MMD can be calculated in a closed form by inserting Equation (3.32) into Equation (3.28) as follows:

$$MMD^2(k,p_x,p_y) = \mathbb{E}_{x,x'\sim p_x,p_x}k(x,x') + \mathbb{E}_{y,y\sim p_y,p_y}k(y,y') - 2\mathbb{E}_{x,y\sim p_x,p_y}k(x,y). \tag{3.33}$$

A commonly used kernel function is the Gaussian radial basis kernel with a tunable parameter Σ:

$$k(x,x') = \exp\left(-\frac{||x-x'||^2}{2\Sigma^2}\right). \tag{3.34}$$

Computation of MMD is tractable for a class of kernels, referred to as characteristic kernels, where the MMD is zero if and only if $p_x = p_y$. In reality, this will often not be possible as only a finite number of samples is observed. The choice of an appropriate kernel function varies for different data distributions and may cause issues for MMD when the sample size is small (Muandet et al., 2016). Additionally, the kernel function can be parameterised and then learned for a target task with a deep generative network. In the deep learning setting, the network acts as a feature extractor that maps the input space to a lower dimensional space, for which a simple kernel function can be used (Chen, 2017). For example, the Σ parameter in the Gaussian radial basis kernel (3.34) can be learned by the weights of a deep network without the need to be specified manually.

3.2.1 Generative Moment Matching Networks

Generative moment matching networks (GMMN) (Dziugaite et al., 2015; Li et al., 2015) are an example of differentiable generator networks that are not required to be paired with any other network. For example, GMMNs do not require an inference network that is used in variational autoencoders and a discriminator network used in generative adversarial networks, which are described in the subsequent sections. The concept of moment matching refers to the way the generator is trained to match the statistics of the generated data samples, as closely as possible, with the statistics of the examples in the

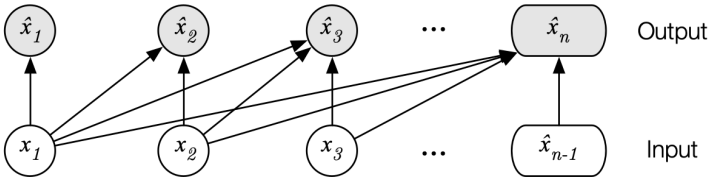

Figure 3.8: Autoregressive model prediction of an output based on the past inputs.

training set. GMMNs can be trained by minimising the MMD objective as follows:

$$\min_G \mathbb{E}_{x,x' \sim p_x, p_x}[k(x, x')] + \mathbb{E}_{z,z' \sim p_z, p_z}[k(G(z), G(z))] - 2\mathbb{E}_{x,z \sim p_x, p_z}[k(x, G(z))], \tag{3.35}$$

where G denotes the generator network. The data distribution and the prior distribution imposed on the latent variables z are denoted by p_x and p_z, respectively.

3.2.2 Autoregressive Networks

Autoregressive (AR) networks have been designed with a focus on modelling sequential data (Bengio et al., 2000; Uria et al., 2014). AR networks define an explicit and computationally tractable model based on the chain rule of probability, where the probability of an n-dimensional variable $x = x_1, ..., x_n$ is factorised as follows:

$$p(x) = p(x_1, ..., x_n) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1}). \tag{3.36}$$

The model predicts the next data sample by multiplying all the probabilities of the samples that came before it. AR networks can be trained to maximise the log-likelihood of the data with regard to the parameters of the network to model $p(x_i | x_{1:i-1})$ by minimising the negative log-likelihood as follows:

$$-\ln p(x) = -\sum_{i}^{n} \ln p(x_i | x_1, ..., x_{1:i-1}). \tag{3.37}$$

Autoregressive models operate on sequential data that is often high-dimensional. A 1-second audio signal, for example, which is sampled at 16 kHz with 8 bits representing each audio sample contains 256^{16000} possible sequences. For these reasons, the parameter space can become large and thus slow down the network during training and inference time. Several techniques have been proposed to improve speed, as well as the receptive field, and memory capacity of AR models through the use of architectures such as RNNs with long short-term memory (Hochreiter and Schmidhuber, 1997), and CNNs with dilated causal convolutions (Oord et al., 2016b). Additionally, the performance of an AR network depends on the modality of the data as it must be decomposable into a fixed order. While the ordering is clear in modelling of raw audio, the performance can suffer when modelling other data such as images.

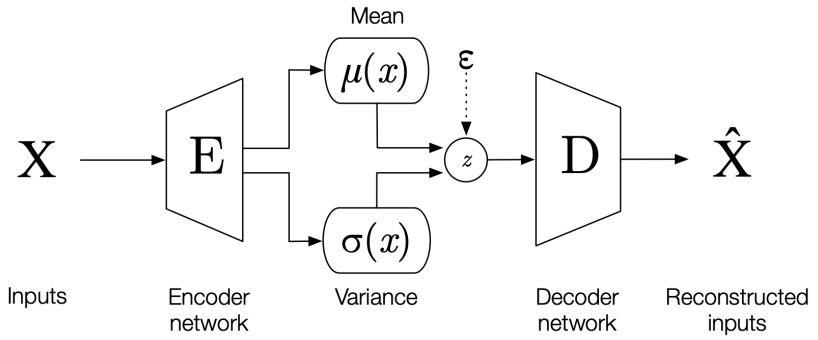

Figure 3.9: Variational autoencoder model with input examples $X = (x_1, x_2, ...)$ processed by an encoder network and mapped into a Gaussian distributed latent space $q(z|X)$ parameterised by μ and σ. The decoder network samples the latent space z with $p(X|z)$ to output the synthetic reconstructions $\hat{X}$.

3.2.3 Variational Autoencoders

Variational autoencoder (VAE) networks (Kingma and Welling, 2013) belong to a family of directed models referred to as latent variable models that can be trained using gradient-based methods. A standard autoencoder model learns a compressed representation of the input examples, whereas a variational autoencoder (see Figure 3.9) learns parameters of a probability distribution in the latent space z that can be later used for sampling and mixing of data. Let $p_\theta(x|z)$ be a latent based model parameterised by θ with $p_\theta(z)$ prior and $p_\theta(z|x)$ posterior. An inference model can be derived from an application of Bayes' rule:

$$p_\theta(x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(z|x)}. \tag{3.38}$$

Unfortunately, optimisation of $p_\theta(x)$ directly is not computationally tractable since $p_\theta(z|x)$ is not known and thus, maximum likelihood can not be used to optimise the model $p_\theta(x|z)$ due to the integral in $p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)dz$. Instead, Kingma and Welling (2013) propose to use KL-divergence to approximate the posterior $p_\theta(z|x)$ through the definition of a new distribution $q_\phi(z|x)$ that transforms it as close as possible to $p_\theta(z|x)$ by minimising the KL-divergence such that $q_\phi(z|x) = \arg\min_q D_{KL}(q_\phi(z|x)||p_\theta(z|x))$. Equation (3.38) can be combined with KL-divergence and rearranged to estimate an upper-bound of the negative log-likelihood $\log p_\theta(x)$ as follows:

$$-\log p_\theta(x) \leqslant \mathbb{E}_{z\sim q_\phi(z|x)}[-\log p_\theta(x|z)] + D_{KL}(q_\phi(z|x)||p_\theta(z)) \tag{3.39}$$

$$\equiv -\mathcal{L}(\theta, \phi, x). \tag{3.40}$$

This bound is referred to as the negative evidence lower-bound (ELBO) (Jordan et al., 1999) and can be denoted by $-\mathcal{L}(\theta, \phi, x)$.[12] To optimise the negative ELBO objective during training with respect to θ and ϕ, gradients are backpropagated through the stochastic process that generates new samples from $\tilde{z} \sim q_\phi(z|x)$. To that end a differentiable function $g_\phi(\epsilon, x)$ of a random variable $\epsilon \sim p(\epsilon)$ is used in a

[12]Full derivation of (3.39) is presented in tutorials by Doersch (2016) and Yu (2020).

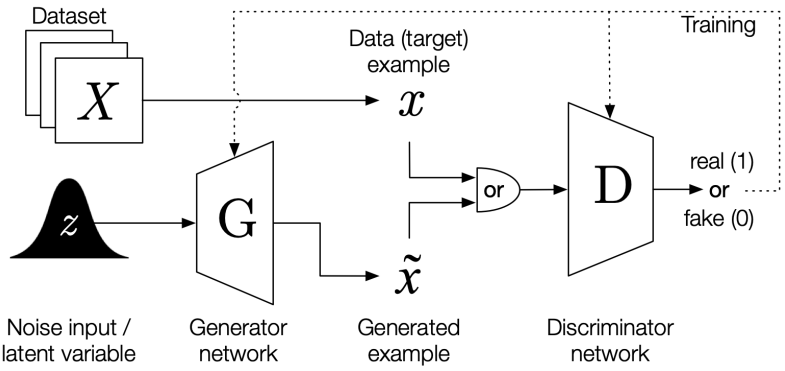

Figure 3.10: Generative adversarial network.

reparametrisation of $\tilde{z}$, referred to as a reparametrisation trick (Kingma and Welling, 2013). Figure 3.9 illustrates a VAE model with a normally distributed prior, where ϵ is sampled from $\mathcal{N}(0, I)$. In practice $p_\theta(x|z)$ and $q_\phi(z|x)$ are parameterised with decoder (i.e., generator) and encoder networks, respectively.

The first term on the right side of Equation (3.39) represents the reconstruction loss that affects the decoder network trained to best reconstruct x based on z. The encoder network is stimulated through the reconstruction loss to increase the signal-to-noise ratio in z by decreasing $||\sigma^2_\phi(x)||$ and increasing $||\mu_\phi(x)||$. The second term in (3.39) encourages this behaviour through regularisation provided by the KL-divergence, where $p_\theta(z)$ is typically an isotropic Gaussian distribution.

3.2.4 Generative Adversarial Networks

Generative adversarial networks (GANs) (Goodfellow et al., 2014) consist of two competing neural networks termed generator and discriminator as illustrated in Figure 3.10. The discriminator $D: \mathbb{R}^n \rightarrow [0, 1]$ estimates the probability that a sample comes from a data distribution $x \sim p_d(x)$. The generator $G: \mathbb{R}^m \rightarrow \mathbb{R}^n$ uses a latent variable $z \sim p_z(z)$ to capture p_d with an aim of tricking the discriminator network into classifying its samples as real examples from a training set. New samples are generated such that $\hat{x} = G(z; \theta_g)$, where θ_g denotes parameters of the generator. The discriminator is trained to classify input samples as either training examples (real) or generated examples (fake) such that $o = D(x; \theta_d)$, where θ_d denotes parameters of the discriminator and o is the classification output. This training process is referred to as adversarial training and can be interpreted as a minimax game between both networks that optimise the value function $V(G, D)$ as follows:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_d(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]. \tag{3.41}$$

Figure 3.11 illustrates the distributions of the generator $p_g(x)$ compared to the data distribution $p_d(x)$ for a one-dimension example of X during adversarial training. Figure 3.11 (d) shows an optimally trained discriminator D^* with regard to the generator in a scenario when it can only guess such that $D^*_G = p_d(x)/(p_d(x) + p_g(x))$. Additionally, if D distinguishes between the generated and real samples

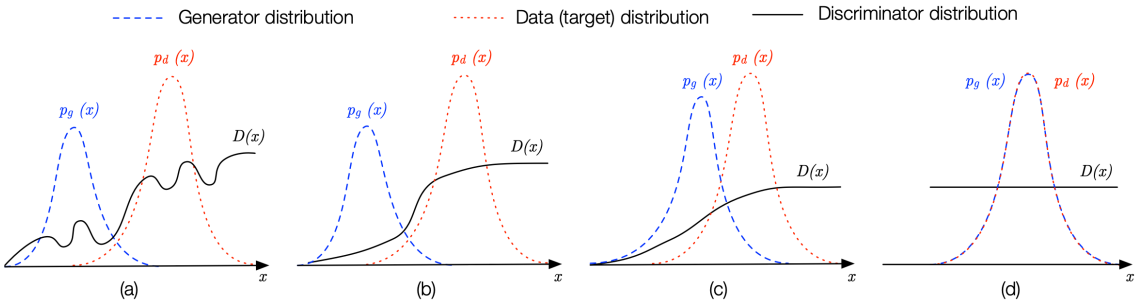

Figure 3.11: Probability distributions for generator $p_g(x)$, target data $p_d(x)$ and discriminator $D(x)$ distributions. Plot (a) shows an untrained discriminator; in (b) the optimal $D(x)$ is found; in (c) $p_g(x)$ becomes more similar to $p_d(x)$; and in (d) $p_g(x)$ produces data samples that are indistinguishable from the real target data.

perfectly, then no information is passed to the generator network. The training can stagnate due to the vanishing gradient problem if the outputs of D are too close to either 0 or 1. This creates instability during training and can be counteracted with a careful tuning of both network architectures. Several stabilisation techniques for GAN training have been proposed by the authors in Arjovsky and Bottou (2017) as equilibrium between D and G, referred to as Nash equilibrium, is difficult to achieve (Salimans et al., 2016). One major improvement proposed by Arjovsky et al. (2017) was the use of the Wasserstein distance instead of KL-divergence. Any f-divergence can be used for GAN training (Nowozin et al., 2016); however, Wasserstein distance was shown to offer an improved stability over other metrics. Using the Wasserstein distance during training is analogous to minimising an IPM with a set of F all 1-Lipschitz functions. The discriminator is then used to approximate the supremum over F. If $\forall f \in F \Rightarrow -f \in F$, then the absolute sign in Equation (3.28) can be removed resulting in the following objective function:

$$\min_{G} \max_{D:||D||_L \leqslant 1} = \mathbb{E}_{x \sim p_d}[D(x)] - \mathbb{E}_{z \sim p_z}[D(G(z))]. \tag{3.42}$$

Intuitively, a Wasserstein GAN (WGAN) does not require classification of examples into real and generated, but rather learns a function f_K (see Equation (3.31)) that helps to differentiate between the two distributions p_d and p_g. In practice the 1-Lipschitz continuity of the function f_K is enforced through weight clipping of D to a small interval (e.g., ± 0.01). This technique however can also contribute to the problem of vanishing gradients when the chosen clipping window is too large. Gulrajani et al. (2017) propose an improvement to circumvent this problem in form of a gradient penalty (GP) loss that results in a deep generative model termed WGAN-GP.

3.2.5 Adversarial Autoencoders

While similar in design to VAEs (Kingma and Welling, 2013), adversarial autoencoders (AAE) appropriate the additional discriminator network from GANs, which aims to distinguish between real and fake (i.e., synthesised) samples. Figure 3.12 illustrates an AAE model. Makhzani et al. (2015) observed that VAEs are largely limited by the Gaussian prior, and relaxed this constraint by allowing $p(z)$ to be any distribution by replacing the KL-divergence with an adversarial loss imposed on the output of the encoder.

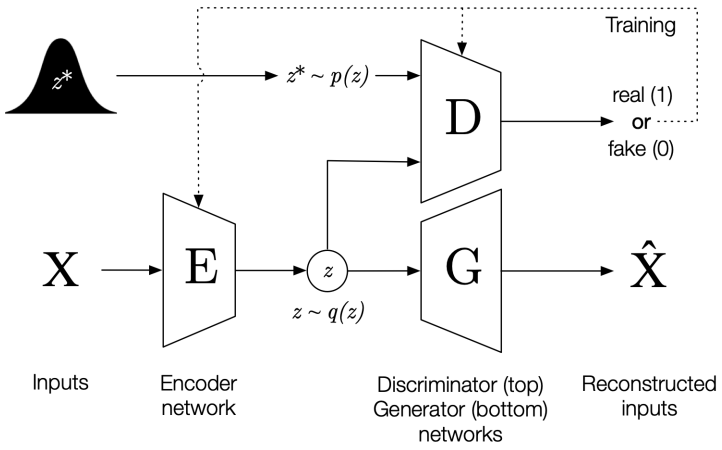

Figure 3.12: Adversarial autoencoder. Input data X is mapped onto a latent variable $z \sim q(z)$. Encoder E tries to trick discriminator D with artificially generated latent samples and generator G outputs $\tilde{x}$. A Gaussian prior distribution $z^* \sim p(z)$ allows the model to juxtapose similar inputs in the latent space.

Thus, the AAE framework makes it possible to leverage any prior knowledge that may be specific to the studied application. The encoder $q_\phi(z|x)$ in AAEs defines an aggregated posterior distribution $q_\phi(z)$ on the latent variables z as follows:

$$q_\phi(z) = \int_x q_\phi(z|x)p_d(x)dx. \tag{3.43}$$

In practice, the decoder $p_\theta(x|z)$ (i.e., generator G) and the encoder E are parameterised with neural networks. The uniform distribution is imposed on z through D_z (i.e., the discriminator on z). The distribution of the training data is denoted by $p_d(x)$, and $q_\phi(z|x)$ is the distribution of the latent variable z. The random sampling process from $p(z)$ is represented by $z^* \sim p(z)$ and the adversarial component of an AAE can be interpreted as a minimax game between the encoder E and discriminator D_z that optimise the value function $V(E, D_z)$ as follows:

$$\min_E \max_{D_z} V(E, D_z) = \mathbb{E}_{z* \sim p(z)}[\log D_z(z^*)] + \mathbb{E}_{x \sim p_d(x)}[\log(1 - D_z(E(x)))]. \tag{3.44}$$

The parameters of the autoencoder are optimised by the reconstruction error, while the adversarial network guides the encoder to match the imposed prior. Thus, the encoder plays the role of the generator during the adversarial part of training, while the discriminator represents the adversarial network defined in Equation (3.41). After training, decoder G acts as a generative model that maps the imposed prior to the data distribution. Training of an AAE is performed in two phases: (1) the reconstruction phase and (2) the regularisation phase. In the reconstruction phase the reconstruction error of E and G is minimised together (e.g., using BCE loss (3.10)) and in the regularisation phase, the parameters of the discriminator D_z are updated by minimising $\mathcal{L}_{D_z} = -V(E, D_z)$ (i.e., to distinguish true samples generated by the prior from the generated codes processed by the autoencoder). The adversarial network then updates the parameters of the encoder to confuse the discriminator.

3.3 Transformation Modes of Deep Generative Models

DGMs generate new examples based on learned parameters of deep neural networks and thus extend techniques used in rhythmic transformation systems based solely on signal processing techniques, such as time-stretching and resequencing (i.e., reordering) discussed in Chapter 2. In the process of controlled manipulation, the latent space is analysed to understand how it correlates with audio characteristics. Certain directions or dimensions within this space correspond to specific attributes, such as rhythmic patterns or timbre. Modifying a latent vector—a compact representation in a reduced-dimensional latent space—in a specific direction and then decoding it results in the generated audio reflecting anticipated changes. For example, if a particular direction in the latent space is tied to a change in a rhythmic pattern, adjusting a latent vector in that direction produces audio with drums rearranged to create a new pattern. Similarly, adjustments corresponding to drum timbre can change the sound from one type of drum to another. The effectiveness of these associations often depends on factors like the model's architecture, the quality of training data, and the complexity of the audio data.

Figure 3.13 presents an analogous transformation to that illustrated in Figure 2.11. Deep generative model in Figure 3.13 outputs different combinations of rhythmic and timbral transformations of source and target recordings, as well as transformations generated from the learned latent space of the model. For example, transformation with time-stretching and with drum segment reordering shown in Figure 2.11 can be represented by a transformation with a DGM shown in Figure 3.13 (d), where the output contains timbre characteristics of the source and the rhythmic pattern of the target. In both transformations, the rhythmic pattern and the drum characteristics have been affected either by a digital signal processing algorithm for time-stretching (e.g., phase vocoder) or by an internal representation of a deep generative model (e.g., GAN). This internal representation—also referred to as the latent space—of a DGM can also be used for other transformations such as generation of new drum characteristics (e.g., Figure 3.13 (e)) with new rhythmic patterns (e.g., Figure 3.13 (g)) or generation of a new pattern with the chosen drum characteristics of audio inputs (e.g., Figure 3.13 (h)). A controlled generation of other drum characteristics and other rhythmic patterns can be performed through manipulation of the latent space parameters of a trained model. Additionally, a DGM can be used for mixing of different drum sounds together in a form of a neural mashup transformation illustrated in Figure 3.13 (f) and introduced in Section 2.3.3.

The degree of control over different transformations depends on several factors such as model architecture (see Section 3.2), training strategy (see Section 3.1.3) and model conditioning. The internal architecture of a DGM cannot be used for manipulation of the output during the generation phase without an additional retraining or reoptimisation of the model. This can be computationally expensive and time-consuming but can be counteracted by introducing control of the output through the training phase of the model. Examples of this include specialised *content* and *style* loss functions used in neural style transfer methods for images (Gatys et al., 2015) and audio (Ulyanov and Lebedev, 2016). An

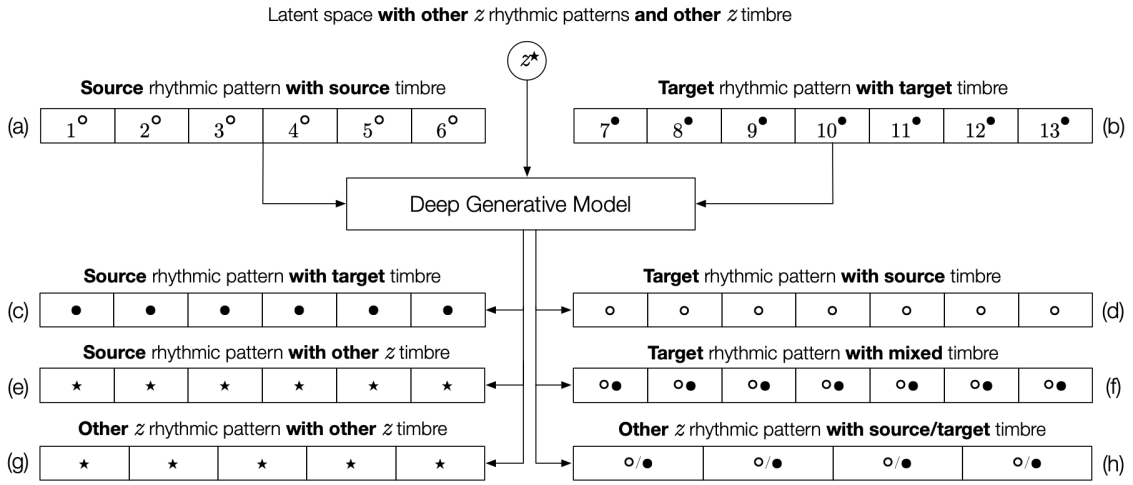

Figure 3.13: Rhythmic transformation with a deep generative model: (a) Source rhythmic pattern, (b) target rhythmic pattern, (c) source pattern with target • drum timbres, (d) target pattern with source ○ drum timbres, (e) source pattern with other ⋆ drum timbres, (f) target pattern with mixed ○• timbres, (g) other rhythmic pattern with other ⋆ drum timbres from latent space z, and (h) other rhythmic pattern with source ○ or target • drum timbres.

alternative strategy involves conditioning of a DGM with an auxiliary input (e.g., drum type or rhythmic pattern style) input into the model during training. This information can provide a more intuitive control over the output generation that affects only a selected portion of the learned data space represented as z in Figures 3.13 and 3.14. This strategy has been implemented in a variety of conditional DGMs (Bitton et al., 2019; Mehri et al., 2016; Oord et al., 2016a).

The abilities of deep generative models for transformation of rhythmic and timbral characteristics of drum recordings can be explored using the following three transformation modes: drum translation, audio style transfer and latent space manipulation. These modes utilise DGMs and affect rhythmic and timbral characteristics in different ways but use the same concepts introduced in Figures 2.11 and 3.13. Rhythmic patterns can be primarily modified in two ways: (1) through the modification of their temporal positions (e.g., time-stretching), and (2) through the transformation of different drums to other drum types within a whole recording (e.g., segment reordering). The timbral characteristics can be modified through transfer of timbre from one recording to another or generation of new timbre from the representation of a trained model. Modes of transformation used in systems proposed in this thesis are illustrated using a simplified example with single drum inputs in Figure 3.14 and a more detailed description of each mode is provided in the following sections.

3.3.1 Drum Translation

Deep generative models that transform (i.e., translate) a drum recording of some type (also referred to as domain) to another target type can be referred to as drum translation systems. More broadly, neural translation methods were first introduced using deep learning in the field of image processing (Isola et al., 2017; Zhu et al., 2017) and are often based on deep autoencoder architectures (see Section 3.2). Figure 3.14 shows a drum translation example, where an input kick recording is translated to the

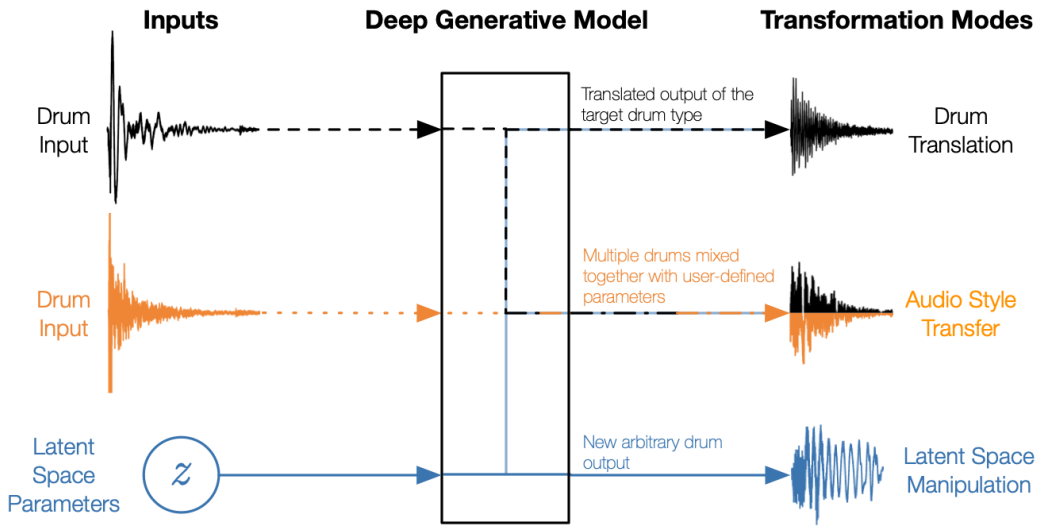

Figure 3.14: Modes of transformation. In drum translation an input drum sound is transformed (i.e., translated) to another target drum (black dashed pathway). In audio style transfer two or more inputs are mixed together in a form of a neural mashup with user defined parameters (orange dotted and dashed pathway). In latent space manipulation a new arbitrary drum is generated from the learned representation of a model (blue solid line pathway).

target snare drum type. A drum translation method applied to a full drum performance can modify the rhythmic pattern of the input analogously to the rhythmic transformation with segment reordering shown in Figure 2.11 and resembles the process of redrumming (see Section 2.3.3) in music production.

3.3.2 Audio Style Transfer

Deep generative models that mix and mash rhythmic and timbral characteristics of two or more audio inputs can be categorised as approaches for audio style transfer that resemble functionalities of automatic mashup systems described in Section 2.3.3. Audio style transfer (AST) was first attempted in Foote et al. (2016) and Ulyanov and Lebedev (2016), which directly extended an algorithm proposed for images in Gatys et al. (2015). In AST, a new output is synthesised by minimising the *content* loss with respect to the *content*-contributing audio input and the *style* loss with respect to one or more audio examples of a given *style*. The *content* loss is based on comparing the network activations of features derived from an audio spectrogram. The *style* loss matches the statistics of the Gram matrix (i.e., inner product between neural feature maps) activations in the higher levels of the network. While the original formulation of image style transfer preserves the composition of the input image by affecting its texture, AST approaches can modify both the timbral characteristics of the input audio as well as longer temporal scales of rhythmic patterns (Barry and Kim, 2018; Tomczak et al., 2018). A type of neural mashup transformation that uses AST approach to mix two drum inputs with user-defined parameters is illustrated in Figure 3.14.

3.3.3 Latent Space Manipulation

Manipulation of characteristics such as pitch, timbre and rhythmic patterns with deep generative models can be performed through the exploration of the latent space—also referred to as latent codes or vectors. Blue pathways in Figure 3.14 illustrate that this type of transformation does not require audio inputs but it can utilise them for the purposes of interpolation transformations. Interpolation between two points in the latent space is a typical manipulation technique that can be applied to transform between drum types and rhythmic patterns. Interpolation can be performed without audio inputs using two random points in the latent space or can resemble any transformation shown in Figure 3.13 (e.g., where source rhythmic pattern is transformed to the target in 3.13 (d)). Latent space manipulation is also the primary mode for rhythmic transformation of drum recordings presented in Chapter 5 and 6 (Tomczak et al., 2019, 2020).

3.4 Chapter Summary

One of the main contributions of this dissertation is the development of automated techniques for transformation of rhythmic patterns of percussion instruments using deep generative models, and an evaluation of the extent of how the patterns have been modified. This chapter has presented the deep learning fundamentals together with deep generative models for neural drum synthesis implemented in the next chapters. These core principles are essential to understand how different DGMs can be utilised for the task of neural drum synthesis and rhythmic transformation. Three modes of rhythmic and timbral transformation of drum recordings proposed in this chapter are drum translation, audio style transfer and latent space manipulation. In addition to employing deep learning fundamentals, Chapters 4–6 incorporate these modes of transformation to develop new contributions which can also be beneficial for a broader range of neural drum synthesis tasks. The following chapter utilises audio style transfer in a rhythmically constrained transformation which overcomes some of the limitations of the original style transfer algorithm making the transformation computationally feasible on drum recordings, while generating rhythmically coherent drum patterns at longer timescales.

Chapter 4

Audio Style Transfer with Rhythmic Constraints

Chapter 2 presented challenges and tasks related to rhythmic description and transformation of drum recordings. Chapter 3 provided an overview of deep learning and deep generative models (DGM) for neural audio synthesis and rhythmic-timbral transformation. In this chapter, a rhythmically constrained audio style transfer (AST) is proposed for the modification of rhythmic and timbral characteristics of two or more audio signals.

Audio synthesis is the task of guided generation of sound given user-defined parameters. In this thesis, two audio synthesis tasks are considered in the context of rhythmic transformation of drum recordings with DGMs. The first task regards learning of an approximate model for the generation of drums and rhythmic patterns, given a large dataset of examples (e.g., individual drum samples and recordings with full drum patterns). The second task regards learning a generative model for the transfer of different music characteristics provided with only a single audio input. The former task can be performed on different types of audio inputs with a trained DGM. The latter requires more assumptions about the input, such that the output transformation has consistent style characteristics of one or more audio examples. This chapter presents an extension of the second task through a rhythmically constrained audio style transfer technique for automatic mixing and mashing of two audio inputs.

In this transformation the rhythmic and timbral features of two input signals are combined together through the use of an audio style transfer process that transforms the musical inputs so that they adhere to a larger metrical structure of the chosen input. This is accomplished by finding beat boundaries of both inputs and performing the transformation on beat-length audio segments.

The main contents of this chapter can be summarised as follows. In Section 4.1, the rhythmic segmentation necessary for preservation of the larger metrical structures during the transformation is introduced along with the architecture of the used deep learning network. In order for the system to perform a mashup between two signals, the audio style transfer formulation introduced in Section 2.3.3

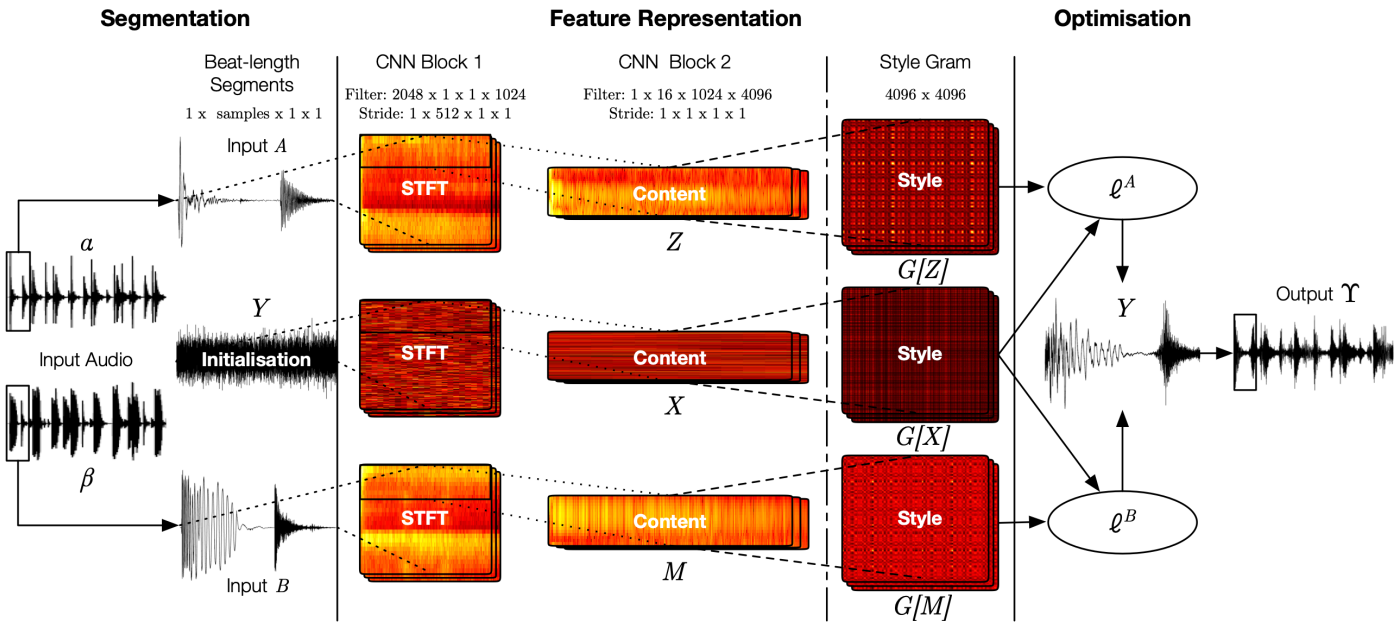

Figure 4.1: Audio style transfer with rhythmic constraints.

is reformulated into five rhythmically constrained transformation objectives to enable them to be independent of the input. These objectives are introduced in Section 4.2 along with the evaluation methodology. Section 4.3 presents evaluation results, where different reconstruction metrics of the transformed input audio signals are measured and compared to investigate the influence of the standard and proposed AST transformation objectives.

4.1 Method

The presented AST method seeks to take advantage of rhythmic structure and timbral characteristics of drum recordings. To that end a computational approach was developed which preserves the metre and the rhythmic structure of the chosen musical signal, while maintaining stylistic elements of both inputs. The method operates on beat-length audio segments and provides user-defined parameters for fine-tuning of interchangeable loss terms with regards to both inputs extending the conventional approaches to AST (Barry and Kim, 2018; Ulyanov and Lebedev, 2016). Figure 4.1 presents an overview of audio style transfer with rhythmic constraints. The method consists of three stages: (1) *segmentation*, in which the two audio files (α and β) are divided into beat-length segments (A and B respectively); (2) *feature representation*, in which feature representations (Z, M and X) of A, B and Y are created using a CNN; and (3) *optimisation* in which Y—a beat-length audio segment initialised with random noise—is iteratively transformed to simultaneously match loss functions related to the feature representations of A and B. The iterative transformation of the noise signal Y dheres to the standard approach documented in the literature (see Section 3.3.2). In this process, the input can be transformed to encapsulate the timbral and rhythmic characteristics of a target, which is associated with two audio recordings (α and

β). The resultant transformation Υ is a concatenation of the transformed beat-length segments Y. A solid-line box in the optimisation stage of Figure 4.1 highlights the first transformed segment Y.

4.1.1 Segmentation

Segmentation process in AST ensures that the audio inputs adhere to a larger metrical structure during the transformation. The segmentation also reduces the memory requirements outlined in previous AST systems (Barry and Kim, 2018; Grinstein et al., 2017; Mital, 2017). Using the segmentation process the transformation benefits from being constrained within rhythmic boundaries and reducing the computation cost of feature creation for the entirety of both inputs, which in turn frees the system to use arbitrarily long audio inputs.

In order for input audio files to be processed by the system, beat and downbeat positions are first extracted. The segment boundaries are computed using a downbeat tracking algorithm (Böck et al., 2016b) included in the madmom Python library (Böck et al., 2016a).[13] The detected beat positions, starting from the first downbeat, are used as segment boundaries denoted as A and B in Figure 4.1. The new noise segment Y uses the same length as segments A and B.

4.1.2 Feature Representation

The goal of the feature representation stage is to project input audio segments onto neural feature maps, leading to the creation of *content* and *style* matrices. While deep generative models employ various audio data representations—depending on tasks like rhythmic analysis, model training, transformation, and audio generation—this stage specifically introduces audio representations tailored to meet the transformation objectives elaborated on in the following sections. The lowest level audio representation used in deep generative models for audio synthesis is the sampled raw audio signal $\boldsymbol{a}$ (i.e., mono WAV file sampled at 22.05 kHz with 16-bit resolution). This uncompressed audio representation is lossless and can be converted to analogue sound using analogue-to-digital conversion systems. An audio input of N samples is converted into T frames of ρ samples. To extract framewise features a Hann window ω of length ρ is applied to each frame, with a hop size δ of $\frac{\rho}{4}$ samples. The resulting framed audio features $\mathbf{\Lambda}$ have dimensions $\rho \times T$. All input features are utilised in a fremewise manner. Before the framed audio can be processed by CNN Block 1 (see Figure 4.1), these raw audio samples must undergo transformation into a time-frequency representation.

Linear Spectrogram: A linear spectrogram is a time-frequency audio representation that facilitates access for DGMs to music characteristics such as pitch and timbre. Discrete Fourier transform is used for conversion of $\mathbf{\Lambda}$ into a frequency representation $\boldsymbol{\chi}$ with dimensions $\frac{\rho}{2} \times T$, where $\frac{\rho}{2}$ is the number of

[13] `https://github.com/CPJKU/madmom`

frequency bins. The k^{th} frequency bin in the t^{th} frame is calculated as follows:

$$\chi_t(k) = \sum_{n=0}^{N-1} a(n)\omega(n - t\delta)e^{\frac{-2\pi ikn}{N}}, \tag{4.1}$$

where $k \in [0, N-1]$ and i is the imaginary number unit. The linear spectrogram $\mathbb{S}$ represents the square of frequency magnitudes in $\boldsymbol{\chi}$ and is calculated as follows:

$$\mathbb{S} = |\chi|^2. \tag{4.2}$$

Linear Spectrogram with Learnable Parameters: A linear spectrogram can also be computed using a convolutional neural network layer, where the Fourier transform kernels can be trained via backpropagation, benefiting from a storage-efficient input preprocessing optimization (Choi et al., 2017). Within a convolutional neural network the k^{th} frequency bin is computed with two 1-dimensional convolutions, each of which is initialised with real and imaginary part of the discrete Fourier transform kernels, respectively. The convolutions describe the transform from Equation (4.1) as follows:

$$\chi(k) = \sum_{n=0}^{N-1} a(n) \cdot \left[cos\left(\frac{2\pi kn}{N}\right) - i \cdot sin\left(\frac{2\pi kn}{N}\right)\right]. \tag{4.3}$$

Following the representation of linear spectrograms for the three inputs A, B and Y, the neural style transfer aligns the distribution of activations in the feature maps of a network between a *content* input and a *style* input, resulting in a new output that combines the *content* of the first with the *style* of the second. This process is typically performed by minimising a loss function that compares the activations of the two inputs in the feature maps of the network.

4.1.2.1 Content

To create the *content* matrices (i.e., projections of the input rhythmic and timbral characteristics), the same two-stage process is performed in separate network branches for A, B and Y, where the weights of signal Y are initialised with random noise and matrices A and B contain input audio data. First, feature maps are created by projecting the audio onto STFT bases following the formulation in Equation (4.3). Then, the feature maps are projected further onto a larger number of channels to create the *content* representation (Barry and Kim, 2018; Grinstein et al., 2017; Mital, 2017; Ulyanov and Lebedev, 2016).

The input audio (A, B and Y) is segmented into T frames using a Hann window of n samples ($n = 2048$) and $\frac{n}{4}$ hopsize. A frequency projection of each of the frames is then created with a single CNN layer that uses filters initialised with real and imaginary parts (see Equation (4.3)) of the discrete Fourier transform resulting in a $\frac{n}{2}$xT spectrogram. The created spectrogram is converted into a

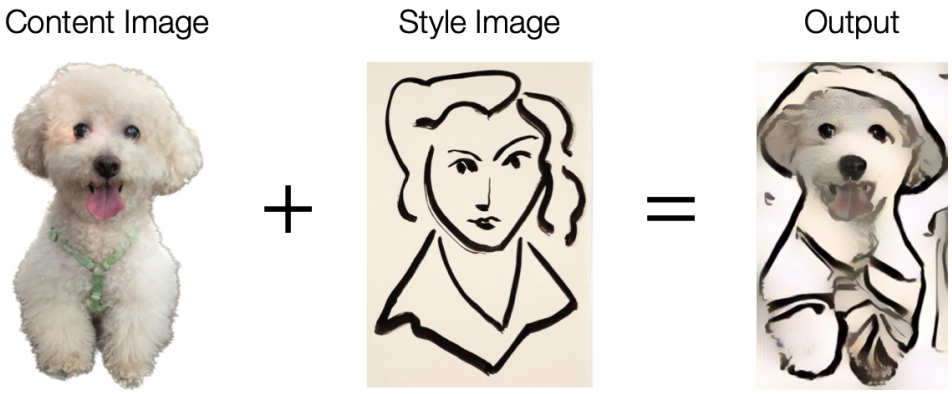

Figure 4.2: Example of image style transfer. A *content* image of a dog is transformed to contain *style* characteristics from a drawing by Henri Matisse.

log-magnitude representation. This transformation is represented in CNN Block 1 in Figure 4.1, where the filter size is nx1x1x$\frac{n}{2}$ with strides of 1x$\frac{n}{4}$x1x1.

CNN Block 2 in Figure 4.1 depicts neural feature computation from the STFT projections that becomes the *content* and can be understood as the low-level features of the input. The CNN architecture consists of a single convolutional layer with a filter size of 1xHxFxQ, where H is the number of time frames convolved with the filter, F is the number of frequency bins and Q represents the number of frequency channels onto which the input spectrogram will be projected. The filter size used throughout the architecture of CNN Block 2 is 1x16x$\frac{n}{2}$x2n. The temporal receptive field (i.e., a contextual window modelled by each hidden state of the network) of 16 frames (∼370ms) is used to capture acoustic information about instruments from a context longer than a half-beat length at 120 BPM. Each network is followed by a SeLU activation layer (see Section 3.1.1), represented as vertical solid-dashed line in Figure 4.1. The *content* matrices for A, B and Y are termed Z, M and X respectively.

4.1.2.2 Style

Style can be understood as high-level information of the input neural features, which are decomposed into patches (e.g., texture patches in images and timbre of sounds). Neural style transform uses these patches for specifying a complex similarity function between the generated output and the input *content*. In this transformation, the *content* of input audio files is shifted into a new *style* domain defining a statistical representation of feature maps generated from previous layers of the network. This feature space is designed to capture texture or intra-feature map statistics. Figure 4.2 illustrates an example of image style transfer with a photograph of a sitting dog (i.e., *content*) which is transformed to contain *style* characteristics extracted from the drawing of a woman by Henri Matisse (UMMA, 2017). The transformed output preserves the *content* of the sitting dog and applies the line art *style* of Matisse's drawing. Similarly in music, a standard audio style transfer approach will preserve the *content* of each sound and affect its timbre acquired from the *style* recording.

To obtain a representation of the *style* of an input spectrogram, a Gram matrix G is used as in Gatys et al. (2015). Let F be any content matrix Z, M and X. For each $F_l = [f_{l,k}]_{q=1}^{Q_l}$ a Gram matrix G is

calculated using the inner product $G[F] := F^T F \in \mathbb{R}^{Q_l \times Q_l}$, where the intra-feature map statistics are extracted over channels Q at layer $l \in L$. The style reconstruction loss can then be calculated as a sum over multiple layers of the Gram and *content* activations (see Equations (4.6) and (4.7)).

4.1.3 Optimisation

Network weights corresponding to the output audio segment Y are updated in the optimisation stage of the system (see Figure 4.1). The updates are directed by a predefined combination of objective *content* and *style* loss functions.

4.1.3.1 Content and Style Loss Functions

The individual ℓ terms can be added and changed according to the transformation objective with regard to the inputs A and B. The *content* loss ℓ_C is a squared error loss between *content* matrix X and the input audio *content* matrices (Z or M):

$$\ell_C^A = \frac{1}{2} \sum_{l \in L} ||X_l - Z_l||^2, \tag{4.4}$$

$$\ell_C^B = \frac{1}{2} \sum_{l \in L} ||X_l - M_l||^2. \tag{4.5}$$

The *style* reconstruction loss ℓ_S is the sum of the squared difference between the transformed Gram matrix $G[X]$ and the input Gram matrices ($G[Z]$ and $G[M]$):

$$\ell_S^A = \frac{1}{Q^2} \sum_{l \in L} (G[X]_l) - G[Z]_l)^2, \tag{4.6}$$

$$\ell_S^B = \frac{1}{Q^2} \sum_{l \in L} (G[X]_l - G[M]_l)^2. \tag{4.7}$$

The motivation for using the *style* loss as formulated above is to preserve the statistics about the convolutional representation over the entire input while losing local information about exactly where different elements are. As shown by Li et al. (2017), the style reconstruction loss can also be interpreted using a specific maximum mean discrepancy as the similarity function (Equation (3.33)). This reveals that neural style transfer algorithm can also be seen as a process of distribution alignment of the neural activations between images.

In order to control the contributions of *content* and *style* from the two inputs, the total loss $\mathcal{L}$ is expressed as a sum of *content* ℓ_C and *style* ℓ_S loss functions for the input audio segments A and B as follows:

$$\mathcal{L} = \sigma \ell_C^A + \delta \ell_C^B + \theta \ell_S^A + \phi \ell_S^B, \tag{4.8}$$

where σ, δ, θ and ϕ are proportion parameters that add up to 1 and help configure loss preferences between the input recordings. The standard AST loss term consists of only ℓ_C^A content and ℓ_S^B style functions from two inputs. The proposed loss formulation in Equation (4.8) allows for rhythmic transformation and timbre blending possibilities through the addition of loss interchangeability. Loss terms can be removed according to their corresponding transformation configuration and user-defined parameterisation. The standard and proposed AST loss term formulations are described in more detail in Section 4.2.3.

4.1.3.2 Training

Different combinations of *style* and *content* loss functions are used to shape the output of the transformation. Following Barry and Kim (2018), the magnitudes of the gradients of loss terms are normalised to 1 to moderate the imbalances in weighting of either function. The limited-memory BFGS (Zhu et al., 1997) gradient descent-based optimisation algorithm is used due to its appropriateness in non-linear problems related to neural style transfer (Gatys et al., 2015; He et al., 2016b). Once initialised, the feature map representations of *content* and *style* from inputs A and B do not change throughout the training stage. In each gradient step the *content* and *style* activations are back-propagated all the way to the network output Y. Hence, only weights originating from Y are being manipulated during the optimisation process, while all feature representations remain unchanged for inputs A and B. The optimisation of the concerned weights is stopped after 5000 iterations. With an NVIDIA Tesla M40 computing processor each algorithm iteration takes an average of 280 milliseconds.

The system is implemented using the Tensorflow Python library.[14] The processing branches of A, B and Y are part of the same CNN in one Tensorflow computation graph. This means that the neural representations of the input time-domain audio Y can be optimised simultaneously in one stage.

4.2 Evaluation Methodology

The following sections introduce the evaluation strategies used to assess the rhythmic and timbral modification capabilities of the system presented in Section 4.1. The evaluations concern three main transformation objectives, namely, the standard audio style transfer formulation, the proposed mashup and the augmented mashup transformations. In addition, a specially curated dataset of drum loops with different rhythmic pattern styles is presented.

4.2.1 Data

The compiled dataset includes 30 drum loops (mono WAV sampled at 22.05 kHz with 16-bit resolution) of 4 bars in length, which differ in rhythmic patterns consisting of various types of kick and snare drums. The top of Figure 4.3 visualises rhythmic patterns present in the dataset using a 3-dimensional image, where 30 rhythmic envelopes R_S (see Equation (4.16)) are plotted over time, where colour intensity

[14] `https://www.tensorflow.org/`

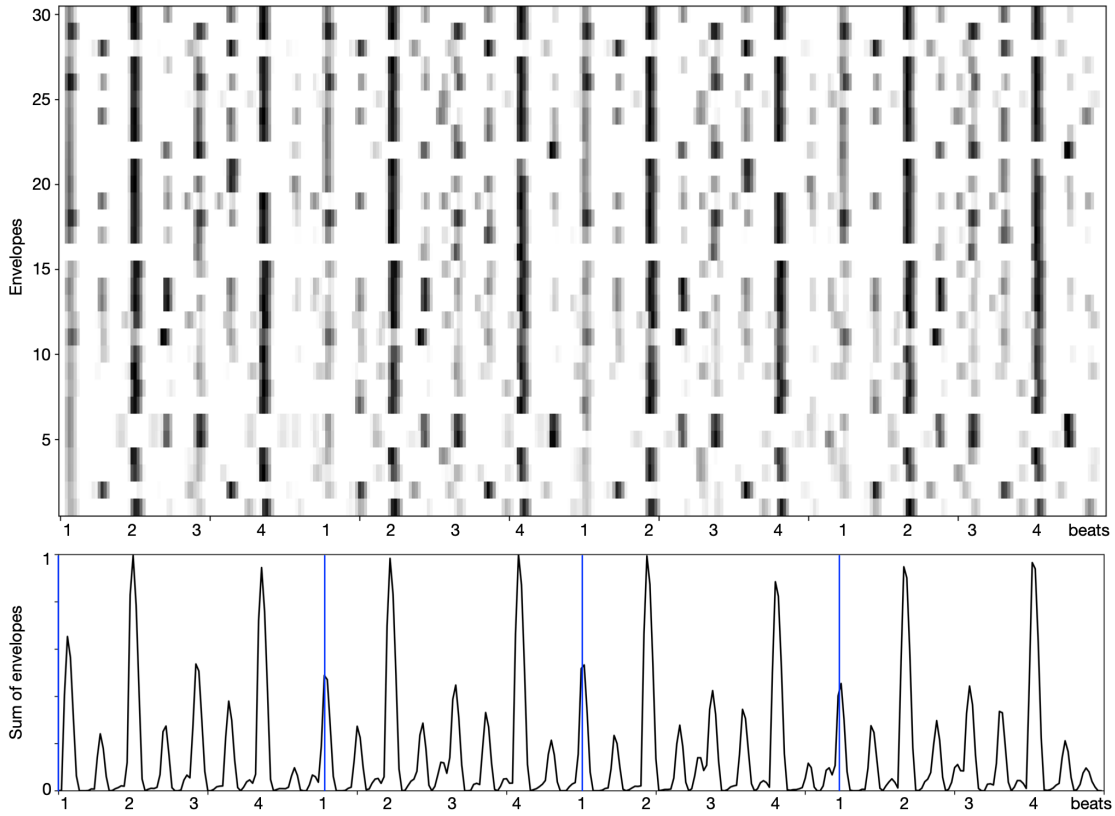

Figure 4.3: Patterns represented as rhythmic envelopes from 30 drum loops in the dataset (top). Sums of rhythmic envelopes computed across the dataset (bottom), where bar boundaries (i.e., 4 bars) are represented with vertical blue lines.

in each row represents amplitude of each rhythmic envelope. Darker shades indicate higher peaks and lighter shades indicate smaller peaks in each envelope. Each R_S is normalised to values $[0, 1]$. The bottom of Figure 4.3 shows normalised sum of all envelopes present in the dataset over 4 bars. Rhythmic patterns present in the dataset indicate a strong presence of events (e.g., kicks) on downbeats in most examples as well as a strong presence of events (e.g., snares) on beats two and four characteristic of popular music. There are several exceptions such as patterns 5, 6 and 16 shown in top of Figure 4.3 which include more syncopation with less events on downbeats but more events around third beats in each bar. In addition, pattern 16 is also the only example which does not start with an event on the first downbeat position.

The tested drum loops were generated with twelve different pattern styles—denoted by the first part of each file name in Table 4.1. All transformation examples are created from 15 pairs of input drum loops to spread over a variety of different playing styles defined by the Logic X Drummer virtual instrument.[15] Each pair is assigned a file termed α and another termed β. The transformation pairs were chosen such that there is no overlap between the same rhythmic pattern styles. The rightmost column in Table 4.1 presents rhythmic cosine similarities (see Section 4.2.4.1) calculated between pairs of envelopes. Higher values indicate more similar patterns with greater overlap of events at different metrical positions while lower values indicate patterns that exhibit less overlap indicate more dissimilar drum loops. The mean

[15] `https://support.apple.com/kb/PH13070`

	α	β	RCS
1	hp_02	cl_01	0.84
2	ep_01	ob_02	0.66
3	ee_02	ob_03	0.32
4	gs_02	cl_03	0.79
5	ep_03	hp_03	0.68
6	br_03	km_03	0.89
7	mt_02	cl_02	0.84
8	mt_01	km_01	0.81
9	nk_01	hp_01	0.05
10	ctp_01	nk_02	0.09
11	ctp_02	nk_03	0.29
12	mt_03	br_01	0.85
13	ep_02	br_02	0.76
14	gs_03	ee_01	0.30
15	gs_01	km_02	0.30
mean(α,β)			0.56

Table 4.1: Compiled dataset with 30 drum loops and 15 transformation pairs used in system evaluations. Inputs α and β are represented with an ID indicating different rhythmic rhythmic styles defined by Logic X Drummer virtual instrument. Rightmost column shows rhythmic cosine similarities (RCS) calculated for each transformation pair together with the mean RCS of all examples α and β.

similarity in the last row of the table shows that all test recording pairs present a varied combination of differing rhythmic patterns. This is important as evaluation of rhythmic transformation capabilities of the proposed system would not be possible if the tested examples were rhythmically identical. This pairing configuration also contributed to reduction of the computation cost of performing all possible transformation combinations. All drum loops have a fixed-tempo set to 120 BPM in $\frac{4}{4}$ metre. The motivation for using a fixed-tempo of 120 BPM was to test how this transformation performs on audio recordings typically used in the processes of remixing and mashup creation. Both processes rely on automatic and manual beat-synchronisation of multiple audio inputs for manipulation of rhythm while preserving listener comfort levels (Ishizaki et al., 2009). The chosen tempo is a typical tapping speed (Moelants, 2002) and is common for many genres in popular and electronic music (e.g., rock, pop, house). It is also the default tempo in various DAWs used in music production (e.g., Logic Pro, Ableton Live, Pro Tools).[16,17,18]

4.2.2 Evaluation Strategies

The rhythmic and timbral modification characteristics of the proposed AST system are evaluated with multiple reconstruction metrics assessing similarities computed between the input and the transformed output audio signals. The system is used to generate new drum audio examples Υ from 15 transformation drum loop pairs from the compiled dataset. The transformation pairs are created from two sets of drum loop inputs termed α and β. To obtain the results, feature representations for evaluation metrics introduced in Section 4.2.4 are created from Υ and compared to those of α and β. The reported metrics

[16]https://www.apple.com/uk/logic-pro/
[17]https://www.ableton.com/
[18]https://www.avid.com/pro-tools

are presented as means across all transformations from three comparisons between the following drum loop pairs: (α, Υ), (β, Υ), and (α, β) for five transformation objectives. For example, a comparison between (α, Υ) for $\mathcal{L}_v$ is represented as $\mathcal{L}_v^{\alpha}$, and a comparison between (β, Υ) as $\mathcal{L}_v^{\beta}$, where v denotes the transformation objective. The following section details transformation objectives which include the standard AST loss formulations and newly proposed AST loss formulations for mashup and augmented mashup transformations.

4.2.3 Transformation Objectives

Different combinations of loss objectives correspond to a separate transformation. The standard AST objective (Gatys et al., 2015) consists of loss functions for a single *content* input and a single *style* input. To perform a transformation of rhythmic patterns as well as the timbral *content* of drum recordings the standard AST objective is extended to include two additional mashup transformations which incorporate combinations of the *content* and *style* loss functions.

4.2.3.1 Standard Audio Style Transfer Formulation

The standard audio style transfer objective uses the content and style reconstruction loss functions. Following Gatys et al. (2015), this objective is formulated as follows:

$$\mathcal{L}_1 = \ell_C^A + \ell_S^B, \tag{4.9}$$

where, the *content* information of input A is transformed to acquire features from the *style* input B and vice versa as follows:

$$\mathcal{L}_2 = \ell_S^A + \ell_C^B. \tag{4.10}$$

This standard formulation of the neural style transfer objective has been utilised in a number of systems for audio style transfer (Barry and Kim, 2018; Grinstein et al., 2017; Mital, 2017; Perez et al., 2017; Ulyanov and Lebedev, 2016; Verma and Smith, 2018; Wyse, 2017) as well as systems for image style transfer (Jing et al., 2019). An example of the standard image style transfer objective is illustrated in Figure 4.2, where the *content* loss ℓ_C^A is calculated from the image of a dog and the *style* Gram characteristics are computed in ℓ_S^B loss from the drawing.

4.2.3.2 Mashup Transformation

In this objective, the transformation uses solely the *style* feature representations to mix rhythmic and timbral characteristics of both recordings. The proposed mashup transformation is formulated as follows:

$$\mathcal{L}_3 = \ell_S^A + \ell_S^B, \tag{4.11}$$

where each input optimises the squared difference between style Gram matrices. This loss function is akin to a mashup of both audio inputs and extends the standard AST formulation with the ability of moving acoustic events to create rhythmically new performances. In addition, the proportion of the used ℓ can be changed with the corresponding proportion parameter from Equation (4.8) to produce the desired transformation.

4.2.3.3 Augmented Mashup Transformation

This objective reinforces the mashup transformation with more *content* information from one the inputs (e.g., A and B). The augmented mashup formulation is computed in order to move the transformation towards the rhythmic performance present in input A as follows:

$$\mathcal{L}_4 = \mathcal{L}_3 + \ell_C^A. \tag{4.12}$$

To move acoustic events to create a rhythmically new performance that is more similar to input B the augmented mashup is computed as:

$$\mathcal{L}_5 = \mathcal{L}_3 + \ell_C^B. \tag{4.13}$$

4.2.4 Metrics

The metrics used in the evaluations measure rhythmic and timbral reconstruction capabilities of the presented system. The rhythmic constraints are tested using rhythmic similarity as well as the standard onset detection metrics. Timbral reconstruction and audio quality of the system are evaluated using spectral cosine similarity and Pearson correlation coefficients. Below, the feature representations necessary for rhythmic and timbral evaluations are presented, with the following sections providing a detailed description of each metric.

Logarithmic Spectrogram: To more closely match human perception of loudness and make the evaluation and training of deep learning models more computationally tractable, the dimensionality of a linear spectrogram can be reduced and converted to a logarithmic scale. The linear frequency bins, measured in Hz, are converted to a logarithmic scale using a preset number of bands b per octave. The filters of the logarithmic filter bank $\boldsymbol{FB}$ are computed as follows:

$$FB_b^k = \begin{cases} 0, & k < g(b-1) \\ \frac{k-g(b-1)}{g(b)-g(b-1)}, & g(b-1) \leqslant k \leqslant g(b) \\ \frac{g(b+1)-k}{g(b+1)-g(b)}, & g(b) \leqslant k \leqslant g(b+1) \\ 0, & k > g(b-1), \end{cases} \tag{4.14}$$

where $g()$ represents the logarithmically-spaced frequencies for the total of B number of bands. The logarithmic spectrogram is calculated through matrix multiplication of the filter bank $\boldsymbol{FB}$ and the linear spectrogram $\mathbb{S}$.

Mel Spectrogram: The Mel scale is a perceptually motivated scale, which overcomes the fact that the FFT has a linear resolution, where all frequency bins are evenly spaced (Stevens and Volkmann, 1940). Since human hearing is not evenly spaced and humans hear pitch exponentially not linearly, the Mel scale ensures a better frequency resolution in the lower frequencies than the higher ones by converting the frequency f in Hz to the Mel scale m using:

$$m = 2595 \log_{10} \left(1 + \frac{f}{700}\right). \tag{4.15}$$

The Mel filter bank can be created using the filter bank computation from Equation (4.14) with $g()$ Mel-spaced frequencies acquired from Equation (4.15) for the total B bands (e.g., $B = 120$). Similarly to computation of a logarithmic spectrogram, the Mel spectrogram is derived through matrix multiplication of the linear spectrogram with the Mel filter bank.

4.2.4.1 Rhythmic Similarity

To test the rhythmic constraints imposed by different transformation objectives within the AST technique, the rhythmic similarity is used to compare pairs of transformations (Davies et al., 2013, 2014a; Hockman, 2014). The rhythmic envelopes R are calculated from the spectral difference function (SDF) (see bottom of Figure 2.7) of a new audio Υ with inputs α or β. Following Dixon (2006), the SDF is computed over a linear spectrogram $\mathbb{S}$ (see Equation (4.1)) with k bins and t frames as follows:

$$R_{\mathbb{S}}(t) = \sum_{k=0}^{K-1} \big\{\mathcal{H}(|\mathbb{S}_k(t+1)| - |\mathbb{S}_k(t)|)\big\}, \tag{4.16}$$

where $\mathcal{H}(x) = \frac{(x+|x|)}{2}$ is the half-wave rectifier function, which returns zero for negative arguments. The STFT parameters from Section 4.1.2 are used for computation of all spectral representations. All envelopes are normalised to range from 0 to 1. Rhythmic cosine similarity (RCS) between every pair of envelopes is calculated as:

$$RCS(\omega, \Upsilon) = \frac{R_\omega \cdot R_\Upsilon}{\|R_\omega\| \|R_\Upsilon\|}, \tag{4.17}$$

where ω is a placeholder representing inputs of either α or β. Thus, the rhythmic similarity will be close to unity for very similar patterns and nearer to zero for dissimilar patterns. RCS values calculated for 15 transformation pairs are shown in Table 4.1 together with the mean RCS calculated from all pairs.

4.2.4.2 Spectral Cosine Similarity

Spectral cosine similarity (SCS) is used to measure timbral reconstruction capabilities of the proposed transformations and was previously employed in evaluations of deep generative models for music signals as an objective measure of audio quality (Ramírez et al., 2021). Let log-spectral power magnitudes $\mathcal{S}$ be defined as $\mathcal{S}(t,k) = \log_{10}|\mathcal{S}|^2$, where $\mathcal{S}$ represents a Mel spectrogram with 120 Mel bands (see Equation (4.15)) computed using STFT parameters from Section 4.1.2. Spectral cosine similarity is calculated as:

$$SCS(\omega, \Upsilon) = \frac{\mathcal{S}_\omega \cdot \mathcal{S}_\Upsilon}{\|\mathcal{S}_\omega\|\|\mathcal{S}_\Upsilon\|}. \tag{4.18}$$

4.2.4.3 Pearson Correlation

Pearson correlation (PC) coefficients are used as a metric measuring the quality of generated musical signals (Kim et al., 2019). Similarly to the spectral cosine similarity metric, the PCs are calculated using Mel spectrograms $\mathcal{S}$, which offer a representation that effectively captures the fundamental frequencies of drums. Pearson correlation coefficients are computed following Gonzalez and Wintz (1987):

$$PC(\omega, \Upsilon) = \frac{\sum(\mathcal{S}_\omega - m_\omega) \cdot (\mathcal{S}_\Upsilon - m_{\mathcal{S}_\Upsilon})}{\sqrt{\sum(\mathcal{S}_\omega - m_\omega)^2 \cdot \sum(\mathcal{S}_\Upsilon - m_\Upsilon)^2}}, \tag{4.19}$$

where m_ω and m_Υ are means of input $\mathcal{S}_\omega$ and transformation $\mathcal{S}_\Upsilon$, respectively.

4.2.4.4 Onset Detection

Onset detection performance metrics are used to evaluate the extent of the rhythmic modifications generated by the proposed transformation objectives. While rhythmic similarity is based on inherent representation of rhythmic patterns as temporal envelopes the onset detection evaluation compares numbers of events that were found against the target ground truth events in the temporal domain. The standard metric used to evaluate performance of event detection algorithms is the mean F-measure (Bello et al., 2005). A tolerance window is used to determine whether a detected onset is within an acceptable range away from the onset in the target event. Here a window of $\pm$50 ms is used which is a standard size seen in the automatic drum transcription literature (Southall, 2019; Southall et al., 2016; Wu et al., 2018; Wu and Lerch, 2015). Once an event is detected in the acceptable range it is labelled as a true positive (TP). If a detected onset is not within this range it is labelled as a false positive (FP). Alternatively, if a target ground truth event does not coincide with any detected onset, it is labelled as a false negative (FN). Using these counts, precision (P) and recall (R) metrics can be calculated. Precision is the fraction of true positives among all detected events, while recall is the fraction of true positives among all missed ground truth events. They are calculated as follows:

$$P = \frac{TP}{TP + FP}, \tag{4.20}$$

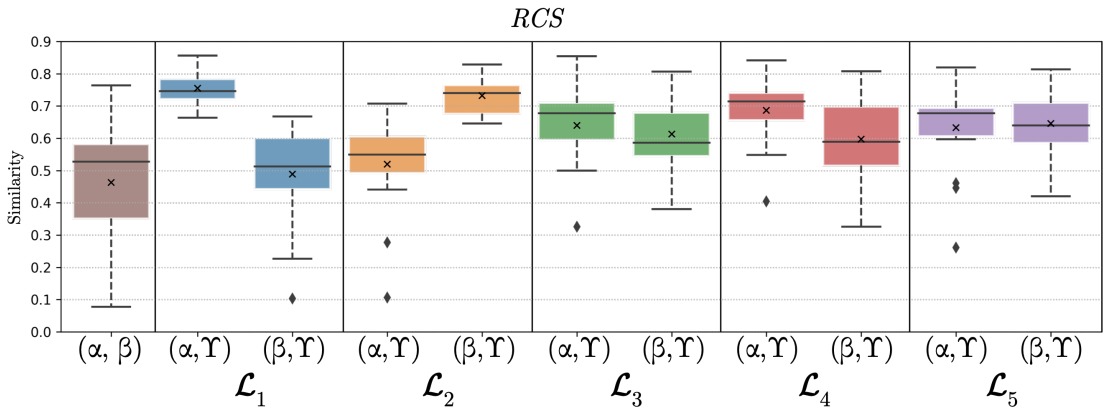

Figure 4.4: Mean rhythmic cosine similarity (RCS) results. Crosses ('x') indicate means per $\mathcal{L}$ objective and comparison between drum loop pairs: (α, Υ), (β, Υ), and reference (α, β). A higher rhythmic cosine similarity indicates that transformations Υ are more similar to either input α or β, with the reference for this comparison depicted in brown on the left-hand side.

$$R = \frac{TP}{TP + FN}. \tag{4.21}$$

The precision and recall can be expressed as a single metric using the F-measure (F) calculation as follows:

$$F = 2\frac{PR}{P + R}. \tag{4.22}$$

All reported results were generated using a state-of-the-art OnsetDetector algorithm (MIREX, 2018a) introduced in Böck et al. (2012b). Drum events are detected automatically for inputs α and β as well as generations Υ for each transformation objective. The detected events from Υ are then used for computation of F-measure when compared against each of the two inputs. Additionally, F-measure is calculated to approximate the baseline performance between the two sets of α and β with, respectively, 368 and 410 annotated onsets in each set.

4.3 Results

Evaluated transformations along with other examples are presented on the supporting website.[19]

4.3.1 Rhythmic Similarity Results

4.3.1.1 Standard Audio Style Transfer: $\mathcal{L}_1$ and $\mathcal{L}_2$

Figure 4.4 presents the mean RCS results for five transformation objectives and different drum loop pairings. A higher rhythmic cosine similarity suggests that the transformations, denoted by Υ, are more similar to either input α or β. For comparison, the reference of rhythmic patterns between α and β is represented in brown (i.e., left-hand side of the diagram). The comparisons between α and β offer a

[19] https://maciek-tomczak.github.io/maciek.github.io/Audio-Style-Transfer-with-Rhythmic-Constraints

baseline to understand the inherent similarity between the inputs before any transformation is applied. Objectives $\mathcal{L}_1$ (blue) and $\mathcal{L}_2$ (orange) denote the standard audio style transfer loss objectives. In these cases, the rhythmic patterns remain unchanged, indicating an unsuccessful transformation. This means that transformations where the rhythmic patterns do not change should ideally exhibit high similarity to inputs α in $\mathcal{L}_1$ and to inputs β in $\mathcal{L}_2$.

The results show that standard AST loss objectives $\mathcal{L}_1$ and $\mathcal{L}_2$ do not achieve transformation of rhythmic patterns. In both transformations, the RCS means for $\mathcal{L}_1^{\alpha}$ and $\mathcal{L}_2^{\beta}$ are higher than $\mathcal{L}_1^{\beta}$ and $\mathcal{L}_2^{\alpha}$, indicating that the rhythmic patterns did not change in the standard AST formulation. Results from t-tests performed across tracks and objectives $\mathcal{L}_1^{\alpha}$ and $\mathcal{L}_2^{\beta}$ demonstrate that the impact of the respective *content* losses ℓ_C^A and ℓ_C^B is significant ($\rho < 0.05$). This demonstrates that the *content* losses determine the rhythmic patterns of the output transformations and the *style* losses ℓ_S^B and ℓ_S^A are not capable of changing rhythmic patterns in the $\mathcal{L}_1$ and $\mathcal{L}_2$ objectives for neither of the inputs. This characteristic of the standard AST objectives is also supported when compared with baseline mean RCS results computed between inputs α and β (i.e., brown box), which are similar with $\mathcal{L}_1^{\beta}$ and $\mathcal{L}_2^{\alpha}$. This means that for these two objectives the transformations are as different from inputs α and β, respectively, as the baseline comparisons between the two inputs resulting in no rhythmic transformation of the input drum recordings.

4.3.1.2 Mashup Transformation: $\mathcal{L}_3$

The proposed mashup transformation objective $\mathcal{L}_3$ (green) generates new rhythmic performances that combine rhythmic features of both inputs. The expectation here is to observe distinct patterns that might not align closely with either α or β. The similar RCS means $\mathcal{L}_3^{\alpha}$ and $\mathcal{L}_3^{\beta}$ indicate that new drum events were created in the generations Υ that were not present in the outputs of the standard AST objectives $\mathcal{L}_1$ and $\mathcal{L}_2$. The variability of both generations for $\mathcal{L}_3$ is also similar with a 0.03 higher mean rhythmic similarity for the comparison between α and Υ. This indicates a slightly larger overlap between rhythmic patterns of the output transformations and inputs α. This difference is consistent with the standard AST objective comparisons where mean results for $\mathcal{L}_1^{\alpha}$ show a 0.02 higher rhythmic similarity than results for $\mathcal{L}_2^{\beta}$. The comparisons of $\mathcal{L}_3$ and means calculated between α and β as well as comparisons of $\mathcal{L}_3$ with $\mathcal{L}_1^{\beta}$ and $\mathcal{L}_2^{\alpha}$ are significant ($\rho < 0.05$). This demonstrates that the proposed mashup transformation is capable of generating drum loops with new rhythmic patterns that are not changed when using the standard AST objectives.

4.3.1.3 Augmented Mashup Transformation: $\mathcal{L}_4$ and $\mathcal{L}_5$

The augmented mashup transformations $\mathcal{L}_4$ (red) and $\mathcal{L}_5$ (purple) enhance the proposed rhythmic transformation with the *content* information acquired from ℓ_C^A and ℓ_C^B loss functions computed on inputs α and β, respectively. Objectives $\mathcal{L}_4$ and $\mathcal{L}_5$ are designed to guide the rhythmic transformations to be more similar to a selected input. Ideally, transformations under these objectives should exhibit higher similarities to one of the inputs, either α in $\mathcal{L}_4$ or β in $\mathcal{L}_5$.

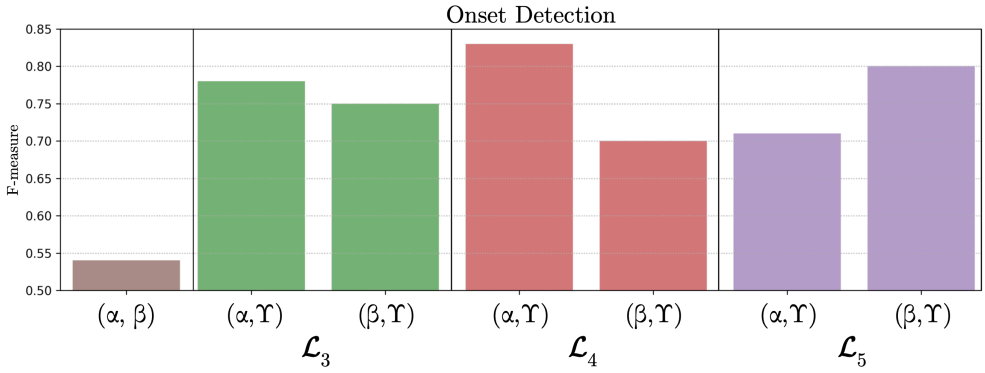

Figure 4.5: Mean F-measure results calculated from different pairs of inputs (i.e., α and β) and output transformations Υ. A higher F-measure indicates that the transformations Υ are more similar to either input α or β, with the reference for this comparison depicted in brown on the left-hand side (i.e., (α, β)).

The comparisons of objectives $\mathcal{L}_4$, $\mathcal{L}_5$ and mean RCS results between α and β are significant ($\rho < 0.05$), which shows the augmented mashup transformations created new rhythmic patterns differing from the baseline. The higher RCS means of objectives $\mathcal{L}_4^{\alpha}$ and $\mathcal{L}_5^{\beta}$ show that rhythmic patterns of Υ can be controlled to be more similar to the respective input when the content loss function is used. The mean similarities for $\mathcal{L}_5^{\beta}$ transformations are 0.08 higher than similarities of $\mathcal{L}_3^{\beta}$ which indicates that new drum events were created, which were not initially present in inputs β. While the difference between means of $\mathcal{L}_5^{\alpha}$ and $\mathcal{L}_5^{\beta}$ is small it is consistent with the results in $\mathcal{L}_3$ comparisons, where the Υ were slightly more similar to the inputs α. In addition to the augmented mashup objective the user can aid the transformation using proportion parameters from the proposed loss formulation shown in Equation (4.8) which can add more controllability over the possible output transformations. Consequently, the rhythmic pattern of the output can be fine-tuned to exhibit more similarities to the target input.

4.3.1.4 Onset Detection: $\mathcal{L}_3$, $\mathcal{L}_4$, and $\mathcal{L}_5$

Figure 4.5 shows F-measures calculated from comparisons between Υ and both inputs for the proposed mashup and augmented mashup transformation objectives. An additional reference comparison between inputs α and β, with a standard deviation of 0.091, is shown in brown on the left-hand side of the diagram. The mean F-measure results for $\mathcal{L}_3$, for comparisons (α, Υ) and (β, Υ) with standard deviations of 0.057 and 0.064, respectively, are consistent and have little variability. This consistency in performance supports the desired outcome for a mashup transformation. Additionally, the slightly higher similarity of Υ to α in $\mathcal{L}_3$ can also be seen in rhythmic cosine similarity results and can be due to a larger overlap of events in these transformation pairs. The results in Figure 4.5 also highlight that as different *content* loss objectives are used the F-measure performance decreases (e.g., $\mathcal{L}_4^{\beta}$ and $\mathcal{L}_5^{\alpha}$). The mean differences between $\mathcal{L}_4^{\alpha}$ and $\mathcal{L}_4^{\beta}$ as well as $\mathcal{L}_5^{\alpha}$ and $\mathcal{L}_5^{\beta}$ are also larger than RCS means for the augmented mashup objectives. Both $\mathcal{L}_4^{\beta}$ and $\mathcal{L}_5^{\beta}$ have standard deviations of 0.071 and 0.066, respectively, which are slightly higher than their counterparts. Specifically, $\mathcal{L}_4^{\alpha}$ and $\mathcal{L}_5^{\alpha}$ have standard deviations of 0.054 and 0.051, suggesting they exhibit lower variability. The comparisons of mean results of the three objectives and α

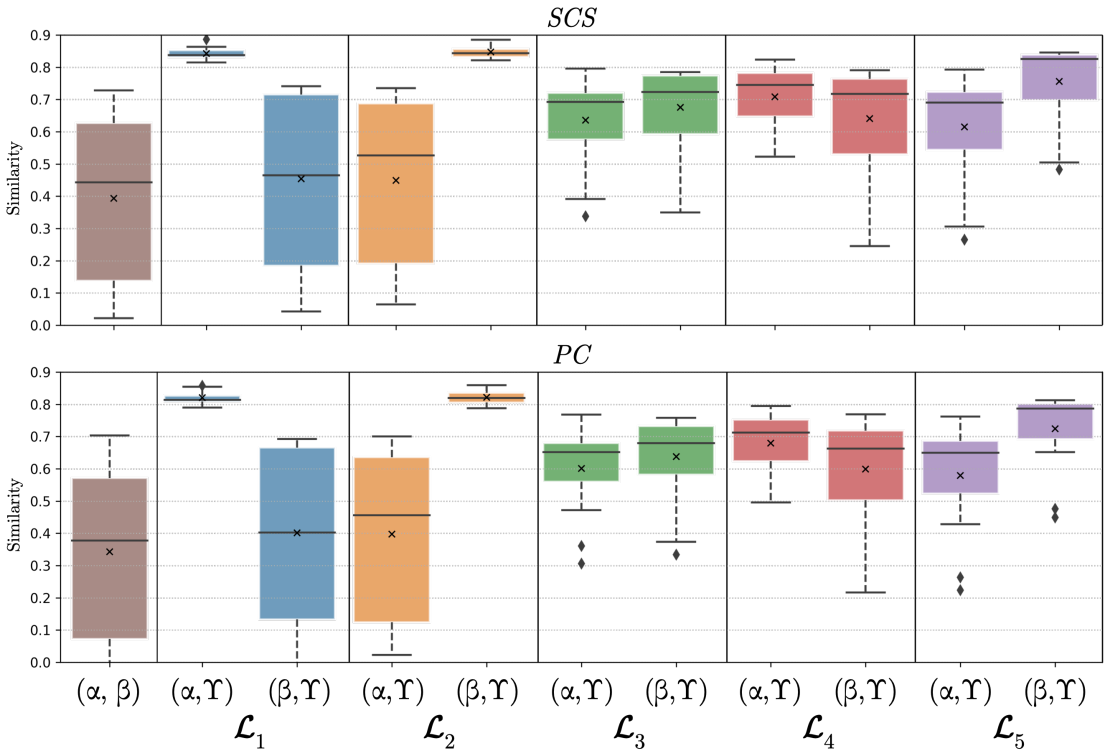

Figure 4.6: Mean spectral cosine similarity (SCS) and Pearson correlation (PC) results. Crosses ('x') indicate means per $\mathcal{L}$ objective and comparison between drum loop pairs: (α, Υ), (β, Υ), and (α, β).

and β are significant ($\rho < 0.05$). This demonstrates that the system generated new events in objectives $\mathcal{L}_3$, $\mathcal{L}_4$, and $\mathcal{L}_5$ which were not present in the original inputs, suggesting successful transformations.

4.3.2 Spectral Similarity Results

The top diagram of Figure 4.6 presents the mean SCS results. The mean results using PC coefficients are presented in two forms. The first, shown in the bottom diagram of Figure 4.6 as means calculated across Mel bands as well as $\mathcal{L}_v$ and different drum loop pairs. The second, plotted across 120 Mel bands in Figure 4.7 as mean results calculated for objectives $\mathcal{L}_v$ as well as comparisons between drum loop pairs PC(α, Υ) (top) and PC(β, Υ) (bottom).

4.3.2.1 Standard Audio Style Transfer: $\mathcal{L}_1$ and $\mathcal{L}_2$

Mean SCS results for the standard AST objectives $\mathcal{L}_1$ and $\mathcal{L}_2$ highlight the spectrogram reconstruction capabilities of the audio style transfer technique. Both $\mathcal{L}_1^{\alpha}$ and $\mathcal{L}_2^{\beta}$ comparisons represent the uppermost reconstructions for the standard AST objective which replicates the spectrograms of the content losses of ℓ_C^A and ℓ_C^B , respectively. There exists some inherent loss of audio quality previously attributed to the addition of high-level information from the *style* loss (i.e., Gram matrix) (Grinstein et al., 2017), which can be seen in the imperfect reconstructions for both $\mathcal{L}_1^{\alpha}$ and $\mathcal{L}_2^{\beta}$. A similar reconstruction trend can be seen in mean PC results in the bottom of Figure 4.6.

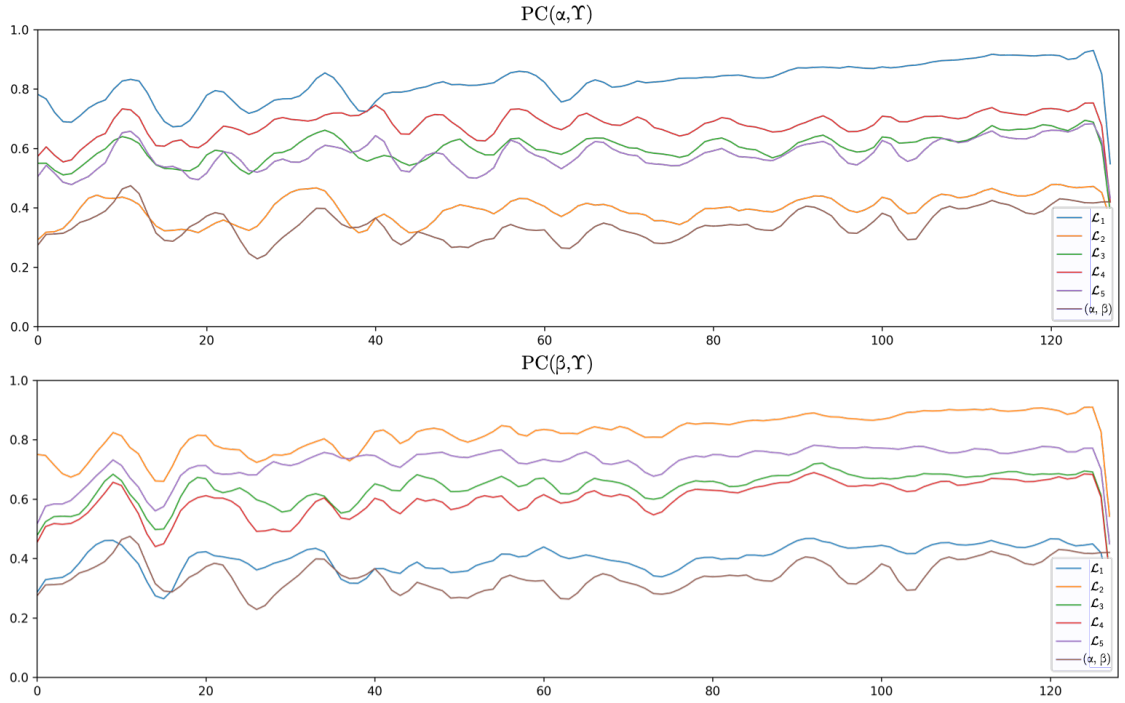

Figure 4.7: Mean Pearson correlation (PC) results. Mean PCs are plotted across Mel bands for different objectives $\mathcal{L}_v$ and different drum loop comparisons PC(α, Υ) (top) and PC(β, Υ) (bottom).

A visualisation of PC reconstructions over Mel frequency bands can be seen in Figure 4.7, where $\mathcal{L}_1^{\alpha}$ (blue line in top plot) and $\mathcal{L}_2^{\beta}$ (orange line in bottom plot) show on average lower reconstruction capabilities in the lower frequencies (Mel bands < 60) than higher frequencies (Mel bands > 60). Conversely, the lowest reconstruction comparison of the model can be seen in the $\mathcal{L}_1^{\beta}$ (blue) and $\mathcal{L}_2^{\alpha}$ (orange) as well as the baseline comparisons between inputs α and β shown in brown. Both objectives are closely correlated with the baseline across the frequency range which is also reflected by similar means shown in Figure 4.6.

4.3.2.2 Mashup Transformation: $\mathcal{L}_3$

The mean SCS in top of Figure 4.6 for both $\mathcal{L}_3^{\alpha}$ and $\mathcal{L}_3^{\beta}$ are similar indicating an equal timbral mashup between drum loops, with a slightly increased similarity towards inputs β. Mean SCS results for the $\mathcal{L}_3$ mashup transformation are comparable with the RCS means. Interestingly, the spectral reconstructions for $\mathcal{L}_3$ reverse the trend visible in the rhythmic similarity and onset detection results, where Υ are more similar to inputs α (see Figures 4.4 and 4.5). In contrast, SCS and PC indicate that mashups Υ acquired more timbral characteristics from the inputs β despite the identical weighting used in all transformations (see Equation (4.8)). Both spectral similarity comparisons of $\mathcal{L}_3$ with α and β as well as comparisons of $\mathcal{L}_3$ with $\mathcal{L}_1^{\beta}$ and $\mathcal{L}_2^{\alpha}$ are significant ($\rho < 0.05$). Additionally, PC means for $\mathcal{L}_3$ in Figure 4.7 indicate that the timbre quality remains comparatively constant across all frequency bands with a slight dip around 400Hz (Mel bands ≈ 15) for the PC(β, Υ). The same dip can be seen in other comparisons of PC(β, Υ) which can be a result of a particularly complex drum timbre in inputs β that is not present in α. The overall stability of reconstruction quality across all frequency bands demonstrates that the

proposed mashup transformation is capable of mixing timbral characteristics of different drum loops evenly throughout transformation examples.

4.3.2.3 Augmented Mashup Transformation: $\mathcal{L}_4$ and $\mathcal{L}_5$

Timbral similarities of the augmented mashup demonstrate that this objective can guide the transformation towards the timbral characteristics of the chosen input. The $\mathcal{L}_4^{\alpha}$ and $\mathcal{L}_5^{\beta}$ successfully generate mashups which are closer to the corresponding *content* loss terms ℓ_C^A and ℓ_C^B , respectively. Similarly to objective $\mathcal{L}_3$ the augmented mashup displays a consistent reconstruction quality across all frequency bands. For PC comparisons against inputs β the $\mathcal{L}_5^{\beta}$ objective (purple) has the second highest similarity which has some timbral overlap (e.g., Mel bands ≈ 40) with results of the standard AST objective (orange). High mean values of 0.76 and 0.72 for $\mathcal{L}_5^{\beta}$ SCS and $\mathcal{L}_5^{\beta}$ PC means, respectively, demonstrate a similar trend seen in means from $\mathcal{L}_3$ caused by more timbral variability in the *content* characteristics of inputs β. The comparisons of objectives $\mathcal{L}_4$ and $\mathcal{L}_5$ with timbral similarity means between α and β are significant ($\rho < 0.05$), which shows the augmented mashup transformations created new timbres differing from the baseline comparisons.

4.3.3 Discussion

Controllable Rhythmic Transformation

The proposed mashup and augmented mashup transformation objectives effectively showcase a significant capability to introduce controlled rhythmic transformations. In contrast, standard AST objectives seem limited in this aspect, predominantly preserving rhythmic patterns similar to their inputs. Results from onset detection and spectral similarity further validate the effectiveness of the mashup objectives. These findings highlight the potential of the introduced techniques in transforming rhythmic content in drum loops, setting the stage for a variety of user-controlled musical applications and transformations. Moreover, due to the segmentation stage in the rhythmically constrained AST, the output adheres to the metrical structure of the chosen input.

Differences in Transformation Techniques

The supporting audio examples on the accompanying website[19] offer a comparative analysis, highlighting the advantages of the rhythmically constrained AST system over existing transformation techniques for automated rhythmic transformation of drum recordings. The $\mathcal{L}_1$, $\mathcal{L}_2$, and the proposed $\mathcal{L}_3$, $\mathcal{L}_4$, and $\mathcal{L}_5$ transformation objectives can be audibly compared against other algorithms in the literature (Barry and Kim, 2018; Mital, 2017; Ulyanov and Lebedev, 2016). In contrast to other methods that fail to change the rhythmic patterns, the proposed system consistently showcases its ability to modify rhythmic patterns without sacrificing coherence. These comparisons not only suggest promising potential for the system but also hint at its unique contributions to the evolving landscape of audio style transfer.

Gram Matrix Feature Representation

The proposed rhythmically constrained transformation differs in that it is capable of generating new rhythmic patterns from both inputs while preserving the beat pattern of the chosen recording. Challenges faced by all AST transformations are the loss of phase information and the addition of noise, potentially due to the high-level representation of the *style* loss (i.e., Gram matrix). As in other AST methods (Barry and Kim, 2018; Grinstein et al., 2017; Mital, 2017; Perez et al., 2017; Ulyanov and Lebedev, 2016; Verma and Smith, 2018), the Gram matrix is here used as a representation for *style*, yet it remains questionable whether this feature representation is suitable for transformations based on high-level musical information. Briot and Pachet (2017) suggest that this technique presents challenges for audio due to anisotropy of the *content* representation. Anisotropy signifies dependence on directions and here it refers to the nature of the audio spectrogram. In this time-frequency representation the dimensions do not correlate together in the same way a pixel would in an image (e.g., see Figure 4.2). A pixel almost always corresponds to one object whereas in music multiple sources overlap (e.g., kick and hi-hat playing at the same time) causing inherent issues when using the Gram matrix to transform local changes in timbres.

4.3.4 Conclusions

Rhythmic and spectral similarity metrics, along with onset detection methodologies, were employed to evaluate the three proposed transformation objectives. The rhythmic similarity evaluations indicate that the proposed transformation objectives consistently generate novel rhythmic patterns that deviate from the source recordings. In contrast, the standard AST objective did not demonstrate rhythmic transformation of the input audio examples. When the *content* loss was applied, the transformed outputs from objectives $\mathcal{L}_1$ and $\mathcal{L}_2$ mirrored the rhythmic patterns of their respective inputs. This led to no noticeable divergence from the baseline, as seen in the comparative study with regard to α and β. The $\mathcal{L}_3^{\alpha}$ and $\mathcal{L}_3^{\beta}$ mashup objectives transform rhythmic patterns, resulting in performances distinct from both source inputs, aligning with the desired outcomes of rhythmic transformation. Furthermore, the augmented mashup demonstrates the capability of the $\mathcal{L}_3$ objective to adjust rhythmic patterns towards a chosen input in terms of timbre, while also presenting innovative rhythmic transformations distinct from baseline benchmarks. The timbral evaluation highlighted the proficiency of the proposed objectives in generating drum loops with new timbral characteristics. The timbral similarity evaluations showed a substantial effect of the proposed transformation objectives in creating drum loops with new timbral characteristics as well as successfully directing the outputs towards the characteristics of the chosen recording. While the results demonstrate that the AST transformation is sensitive to more complex timbres, the mashup objectives are still able to preserve a constant reconstruction quality across frequencies and guide the output to become significantly different from the standard AST objective. Both rhythmic cosine similarity and onset detection metrics indicate that the *content* loss primarily controls the augmented mashup transformations. Moreover, by adjusting the proportion parameters for

each transformation objective, as shown in Equation (4.8), a further user-defined rhythmic controllability can be introduced.

4.4 Chapter Summary

In this chapter, a rhythmically constrained audio style transfer (AST) system was developed, employing a time-domain approach that operates on beat-length segments of input music signals. The chapter presented evaluations of a combination of both standard and novel objectives and the role they play in shaping the outcome. The rhythmically constrained AST system consists of three stages: segmentation, in which audio inputs are split into beat-length segments; feature representation, where a neural network is used to create *content* and *style* representations for the chosen inputs; and optimisation, where a new beat-length segment is generated to encapsulate the timbral and rhythmic characteristics from chosen recordings. Results from the standard AST objectives, $\mathcal{L}_1$ and $\mathcal{L}_2$, showcase a rigid adherence to the original rhythmic patterns, mirroring them in their transformations. The outputs Υ for both $\mathcal{L}_1$ and $\mathcal{L}_2$, largely influenced by the *content* losses, ℓ_C^A and ℓ_C^B , remain significantly similar to the baseline rhythmic patterns observed between the inputs α and β. This resulting similarity, along with the listening examples provided on the supporting website,[19] underscores the limitation of the standard AST objectives in facilitating meaningful rhythmic alterations. The mashup transformation objective $\mathcal{L}_3$ successfully produces outputs with new rhythmic patterns. By crafting unique rhythmic patterns that combine features from both inputs, $\mathcal{L}_3$ establishes itself as an effective transformation objective. This results in tangible transformations that produce significantly different rhythmic patterns from the content input, a distinction not observed with the standard AST objective. This is further supported in the onset detection F-measure results, where $\mathcal{L}_3$ transformations demonstrate the generation of new events significantly differing from the baseline comparisons. The augmented mashup transformations, $\mathcal{L}_4$ and $\mathcal{L}_5$, highlight the ability to combine the content information with rhythmic patterns, affording users the capability to guide transformations towards a desired input. This is corroborated by the findings that indicate significant differences in rhythmic similarity between $\mathcal{L}_4^\alpha$ and $\mathcal{L}_4^\beta$, as well as $\mathcal{L}_5^\alpha$ and $\mathcal{L}_5^\beta$. Such controllability suggests that users can influence the rhythmic patterns based on the selected *content* loss. Additionally, by utilising the proportion parameters integrated within the proposed loss formulation, users can fine-tune their audio outputs. This capability allows users to modify the rhythmic patterns of the output to align with their envisioned outcome, further enhancing the versatility of the AST system.

Overall, this chapter has presented a new system for audio style transfer with rhythmic constraints together with two new transformation objectives. Chapter 4 serves as a foundation, highlighting the importance of developing and refining the transformation objectives for precise and desired transformations in rhythmic patterns. The introduced objectives offer a unique perspective on manipulating rhythmic patterns and timbral characteristics of the input drum recordings. While the AST method employed the Gram matrix feature representation, this represents just one of many potential representations used in deep generative models. Exploring other feature representations can provide alternative capabilities

for rhythmic transformation of drum recordings. In Chapter 6, an alternative feature representation is introduced which is designed to capture entire bar-length patterns, encompassing all timbral characteristics of drums. The evaluation techniques presented in this chapter lay the groundwork for methodologies discussed in subsequent chapters. The next chapter explores another neural drum synthesis approach, towards a transformation which relies on translation of input drums of one type to another. To enhance the evaluation of transformations resulting from this method over a diverse range of drum recordings, three existing datasets were employed and two new datasets were also created.

Chapter 5

Drum Translation for Rhythmic and Timbral Transformation

This chapter presents a deep generative system for the synthesis of drum recordings using an autoencoder-based WaveNet model. The system synthesises percussion instruments by learning from an extensive dataset of drum samples, enabling precise control over the rhythmic and timbral characteristics of the generated sounds. This method builds upon the foundations laid in previous chapters and advances the field of neural drum synthesis by adapting the translation network and proposing a novel training strategy for percussion instruments. Distinct from the methods explored in Chapter 4, which focused on rhythmic transformation derived from multiple input recordings, the drum translation system operates on a single input source. It introduces a comprehensive training dataset to facilitate drum translation—the process by which an input file is transformed to emulate different drum types. This presents a granular level of control of a rhythmic and timbral transformation of drum recordings.

The deliberate positioning of this chapter preceding Chapter 6 serves a dual purpose. Firstly, it sets the stage for the integration and synthesis of adversarially trained autoencoders, central to the subsequent chapter's end-to-end system for drum synthesis and rhythmic transformation. Secondly, it plays an essential role in the thesis' development by enabling a shift from an audio style transfer system reliant on input recordings and system parameters to one that users can personalise with their own drum sample collections, thus offering a unique dimension to the creative transformation of percussion sounds.

This chapter is organised as follows: Sections 5.1 and 5.2 introduce the DGM model for drum translation, outlining its components that are adapted to the task of generating individual drum samples of the target type. Section 5.3 presents evaluations that employ automatic drum transcription assessment to gauge the success of drum-to-drum translations and the reconstruction capabilities of the proposed system. Section 5.4 discusses the results, focusing on the effectiveness of the drum translation system. A summary of the chapter is provided in Section 5.5.

5.1 Drum Translation

The aim of the transformation addressed in this Chapter is mapping (or *translation*) of musical audio to drum sounds. The rhythmic structure of an arbitrary audio input can be encoded as a combination of different percussion instruments of the drum kit and transformed such that it sounds as if it were played on different drums. This mode of transformation with DGMs (see Section 3.3) is capable of transforming rhythmic patterns of the inputs in configurations shown in Figure 3.13e and 3.13g. The proposed transformation is performed using autoregressive (AR) DGMs to further relax the memory restrictions of AST with rhythmic constraints discussed in Chapter 4 and facilitate the generation of temporally extended drum sequences.

Audio synthesis in AR models is achieved by learning a distribution that predicts the next audio sample from the previous samples in its receptive field using a series of dilated convolutions. The majority of these systems have been developed to address pitched instruments (Child et al., 2019; Dhariwal et al., 2020; Engel et al., 2017; Mor et al., 2018; Oord et al., 2016a), while relatively few such systems have focused on the rhythmic aspects of such transformations and on the generation of percussive instruments in a *drum-to-drum* translation akin to the task of redrumming or drum replacement (see Section 2.3.3). To that end, a denoising WaveNet autoencoder architecture (Engel et al., 2017) is modified and specialised for drum translation by utilising an unsupervised training strategy of a multi-domain latent space that is trained end-to-end on combinations of drum samples. In this architecture, a single encoder learns a shared latent space for multiple decoders to use during training and audio generation. The size of the architecture is adjusted to learn short-term sounds of drum samples, while maintaining encodings for different drum instruments.

5.2 Method

This approach to drum translation concerns the task of synthesising source audio to corresponding drum sounds. The system is inspired by architecture of Mor et al. (2019), in which music signals can be translated across instruments and styles. While Chapter 4 centred on a rhythmically constrained audio style transfer system, emphasising the modification of rhythmic patterns and timbral characteristics through Gram matrix feature representation, the drum translation approach specialises in the translation of specific drum sounds using an adversarially trained autoencoder network. It transitions from a broader stylistic transformation to a more detailed and precise synthesis of drum types, presenting another avenue for rhythmic transformation. The primary contributions of this approach to drum translation are the simplification of the translation network and the proposition of a novel training strategy designed specifically for percussion instruments.

Figure 5.1 provides an overview of the proposed drum translation system, which is comprised of three stages: (1) Feature representation; (2) WaveNet autoencoder; and (3) Generation. At the core of the system is a WaveNet autoencoder network with a shared encoder and a disentangled latent

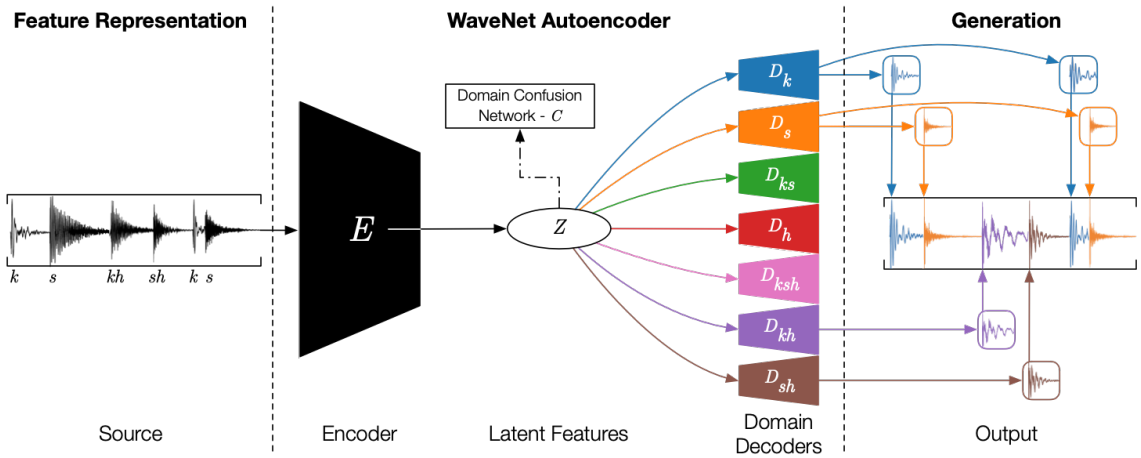

Figure 5.1: Drum translation overview in three stages. Source audio is transformed to output through a single shared autoencoder of domain p specialised on domain decoders D_p, where p represents: kick (k), snare (s), kick and snare (ks), hi-hat (h), kick, snare and hi-hat (ksh), kick and hi-hat (kh) or snare and hi-hat (sh). Colours illustrate pathways between source and corresponding D_p trained to synthesise the target instrument (e.g., orange decoder D_s synthesises snare drums). Solid lines represent information flow during synthesis and dashed-dotted line represents information flow to a domain confusion network present only during training.

space, distributed across each drum domain decoder D_p, where p represents a percussion domain for P total number of domains. Figure 5.1 illustrates seven percussion domains ($P = 7$). These are defined as the kick (`k`), snare (`s`), and hi-hat (`h`) instruments, as well as their combinations such as kick and hi-hat (`kh`). However, the system is adaptable and can be trained with a different number of drum domains. During training, multiple source-target pathways p (one per drum domain illustrated by different colours in Figure 5.1) are encoded by a domain-independent encoder E. The input to the neural network is an audio segment X_p of length T samples ($T = 6000$) representing a waveform of one of the seven drum domains. Each segment is distorted by random pitch modulation to prevent the network from memorising the input signal and provide a semantic encoding.

The employed approach is influenced by autoencoder architecture design choices present in relevant work in neural audio synthesis (Child et al., 2019; Dhariwal et al., 2020; Engel et al., 2017; Mor et al., 2019). Figure 5.2 shows audio input into the encoder E, a fully convolutional network, which outputs a latent space Z that is downsampled using 8-bit μ-law encoding. This encoding method was adopted considering its success in speech and music synthesis tasks, particularly for its efficiency in representing complex audio signals with minimal information loss (Oord et al., 2016a). The latent space is then used to condition a domain confusion network (Ganin et al., 2016). The inclusion of this network is motivated by its capability to provide an adversarial signal to the encoder during training, a strategy which has proven effective in ensuring domain-invariant feature extraction across multiple domains. Subsequently, the latent signal is temporally upsampled to the original audio rate and used to condition a WaveNet decoder D_p. Each decoder uses a softmax activation to predict the probability of the next time step, a decision grounded in literature highlighting the effectiveness of softmax in probability-based predictions for sequential data (Oord et al., 2016a). Once training is finished, the embeddings of all drum domains in the shared latent space can be used to transform source audio from any arbitrary audio domain.

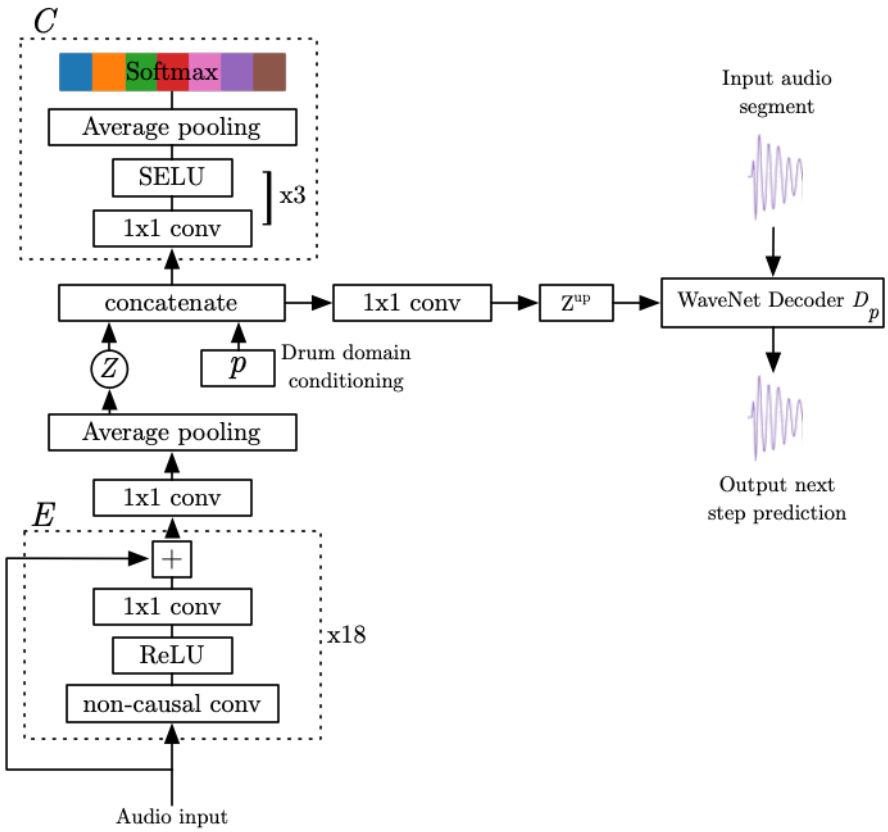

Figure 5.2: Conditional autoencoder architecture for drum translation. Dotted lines present architectures for the encoder E and domain confusion network C used during training with the WaveNet decoder for drum domain p. During training, the decoder D_p uses input audio segment and predicts the next step.

This approach was proposed based on the principle that a shared latent representation can enable more versatile and accurate audio transformations (Mor et al., 2018).

5.2.1 Conditional Autoencoders

A similar dilated convolutional WaveNet encoder architecture as Engel et al. (2017) is adopted and a WaveNet decoder from Mor et al. (2019) to model percussion sounds in the time domain. Dilated convolutions are convolutions with holes that greatly increase the receptive field while significantly reducing the model parameters and computational cost. Dilated convolutions are preferred for audio-related tasks due to their efficiency in managing large receptive fields, essential for capturing the details in audio signals. Moreover, the WaveNet model integrates a residual learning framework, as suggested by He et al. (2016a). This integration, as established in the literature, is aimed at circumventing the vanishing gradient problem commonly associated with deep neural network training and expediting the training process.

The shared encoder E has 18 layers with two blocks of nine residual-layers and a maximum dilation of 512 samples. As in Mor et al. (2019), a residual-layer structure with a ReLU non-linearity is used, a non-causal dilated convolution with an increasing kernel size, a second ReLU, and a 1x1 convolution (i.e., a time-distributed fully connected layer) followed by a residual summation of the activations before

the first ReLU. Unless specified otherwise, 64 channels are used in all hidden layers of the autoencoder architecture. After two blocks, the encoding goes through a 1x1 layer and an average pooling with a kernel size of υ samples ($\upsilon = 400$).

The WaveNet domain decoder D_p is conditioned with a temporally upsampled version of the latent encoding Z^{up} obtained with nearest neighbour interpolation. The conditioning signal adds parameters to the probability distribution so that it depends on variables that describe the audio to be generated instead of only using the previously generated samples. Without conditioners, WaveNet has been shown to mix sequences of speech by repeated phoneme shifting between voices of all speakers used in training of the model. As in Mor et al. (2019), the conditioning goes through a different 1x1 layer for each decoder D_p to ensure that the latent space is domain independent. This reduces source-target pathway memorisation, which is also aided by pitch augmentation. To ensure that only previous samples are used in the generation of the new ones, decoders D_p use dilated causal convolutions together with additional non-linear operations to enable them to learn input audio representations that cannot be captured with just linear operations. Each D_p has two blocks of nine residual-layers, where each layer contains a causal dilated convolution with an increasing kernel size, a gated hyperbolic tangent activation (Oord et al., 2016a) (the main source of non-linearity), a 1x1 convolution followed by the residual summation of the layer input, and a 1x1 convolution layer for skip connections. Encoding Z^{up} is used to condition each residual-layer during training. The skip connections are summed with a ReLU non-linearity activation and passed through a 1x1 convolution layer before a softmax activation layer.

5.2.2 Domain Confusion Network

In order to introduce an adversarial signal to the autoencoder and ensure that the encoding is not domain-specific, a domain confusion network C is implemented following Mor et al. (2019). The network predicts the percussion domain label of the input data based on the latent vectors Z. It uses a single gradient reversal layer defined in Ganin et al. (2016) and three 1D-convolution layers. The gradient reversal layer (GRL) reverses the gradient by multiplying it with a negative scalar λ ($\lambda = -0.01$). The GRL ensures that the feature distributions over the P drum domains are made similar (i.e., as difficult as possible to recognise for the domain classifier C), thus resulting in the domain-independent features. The three 1D-convolutional layers all include SELU non-linearities (Clevert et al., 2015) with 128 channels in all hidden layers. After three layers the output is passed through an average pooling layer to project the vectors to P total number of domains using a softmax layer (see Figure 5.2).

5.2.3 μ-law Quantisation

WaveNet predicts a non-normalised probability distribution from the residual-layers and transforms it into a proper probability distribution by using a softmax function. The authors of the original WaveNet (Oord et al., 2016a) show that softmax distribution tends to be more flexible than other mixture models and can more easily model arbitrary distributions as it makes no assumptions about their shape.

All audio files processed by the model use a sampling rate of 22.05 kHz and are stored as 16-bit integers. To model all possible values per time step, a softmax layer would need to output 2^{16} probabilities. To moderate this high bit-depth resolution of the input audio, a μ-law algorithm (ITU, 1988) is implemented and quantises the data to 2^8 quantisation levels:

$$f(x_p) = sign(x_p)\frac{ln(1+\mu|x_p|)}{ln(1+\mu)}, \tag{5.1}$$

where $\mu = 255$ and $-1 < x_p < 1$. This quantises the high resolution input to 256 possible values causing a loss in audio quality (Oord et al., 2016a); however, it makes the model feasible to train. The inverse of the μ-law can be calculated to decode a quantised audio example as follows:

$$f^{-1}(y_p) = sign(y_p)\frac{(1+\mu)^{|y_p|}-1}{\mu}. \tag{5.2}$$

5.2.4 Data Augmentation

To improve the generalisability of a single encoder during training and to increase the size of the training data, a pitch augmentation approach is implemented (Mor et al., 2019). In popular music production, pitch shifting of individual drum samples is a common processing technique that is used either on all drums or on a subset of drum samples that are layered underneath other sounds to create richer timbres. Instead of augmenting only parts of the input data as in Mor et al. (2019), pitch is modulated across the whole length of each audio segment X_p by a random value between $\pm$6 semitones with LibROSA (McFee et al., 2015). The final representation of each augmented percussion segment used for training is $X_p = \{x_{p,1}, ..., x_{p,T}\}$.

5.2.5 Training

Two training objectives are minimised with regard to an input sample x_p at time step t from the augmented segment X_p. The fist is the domain confusion loss $\mathcal{L}_{dc}$ computed as follows:

$$\mathcal{L}_{dc} = \sum_{p}\sum_{x_p}\ell_{ce}(C(E(x_p)), p). \tag{5.3}$$

This objective applies cross entropy loss ℓ_{ce} to each element of the output Z and the corresponding percussion label p. The second objective is the autoencoder reconstruction loss $\mathcal{L}_{ac}$ computed as:

$$\mathcal{L}_{ac} = \sum_{p}\sum_{x_p}\ell_{ce}(D_p(E(x_p)), x_p). \tag{5.4}$$

The decoder D_p is an AR model conditioned on the output of the shared encoder E. Final loss $\mathcal{L}$ is defined as:

$$\mathcal{L} = \mathcal{L}_{ac} - \lambda \mathcal{L}_{dc}, \tag{5.5}$$

where λ is a scaling factor for $\mathcal{L}_{dc}$ described in Section 5.2.2.

An Adam optimiser with the initial learning rate of 0.001, and a decay factor of 0.98 is used. The model is trained for 10 epochs and 50,000 iterations in total, where each iteration takes a random mini-batch of 8 randomly pitch shifted X_p segments. All weights in the network are initialised using Xavier initialisation (Glorot and Bengio, 2010).

5.2.6 Generation

During the transformation of an unaugmented audio sample y from any source domain, the autoencoder of domain p with its corresponding D_p is used to output the new sample $\hat{y}$. The input y is quantised using Equation (5.1) and then processed as follows:

$$\hat{y} = D_p(E(y)). \tag{5.6}$$

The output $\hat{y}$ is decoded using the inverse μ-law (see Equation (5.2)). The above generation procedure is also referred to as the inference stage of DGMs and requires both the trained decoder and encoder networks. During inference, the average time to generate 1 second of audio with a sampling rate of 22.05 kHz is 30 minutes when using an NVIDIA Tesla M40 computing processor.

5.3 Evaluation

To determine the efficacy of the drum translation approach, which focuses on translating specific drum sounds through a trained autoencoder network, two evaluations are undertaken. The first evaluation employs an automatic drum transcription assessment to gauge the success of drum-to-drum translation, specifically for kick and snare drums. For this evaluation, the system is trained exclusively on kick, snare, and hi-hat drum domains. In the second evaluation, the quality of timbral and pitch translation reconstruction as well as rhythmic similarity was assessed using a system trained on seven drum domains, as outlined in Figure 5.1.

5.3.1 Drum Sample Dataset

The dataset consists of short drum samples (i.e., isolated drum events) of different instruments from the common drum kit. The dataset must consist of segmented samples to ensure control over the drum types (i.e., domains p) in each training mini-batch as well as to exclude the influence of acoustics and bleed from other instruments commonly present in recordings of a full drum kit. The system is trained

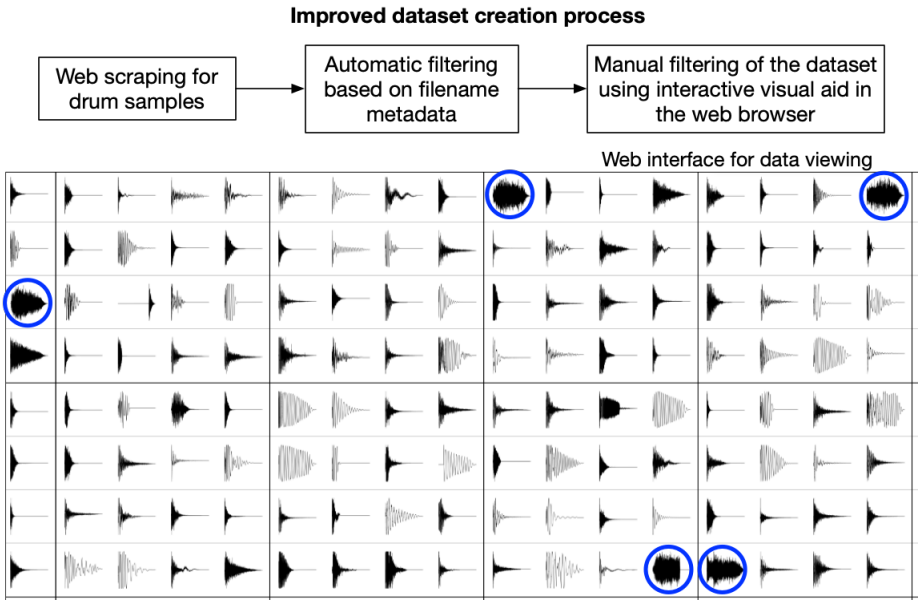

Figure 5.3: Process for creation of the drum sample dataset (top) and a capture of a web interface (bottom) for data viewing used for removing noisy and erroneous audio examples (i.e., blue circles). A demo of the web interface used for fine-tuning of the dataset can be viewed on `https://tdsdne.vercel.app/`.

using raw audio waveforms as input features and labels that heavily rely on the accurate representation of each drum type.

Dataset Creation

The top of Figure 5.3 outlines the creation process for the drum sample dataset. Drum sample libraries were sourced from sample packs included in a range of monthly online magazines such as Future Music, Computer Music and Estrada i Studio.[20,21,22] Additionally, the majority of drum recordings were acquired in the process of web scraping from collaborative repositories of audio samples (e.g., FreeSound, Loopmasters and Splice) as well as from a variety of different sample libraries included in the Ableton Live 10 Suite software.[17]. The collected audio files were then filtered using a rule-based approach relying on file and directory names coinciding with the three primary instruments (i.e., kick, snare and hi-hat). Files which did not include the correct instrument name or were not inside a directory with a desired name were removed. The final stage involved manual filtering of noisy data which could negatively affect the training of the drum translation system. All drum samples are reduced to mono 16-bit WAV files and downsampled from 44.1 kHz to 22.05 kHz.

Manual fine-tuning of large datasets is a laborious task which sometimes cannot be circumvented or replaced with automated methods. While listening to every audio example in the dataset would be time consuming and potentially error prone, a visually aided approach, which is both efficient and easy to use, can significantly speed up the data filtering process. A new approach is proposed inspired by visualisation solutions implemented using GANs for systems which generate non-existing images, also

[20] `https://futuremusic.com/`
[21] `http://computermusic.co.uk/`
[22] `https://estradaistudio.pl/`

known as This X Does Not Exist, where X could refer to anything from an object to a human person.[23] The implemented approach uses a static website template from Gwern (2020). On the website, a large number of audio examples in the dataset can be viewed simultaneously as shown in the bottom of Figure 5.3. All audio waveforms are first saved as images and later loaded using a static website. Upon visual inspection, filenames of noisy or otherwise faulty data samples can be identified and removed. Two experts conducted a comprehensive manual filtering of the dataset using the illustrated web interface, complemented by random listening checks and cross-examining of all examples for each drum type. Furthermore, an external reviewer, uninvolved in the initial filtering stage, provided an additional layer of scrutiny by both visually and aurally inspecting the dataset.

Dataset Details

The dataset of drum samples comprises 9000 examples with 3000 recordings per kick, snare and hi-hat instrument types. All audio files have a constant length T ($T = 16384$) audio samples (i.e., 0.37s), which represents the nearest power of two from the original mean sample length of 18234 samples (i.e., 0.41s). Each audio example is trimmed, with a linear fade applied to the last 1000 samples and zero-padded to a constant length. The constant length requirement is dictated by the symmetric autoencoder architecture. To ensure that each drum domain is accurately represented by the model, silence segments are removed from the beginning of each audio recording. Subsequently, a short linear fade is applied at the start of each file to guarantee that all files begin and end with an amplitude of 0. To mitigate the low resolution at ranges near ± 1 that results from the μ-law encoding stage, the amplitudes of all audio segments are randomly scaled between 0.5 and 0.6. The dataset was introduced in work on neural drum synthesis by Drysdale et al. (2020) and is a contribution of this thesis.

Dataset Training Configurations

To facilitate the evaluation of different drum domains, the system is trained using two configurations of drum sample dataset. The first configuration includes the entire dataset with 3000 recordings per kick, snare and hi-hat instrument types. The first configuration is referred to as $P3$. The second configuration, referred to as $P7$, uses a limited number of 1000 recordings from each of the sever instrument type. This configuration was designed to preserve compute requirements comparable to the first configuration, as they pose significant computational challenges. It also facilitates the generation of the additional four drum domains, which are created as combinations of the three instrument types illustrated in Figure 5.1.

5.3.2 Test Data

'To assess the ability of the system to perform rhythmic and timbral transformations, three distinct test sets, namely S-AST, S-DT20 and S-DT70 were designed. The test sets were created to span a diverse range of drum loop complexities, from basic to intricate, ensuring a comprehensive system evaluation. The S-AST and S-DT20 subsets comprise the initial two beat-length segments from the drum loop

[23] `https://thisxdoesnotexist.com/`

Subsets	Description	Onsets
S-AST	kick drums	30
	snare drums	23
S-DT20	kick drums	20
	snare drums	25
	hi-hats	14
S-DT70	kick drums	10
	snare drums	10
	hi-hats	10
	kicks, hi-hats	10
	snares, hi-hats	10
	kicks, snares	10
	kicks, snares, hi-hats	10

Table 5.1: Instrumentation and drum event onset counts in S-AST, S-DT20 and S-DT70 test sets.

datasets introduced in Sections 4.2.1 and 5.3.1. These segments are accompanied by onset and drum label annotations. Additionally, S-DT70 offers less complex drum tracks, with isolated drum domain samples per track. Table 5.1 summarises the instrumentation and onset counts for each drum type.

S-AST

The first subset, termed S-AST, consists of 15 drum loops with only kick and snare drums. It represents an easier translation material with no overlapping drum sounds and non-complex rhythmic patterns as illustrated in Figure 4.3 in Section 4.2.1. The drum loops are in mono WAV format and are resampled from 44.1 kHz to 22.05 kHz with 16-bit resolution.

S-DT20

The second subset, termed S-DT20, has 20 drum loops with the addition of hi-hats and instrument overlap which can be more challenging for the transformations under evaluation. The simpler dataset S-AST and the more complex S-DT20 have a comparable number of onsets with drum labels and onset times extracted from MIDI and cross-examined by two expert annotators. To evaluate the capability of the system to generate the three primary instruments in different music material, the combined drum labels are reduced to their primary drum domain (e.g., `kh` label is changed to `k` and `sh` is changed to `s`). This reduction of overlapping labels as well as varying tempi of S-DT20 contributes to a slightly different number of drum events from S-AST.

The S-DT20 comprises drum loops compiled using the Apple Logic Pro sample library.[16] The motivation for creation of a new dataset was to introduce more drum types for evaluation as well as introduce more complexity through varied tempi of the drum styles. Due to the increased complexity of the data as well as the compute cost required for this transformation (see Section 5.2.6), the dataset has been reduced to from 30 to 20 loops in contrast to the dataset in Chapter 4. The drum loops are chosen to reflect a variety of different drum patterns and styles, with multiple domains reflected in each loop. The drum loops have a mean duration of 3.5s and tempo ranges from 100 BPM to 170 BPM. All loops are in the mono WAV format and are resampled from 44.1 kHz to 22.05 kHz with 16-bit

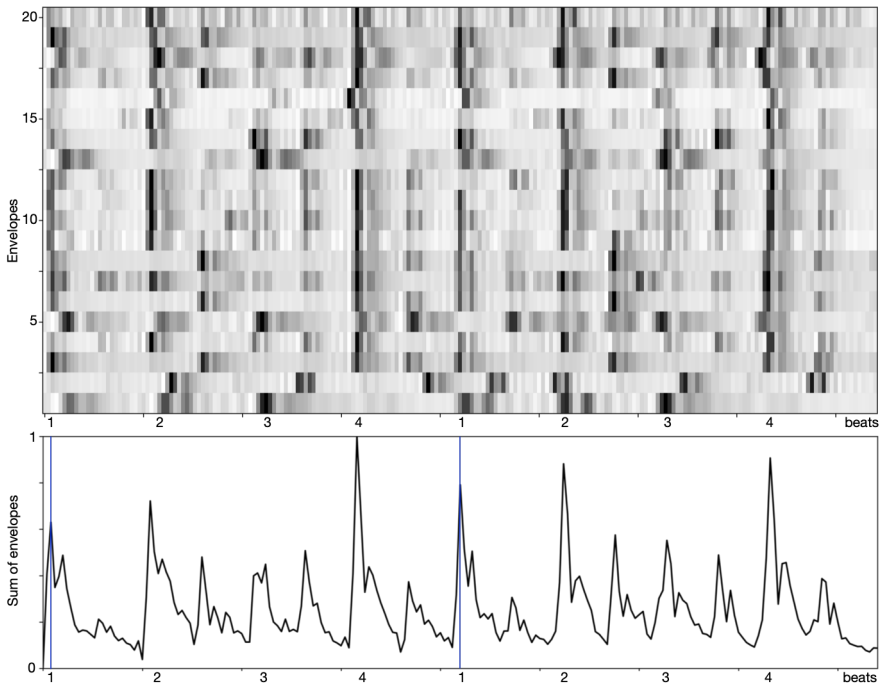

Figure 5.4: Patterns represented as rhythmic envelopes from 20 drum loops in the S-DT20 dataset (top). Sums of rhythmic envelopes computed across the dataset (bottom), where bar boundaries (i.e., 2 bars) are represented with blue vertical lines.

resolution. The dataset is available on the supporting website.[24] The rhythmic patterns in the dataset are portrayed in Figure 5.4. STFT parameters use a w-length Hann window ($w = 2048$) with a hop size of $\frac{w}{4}$ for computation of all rhythmic envelopes (see Equation (4.16)). Due to different tempi present in the drum recordings, every rhythmic envelope was resampled to the same length for the purpose of visualisation. The visualised rhythmic patterns are more complex as compared to the dataset presented in Chapter 4. This is due the addition of the hi-hat cymbal as well as an added complexity stemming from more varied tempi and drum timbres. The larger overlap of different drums (e.g., kick and hi-hat playing together) and resampling result in a more noisy sum of envelopes at the bottom of Figure 5.4. Similar to the dataset from Chapter 4, the summed representation shows a high presence of events (i.e., kicks) on downbeats in most examples as well as a strong presence of events (e.g., snares) on beats two and four, which is characteristic in Western popular music. Additionally, several patterns have less events and their representations appear to be more smeared (e.g., in envelopes 1, 13 and 16). The smearing artifact is introduced in the resampling process and can be further aggravated by the length differences of the original loops (e.g., 1 bar-length pattern compared to 2 bars).

S-DT70

The S-DT70 dataset consists of seven audio tracks, each comprising ten drum samples with distinct timbres sourced from the Logic Pro sample library.[16] Table 5.1 outlines the seven drum domains. Each track features individual percussion instruments, as well as combinations thereof, playing quarter notes at a tempo of 120 BPM.

[24] https://maciek-tomczak.github.io/maciek.github.io/Drum-Translation-for-Timbral-and-Rhythmic-Transformation

5.3.3 Evaluation Metrics

Automatic Drum Transcription

While the rhythmic cosine similarity reconstruction metric reveals differences in rhythmic patterns of the transformed signals, it cannot differentiate between each individual event type present within the rhythmic envelope. To mitigate this, an objective drum classification approach is proposed using a state-of-the-art supervised automatic drum transcription (ADT) model (i.e., lstmpB[25]) (Southall et al., 2017). The standard rates of true positive (TP) and false negative (FN) onsets are used as evaluation metrics. Drum event onset candidates are accepted as true positives if they fall within 50ms of the ground truth annotations while the remaining missed detections represent false negatives. In case of multiple detections within the tolerance window, only the first onset and the corresponding label is accepted and the remaining are discarded. The onsets are reported as percentages of true positives (i.e., the proportion of correctly identified onsets to the total onsets in the target transformation) in each test set with the remaining percentage indicating the missed false negatives. Since the aim is to assess the success of translations between different drum domains and not evaluation of the ADT model, the false positive rates (i.e., detections outside the tolerance window) are not reported. The different translation scenarios include translations to the same drum domain (e.g., `k` to `k`) or to another domain (e.g., `k` to `s`). Prior to automatic drum transcription, all evaluation audio files are converted to 44.1 kHz sampling rate to facilitate the requirements of the ADT model. Additionally, due to significant differences in complexity between the test sets, the ADT evaluations using the improved training set are performed for the same domain translations of the three primary instruments across both S-AST and S-DT20, while translations to opposite `k` and `s` domains are performed using the S-AST test set.

Reconstruction Metrics

The timbral and pitch translation reconstruction quality of each drum domain is evaluated using the Pearson correlation (PC) coefficients comparing the source audio with the output generations. The evaluations utilise two different spectrogram types which can provide an objective measure of reconstruction quality with respect to different aspects of the signal. The first is the perceptually motivated Mel scale filtered spectrogram (see Equation (4.15)) and the second is the Constant-Q transform (CQT). The Mel scale provides a better frequency resolution in the lower frequencies than the higher frequencies, and is an appropriate choice for evaluation of non-pitched instruments such as drums. On the other hand, the CQT creates logarithmically-spaced frequency range over musical octaves and has been previously used in evaluation of DGMs for generation of isolated pitched instrument audio samples (Kim et al., 2019). The CQT and Mel representations are used for comparability with other studies. Following Schörkhuber and Klapuri (2010), the CQT(k, n) for an audio signal $x(n)$ is defined as:

$$CQT(k,n) = \sum_{j=n-\lfloor N_k/2 \rfloor}^{n+\lfloor N_k/2 \rfloor} x(j) a_k^*(j - n + N_k/2), \tag{5.7}$$

[25]The algorithm implementation is publicly available as a Python library on `https://github.com/CarlSouthall/ADTLib`

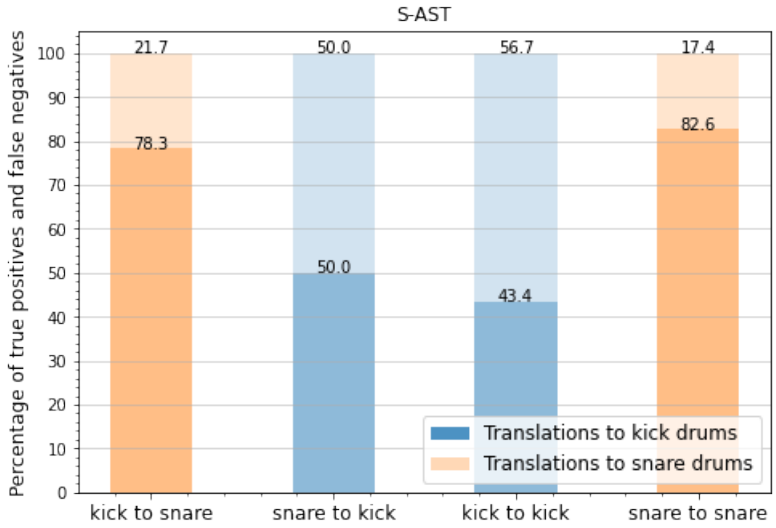

Figure 5.5: Percentages of true positive onsets (i.e., those in a dark shade) and false negative onsets (i.e., those in a light shade) as detected by an automatic drum transcription (ADT) system are presented. The results reflect the ADT's performance in translating between kick (blue) and snare (orange) drum domains using the $P3$ training configuration, tested on recordings from the S-AST subset. A successful transformation from kick to snare is indicated by low automatic detections for kicks and high for snares, whereas a successful snare to kick transformation would be indicated by the opposite.

where a_k^* is the complex conjugate of the basis function $a_k(n)$ calculated as:

$$a_k(n) = \frac{1}{N_k}\omega(\frac{n}{N_k})\exp[-i2\pi n\frac{f_k}{f_s}]. \tag{5.8}$$

The CQT frequency bins and the Hann window function are denoted by k and ω, respectively, and $\lfloor\cdot\rfloor$ denotes rounding towards negative infinity. The center frequencies of the equal tempered scale are calculated as $f_k = f_1 2^{\frac{k-1}{12}}$ for 12 semitone bins spacing per octave, where f_1 is the centre frequency of the lowest-frequency bin and f_s is the sampling rate of the signal. In the following reconstruction evaluations, the PC coefficients are visualised over a frequency range of 32–10548 Hz and calculated using 101-bin log-magnitude CQT with 12 bins per octave starting from C1 (≈32.70Hz) and hop size of 512 using LibROSA (McFee et al., 2015). The PC coefficients are then calculated as follows. For each domain, the mean PCs are taken for ten different source and output log-magnitude CQT spectrograms. The correlation values across all frequencies are smoothed using a median filter. High Pearson correlations indicate that translation quality for a particular drum domain is well preserved, whereas lower correlations indicate larger quality degradation from the source drum recording.

Rhythmic reconstruction evaluates the capacity of the proposed system for preservation of rhythmic patterns during transformations based on the rhythmic envelopes computed from the generated audio. The rhythmic reconstruction evaluations follow the procedure described in Section 4.2.4.1 using the drum loop dataset described in Section 5.3.1. The events in each drum loop are manually labelled, then translated into an output drum loop where domains correspond to the source. The cosine similarity (see Equation (4.17)) is measured between the rhythmic envelopes of source and output transformation

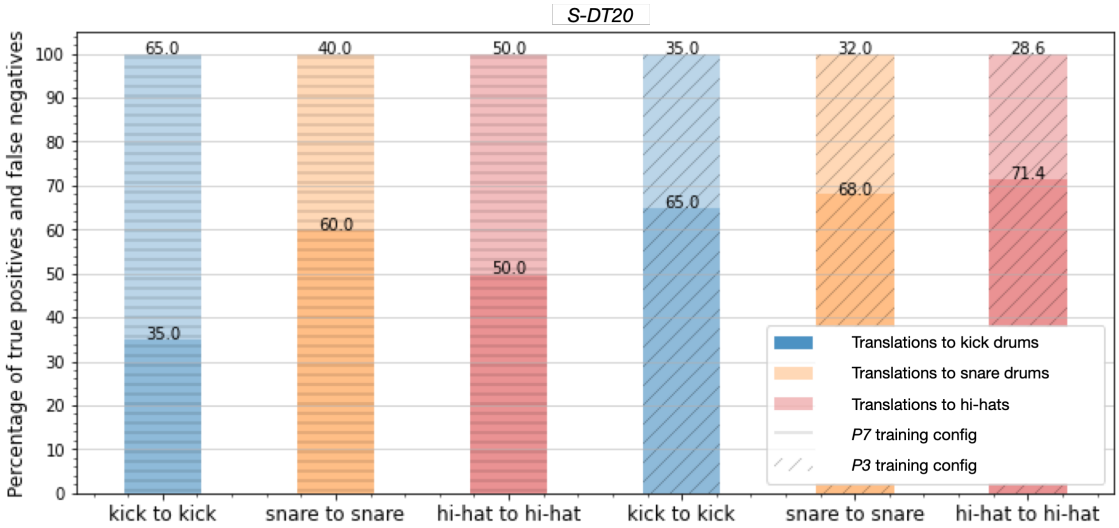

Figure 5.6: Percentages of true positive onsets (i.e., dark shade) and false negative onsets (i.e., light shade) from an automatic drum transcription (ADT) system for the model trained using configurations $P7$ (i.e., horizontal lines) and $P3$ (i.e., diagonal lines). From left to right, the first three diagrams correspond to kicks in blue, snares in orange, and hi-hats in red.

pairs. The rhythmic pattern reconstruction is measured with cosine similarity calculated on the standard spectral difference envelopes (see Equation (4.16)). A high cosine similarity score close to 1 suggests the rhythmic envelopes are almost identical whereas a low score suggests that they are dissimilar.

5.4 Results

5.4.1 Automatic Drum Transcription

S-AST Test Set

Figure 5.5 showcases the ADT system's performance when evaluating the proposed transformations on the S-AST test set. Specifically, the analysis juxtaposes two distinct translation scenarios: (1) inter-domain translations, represented by `k` to `s` and `s` to `k` translations, and (2) intra-domain translations, where the source and target domains remain consistent. A successful inter-domain translation, for example from `k` to `s`, is manifested as a true positive (TP) snare detection by the ADT system on the resultant drum sample. Results are expressed as percentages reflecting the TP onset counts. A scrutiny of the outcomes reveals that the detection efficacy remains relatively consistent across both scenarios, regardless of whether the translation target is the `k` or `s` domain. Furthermore, translations targeting the `s` domain exhibit higher TP detection rates compared to those aimed at the `k` domain.

S-DT Test Set

Figure 5.6 offers a comparative visualisation of TP percentages for intra-drum domain translations assessed on the S-DT test set. The performance metrics are presented for two distinct training configurations: $P7$ (depicted with horizontal lines) and $P3$ (illustrated with diagonal lines), across `k`, `s`, and `h` domains. The results indicate that the $P3$ configuration improves achieves higher of all translated

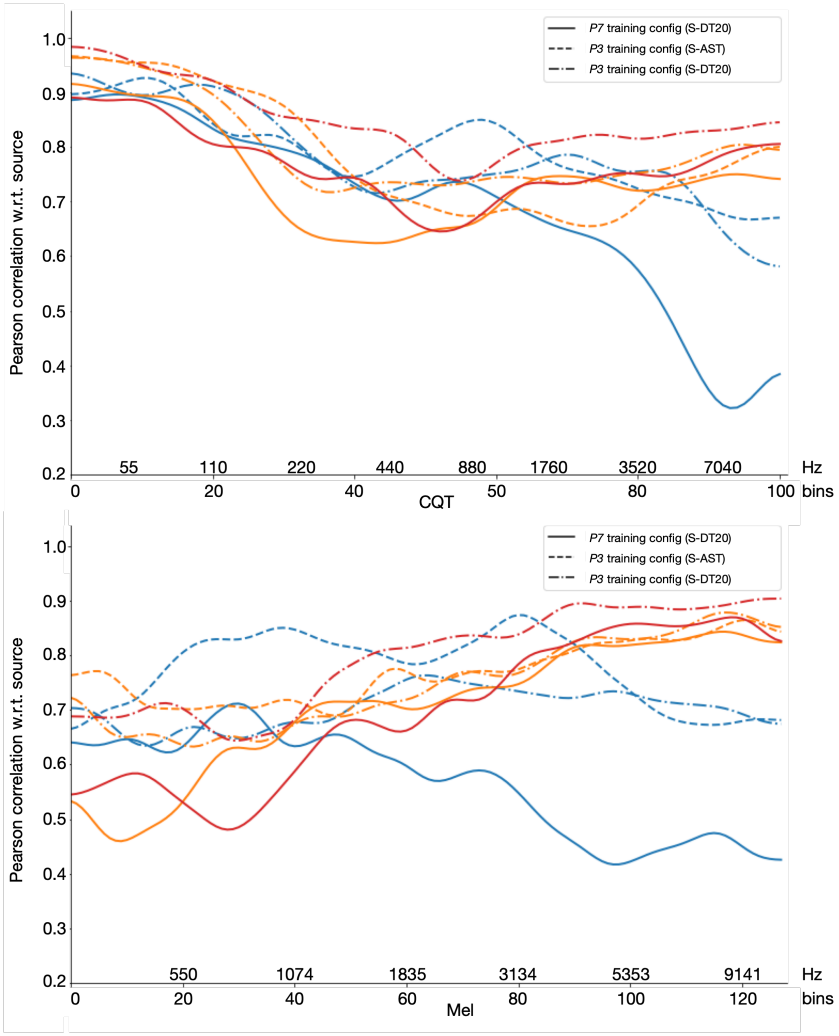

Figure 5.7: Smoothed mean Pearson correlations between the translated and source audio for `k` (blue), `s` (orange) and `h` (red) drum domains.

instruments. The highest improvement of the correctly detected translations can be observed for `k` and `h` domains, whereas it is lesser for `s`. With the new dataset the correct classifications for `k` are improved by 30% and by 21.4% for `h`.

5.4.2 Reconstruction

S-AST and S-DT20 Test Sets

The reconstruction quality is evaluated using PC coefficients, determined as the mean values between the translated drum loops and their corresponding source recordings. For a comprehensive analysis of the generational reconstructions, we present results using both CQT (5.7) and Mel (4.15) spectrogram representations.

Figure 5.7 illustrates the average PC reconstruction outcomes for the `k` (blue), `s` (orange), and `h` (red) domains. The outcomes derived from the $P7$ training set are depicted as solid lines for the S-DT20. In contrast, results obtained from the $P3$ training set for both the S-AST and S-DT20 are denoted

with dashed and dotted-dashed lines, respectively. Notably, the frequency reconstruction for frequencies below 100 Hz is higher across all domains for both spectral representations for configuration $P3$.

The results from S-AST show a higher PC for k around the 60 Hz mark (a typical fundamental frequency for kick drums, as per (Major, 2014)), compared to the S-DT20. This disparity can be attributed to the more straightforward and non-overlapping audio samples in the S-AST test set. Additionally, the S-AST displays elevated PC reconstructions around the mid-range frequencies (approximately 1 kHz) in both spectrogram representations for the k and h domains. In the higher frequency bracket (exceeding 9 kHz), reconstructions show higher results for all instruments in both spectral representations. The most substantial difference is evident in the higher kick reconstructions (blue lines), with modest enhancements observed in the snares and hi-hats (red lines). This trend suggests that, relative to kick drums, snares and hi-hats possess noisier characteristics, making their reconstruction more challenging.

In terms of low frequencies, the CQT spectrogram representations depict a significant degradation, a phenomenon previously observed in pitched instruments (Kim et al., 2019). For non-pitched percussion instruments, the octave spaced CQT representation achieves superior reconstruction of low frequencies and diminished reconstruction of higher frequencies. Conversely, the Mel spectrogram demonstrates lower reconstruction scores for low frequencies and heightened scores for higher frequencies. The primary exception to this trend is observed in the kick drums in S-AST, which achieves higher reconstructions than its counterparts in S-DT20. Drum kit instruments, characterised by complex transients and a lack of regularly spaced overtones, contribute to the Mel spectrogram's subpar reconstruction at low frequencies and enhanced reconstruction at high frequencies. Overall, higher frequency reconstructions are closely matched by both representations, suggesting a similar upper bound of the synthesis quality (e.g., PC$\approx$0.8 for s and h).

The average cosine similarity score across the 20 transformations in S-DT20 dataset training stands at 0.75. This score suggests that, in many instances, the system effectively retains the rhythmic structure during translation from the source to the target domain. The highest rhythmic similarity score of 0.96 for a particular transformation indicates that the majority of the target domains were accurately translated, yielding a rhythmic envelope closely mirroring the input. Conversely, transformations with lower scores often missed parts of the input, leading to inconsistent rhythmic envelopes. The most underperforming transformation registered a rhythmic similarity score of 0.48, attributable to consistent misinterpretations of the kick domain, represented by k. The possible reasons for the various artifacts and behaviours of the synthesised drums are discussed later, using two translation outputs presented in Figure 5.9.

S-DT70 Test Set

Figure 5.8 presents the smoothed Pearson correlations for seven drum domains (e.g., kick k, snare s, hi-hat h, and their combinations). The top diagram showcases PCs computed with CQT spectrograms, while the bottom diagram features PCs derived from Mel spectrograms. For all domains, CQT results suggest a strong preservation of frequency information below 100 Hz. A noticeable decline in correlation for low-mid frequencies (220–1760 Hz) is observed across all domains. Domain k (blue curve) stands out

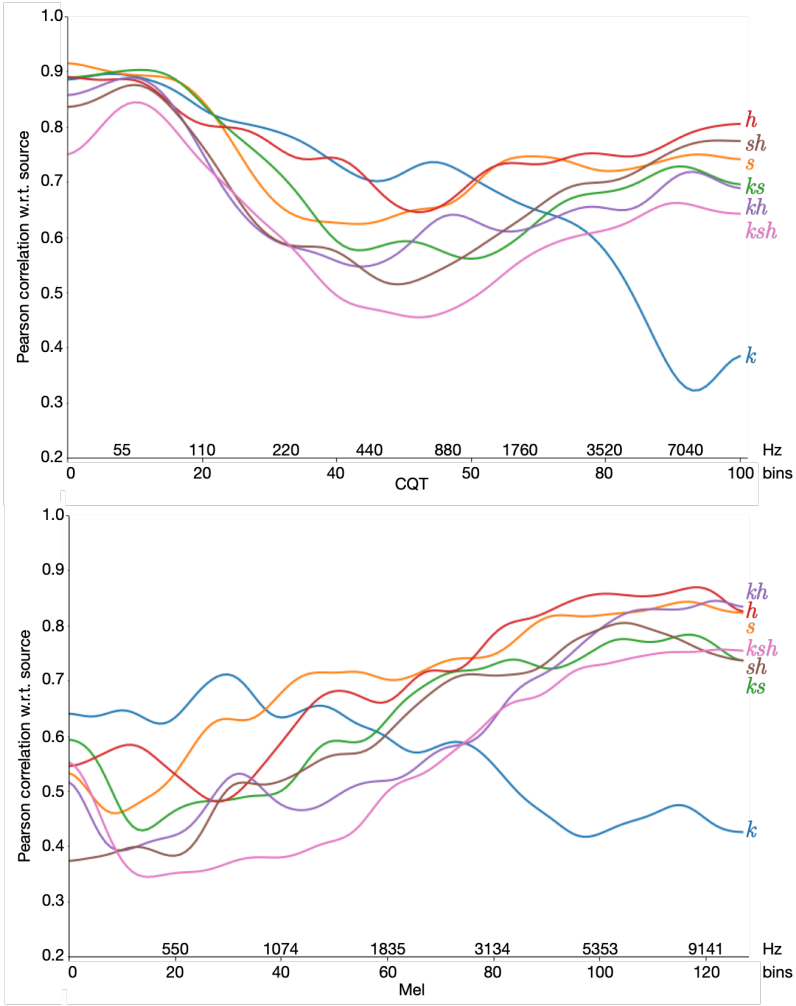

Figure 5.8: Smoothed mean Pearson correlations between the translated and source audio for all drum domains in S-DT20 for system trained with configuration $P7$.

with a higher correlation in these frequencies, but experiences a significant roll-off for high frequencies above 1760 Hz, likely due to model-introduced noise. Conversely, the Mel spectrogram results reveal notably lower reconstructions for frequencies under 3 kHz. Domains `s` and `h` exhibit the highest reconstructions across all frequencies with CQT representation, yet score lower in the low-frequency regions with Mel spectrograms. Domain `ksh` (pink curve) embodies simultaneous kick, snare, and hi-hat playbacks and records the lowest Pearson correlation across all frequency bands. This suggests the heightened complexity of simultaneously playing three instruments poses reconstruction challenges. This trend is also observed, though to a lesser extent, in combined drum domains (i.e., `ks`, `kh`, and `sh`) when juxtaposed against individual `s` and `h` domains. Additionally, drum translations and additional examples are accessible on the affiliated website.[24] As can be heard from many of the translations, the system is capable of generating samples indicative of the intended target domains. Nevertheless, considerable noise surfaces during transformations.

The mean cosine similarity score across the 20 transformations in the rhythmic similarity experiment is 0.75. This indicates that in most cases, the proposed system is capable of preserving rhythmic structure

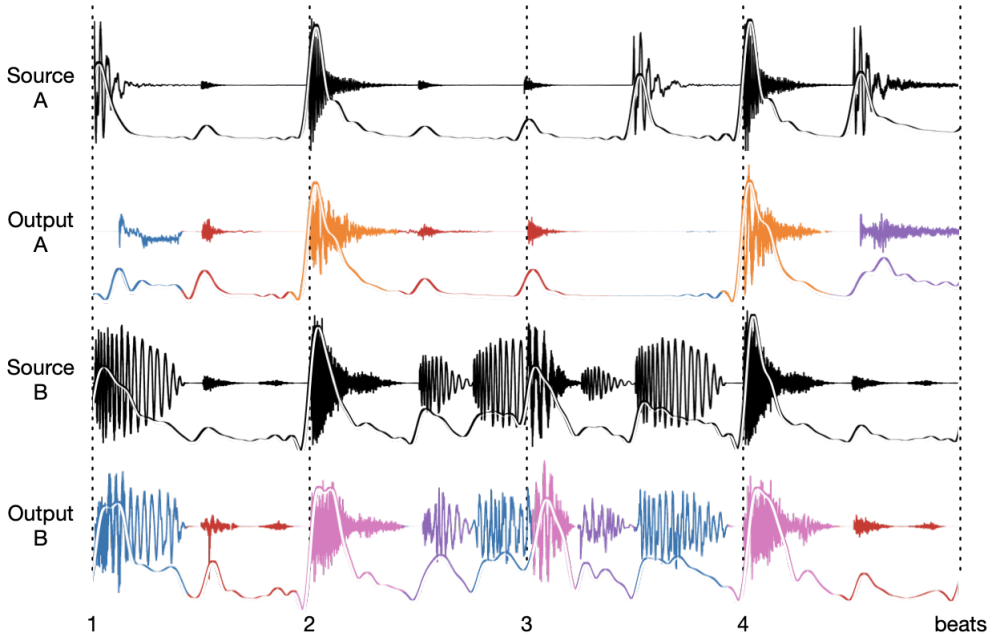

Figure 5.9: Example translations generated from two sources (A and B), with spectral difference functions as solid lines over each waveform. Output colours correspond to target drum domains (e.g., blue represents kick drum translations).

when translating from source to target domain. The highest rhythmic similarity score is 0.96, which suggests that for this transformation the majority of the target domains were successfully translated resulting in a rhythmic envelope that is almost identical to the input. Transformations receiving low similarity scores failed to translate parts of the input resulting in dissimilar rhythmic envelopes. The lowest performing transformation has a rhythmic similarity score of 0.48 due to a frequent failure in properly translating kick domain `k`. The possible reasons for different artifacts and behaviours of the synthesised audio are discussed using two translation outputs presented in Figure 5.9.

5.4.3 Discussion

To delve deeper into the challenges faced in translated outputs, Figure 5.9 plots two sets of source and output waveforms from the complex S-DT20 dataset of drum loops. The cosine similarity for translation A is 0.79, implying certain rhythmic disparities. Upon observing the first beat of output example A, one notes that the waveform of the source kick drum appears distorted. In the first beat of the output example A, it can be seen that the waveform of the source kick drum was unnaturally smeared, whereas the source kick drum A from beat 3 was not translated at all. Both these anomalies—the distortion and the omission—highlight instances where the system did not accurately capture the kick domain, leading to numerous unsuccessful transformations and missed detections in Section 5.4. In the case of example B, all drum domains were translated efficiently, reflected in its high rhythmic similarity score of 0.96. It is plausible that the timbral and rhythmic characteristics of source B enhanced the translation, revealing a well trained latent space across various drum domains. For instance, the translated output kick drum from source B in the first beat shows a noisy onset, but it effectively captures the anticipated low frequencies thereafter. It does, however, noticeably modify the source's traits, thereby generating a

distinct kick sound. While the rhythmic reconstruction metric aids in analysing rhythmic alterations in the produced outputs, it does not provide insights into the specific drum types in the signal. Hence a more detailed investigation into drum translation effectiveness was conducted using a new approach based on automatic drum transcription introduced in Section 5.3.3.

The analysis of Figure 5.5 provides crucial insights into the robustness of the translation system under varying translation scenarios. Both inter-domain and intra-domain translations pose unique challenges. The consistency in detection efficacy across these scenarios suggests that the system adapted for percussion instruments has achieved a level of domain invariance. This means that the transformation procedure is capable of retaining essential rhythmic information irrespective of the source or target domain. The observed higher TP detection rates in translations targeting the s domain could be attributed to distinctive characteristics inherent to the snare drum, which might be more pronounced or easier to detect post-translation. The comparative analysis facilitated by Figure 5.6 highlights the efficacy of different training configurations. The relatively lower improvement for the s domain, when juxtaposed with the results from the S-AST test set, further supports the hypothesis that the distinctive features of snare drums make them resilient to variations in training configurations. Such insights could pave the way for tailored training approaches for each drum domain in future research.

The interpretation of the reconstruction quality using PC coefficients, as showcased in Figure 5.7, offers an intricate understanding of how the translation system processes and maintains the quality of translated drum loops when compared with their source recordings. Two key observations emerge. The first being the system's effectiveness at recreating low-frequency content, particularly under the configuration $P3$. This reconstruction ability in the lower frequency band, especially around the 60 Hz mark for the k domain in the S-AST test set, underscores the system's sensitivity to the kick drum's fundamental frequency, which is a desired trait of a DGM system trained for neural drum synthesis. This fidelity to the original, combined with the more simplified nature of the S-AST audio samples, might explain the elevated PC reconstructions observed. The second observation revolves around the disparities in the reconstruction of mid-range and higher frequencies. The increased PC values at mid-range frequencies for the k and h domains signify that the system is adept at capturing the characteristics of these drums' harmonic content. The distinction in the reconstruction quality in the higher frequency bracket further outlines the complexity inherent to different drum types. The prominence of the kick drums in higher frequency reconstructions, when juxtaposed against the snares and hi-hats, lends support to the hypothesis that the inherent noisier characteristics of snares and hi-hats pose intricate challenges in the reconstruction process.

5.4.4 Conclusions

The drum translation system was evaluated using two distinct methodologies. Firstly, an automatic drum transcription assessment was utilised to determine the effectiveness of drum-to-drum translation, with a specific focus on kick and snare drums. This evaluation hinged on the system being rigorously

trained on kick, snare, and hi-hat drum domains. The second evaluation delved deeper, assessing the quality of timbral and pitch translation reconstruction in conjunction with rhythmic similarity. To ensure a comprehensive evaluation, this system was trained across a wider spectrum, encompassing seven drum domains, as illustrated in Figure 5.1. Our results highlight a marked variance in model performance across different drum types. Kick drums demonstrate a distinct improvement, whereas snares and hi-hats do not follow the same trajectory. Several factors might contribute to this distinction. Among them are the inherent challenges of reconstructing low-frequency content and the system's heightened sensitivity to certain characteristics of the training data. Interestingly, kick drums seem more affected by these factors than the naturally noisy signals of snares and hi-hats. Such drawbacks hint at potential architecture-related improvements.

5.5 Chapter Summary

This chapter has introduced a new mode of transformation through drum translation, which explores the rhythmic and timbral capabilities of generative audio synthesis with WaveNet autoencoders. The introduced system serves as a method for redrumming, fostering innovative possibilities in musical composition and rhythmic transformation. In this transformation, an input file of an arbitrary length is transformed such that it sounds as if it were performed by different drum instruments. To further enhance the system's efficacy, a unique method was introduced for the visual fine-tuning of large audio datasets, specifically to identify and eliminate noisy outliers. Moreover a novel evaluation technique based on automatic drum transcription was proposed. The technique aimed to gauge the system's success rate in translating drums both within and across types. The findings revealed that kick drum generations benefited the most from the enhanced training dataset. In contrast, translations of snares and hi-hats displayed marginal improvements, attributed to their inherently intricate and erratic timbral attributes. Building upon the foundational metrics presented in Chapter 4, this chapter extended the reconstruction metrics to offer comprehensive evaluation of the proposed transformation system.

Overall, this chapter unveiled the transformative potential of redrumming. Advancing beyond the rhythmic pattern modifications and timbral adjustments of the rhythmically constrained audio style transfer presented in Chapter 4, the drum translation system explores the creation of specific target drum sounds. Users can customise these sounds with their unique drum sample collections, offering a personalised touch to creative transformations of percussion recordings. The next chapter presents an end-to-end drum synthesis and transformation system, which allows a user to synthesise individual drum instruments, but also extends neural audio synthesis to include the manipulation of rhythmic patterns within bar-length segments of arbitrary percussion recordings.

Chapter 6

Drum Synthesis and Rhythmic Transformation with AAE

This chapter presents a deep generative system for automated rhythmic transformation based on adversarial autoencoders (AAE). In this system for combined drum synthesis and rhythmic transformation—akin to the popular task of redrumming—a user is provided with control to continuously navigate among complex rhythmic possibilities by interpolating through a low-dimensional latent space. This is achieved by integrating Gaussian mixture latent distributions for rhythmic pattern conditioning with state-of-the-art adversarial autoencoders. To train and evaluate the system, a dataset of over 500,000 bars from 5,418 audio tracks from a variety of musical genres is collected and annotated. This dataset serves as both the proving ground and the training field for the system's capabilities. Chapter 4 introduced a rhythmically-constrained audio style transfer system, which required two audio inputs to function. Then, Chapter 5 discussed drum translation, working with individual drum samples within a percussion recording. Analogous concepts are explored in this chapter, enabling the system to work with either a single input or a pair of inputs, at the scale of bar-length rhythmic patterns.

A major contribution of this chapter is the development of a system that does not require tedious discretised note segmentation or rhythmic event selection prior to transformation. A user is given the freedom to manipulate the structure within a bar without reliance on discrete identification of rhythmic boundaries towards a continuous transformation. This is achieved with the proposed framework based on Gaussian mixture adversarial autoencoders (`AAE-GM`) conditioned on rhythmic patterns present in real music recordings. AAEs are neural networks that use adversarial training to learn a compact and informative representation of the training data, and then use this representation to generate new samples. AAEs have been demonstrated to successfully generate latent space which preserves timbral characteristics of different instruments (Bitton et al., 2019) as well as face features in photographs relating to age (Zhang et al., 2017). The authors in Makhzani et al. (2015) observed that variational autoencoders (VAE) are largely limited by the Gaussian prior, and proposed to use AAEs which can be

trained with any distribution by replacing the KL divergence with an adversarial loss imposed on the encoder output.

The main contents of this chapter can be summarised as follows: Section 6.1.1 introduces the method for continuous transformation with the implementations specifications of a unified AAE architecture for drum synthesis and rhythmic transformation presented in Section 6.1.2. Section 6.1.3 proposes the methodology used to find and transform rhythmic patterns. Section 6.2 describes the creation of the new dataset of percussion performances extracted from real music recordings as well as the evaluation methodologies of the transformation systems. In Section 6.3, the system performance is measured in evaluations of the rhythmic pattern organisation in the latent space, as well as an evaluation of the audio reconstruction performance compared with three pre-existing algorithms, and the transformation quality between source and target patterns through latent space interpolation. The development and evaluations of the proposed system would not be possible without the generous support of Masataka Goto and the compute resources of the National Institute of Advanced Industrial Science and Technology (AIST).

6.1 Method

An overview of the proposed method for joint drum synthesis and rhythmic transformation is presented in Figure 6.1. The system is based on adversarial autoencoders introduced in Makhzani et al. (2015) and is inspired by adversarial audio synthesis approaches in Bitton et al. (2019), Donahue et al. (2018), and Engel et al. (2019). To achieve both drum synthesis and rhythmic transformation in a unified architecture, the proposed model originally extends adversarial audio synthesis to include a regularisation based on a Wasserstein GAN adversarial framework for the transformation of rhythmic and timbral qualities of drum recordings. It supports an AAE with gradient penalty and Guassian mixture prior for conditional disentanglement, the ability of the model to separate and manipulate the underlying factors of variation of rhythmic pattern styles in the dataset. Disentanglement is particularly important for synthesis control of rhythmic patterns and drum types and allows for more fine-grained control over the generated audio, enabling the model to synthesise specific rhythmic styles and instruments.

6.1.1 Adversarial Autoencoder

While similar in design to VAE (Kingma and Welling, 2013), adversarial autoencoders (AAE) appropriate the additional discriminator network D_z from GANs, which aims to distinguish between real and synthesised (i.e., fake) samples. Real samples $z^* \sim p(z)$ are sampled from an assumed prior distribution $p(z)$ imposed on the latent variables z, while synthesised samples are generated through the use of an encoder E conditional distribution $q_\phi(z|x)$. The decoder (i.e., generator network G) conditional distribution is denoted by $p_\theta(x|z)$. In practice $p_\theta(x|z)$ and $q_\phi(z|x)$ are parameterised with neural networks and sampling from $q_\phi(z|x)$ is performed using a reparameterisation trick (Kingma and Welling, 2013). Let $p_d(x)$ be the data distribution of data sample x, and $p_g(x)$ be the distribution of data generated by the model. The encoder defines an aggregated posterior distribution $q_\phi(z)$ on the z as in

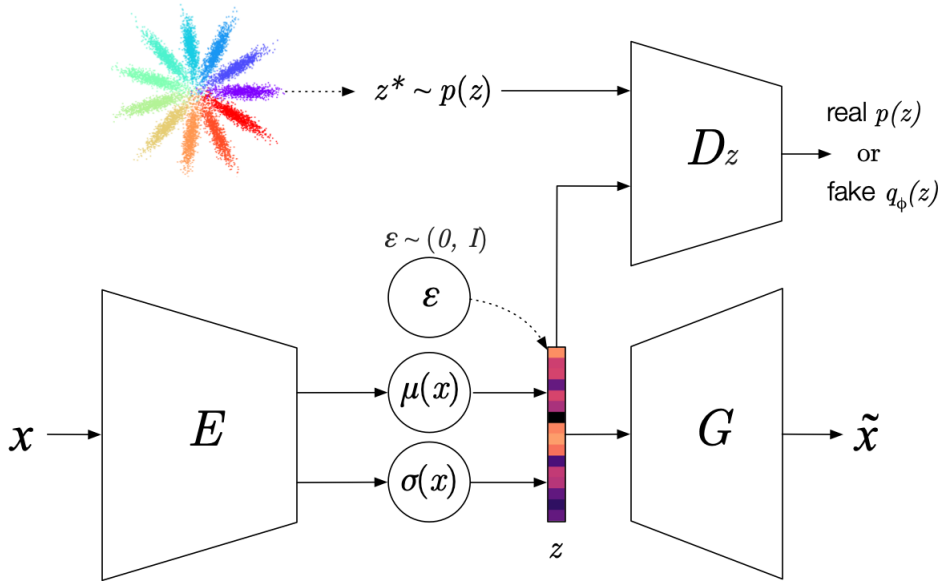

Figure 6.1: Proposed architecture for joint drum synthesis and rhythm transformation. Input data x is mapped onto a latent variable $z \sim q_\phi(z|x)$. Encoder E tries to trick discriminator D_z with artificially generated latent samples and generator G outputs spectrograms $\tilde{x}$. A Gaussian prior distribution $z^* \sim p(z)$ (star) allows the model to juxtapose similar rhythmic patterns in the latent space. Solid lines represent deterministic operations of the network and dashed lines represent stochastic operations.

Equation (3.43). Following the more general formulation for GANs (Nagarajan and Kolter, 2017), the adversarial component of an AAE can be trained as:

$$\min_E \max_{D_z} V(E, D_z) = \mathbb{E}_{z* \sim p(z)}[f(D_z(z^*))] + \mathbb{E}_{x \sim p_d(x)}[f(-D_z(E(x)))], \tag{6.1}$$

where $\mathbb{E}[\cdot]$ denotes expectation and objective V is optimised by alternating parameter updates of encoder E and discriminator D_z in a minimax game characteristic of GAN models. When the concave function $f : \mathbb{R} \to \mathbb{R}$ is set to $f(x) = -\log(1 + \exp(-x))$, the formulation resembles that of the GAN by (Goodfellow et al., 2014). The Wasserstein GAN (WGAN) criterion—introduced in Arjovsky et al. (2017)—can be obtained by setting $f(x) = x$ (see Section 3.2.4).

The parameters of the autoencoder are optimised by the reconstruction error, while the adversarial network guides the encoder to match the imposed prior. Thus, the encoder plays the role of the generator during the adversarial part of training, while the discriminator represents the adversarial network of GANs. After training, decoder G acts as a generative model that maps the imposed prior to the data distribution. Training of an AAE is performed in two phases: (1) the reconstruction phase and (2) the regularisation phase. In the reconstruction phase the reconstruction error of E and G is minimised together and in the regularisation phase, the parameters of the discriminator D_z are first updated by minimising $\mathcal{L}_{D_z} = -V(E, D_z)$ (i.e., to distinguish true samples generated by the prior from the generated codes processed by the autoencoder). The adversarial network then updates the encoder to confuse the discriminator. When combined, the two terms represent $\mathcal{L}_{total}$ as follows:

$$\mathcal{L}_{total} = BCE(p_d, p_g) + \beta \mathcal{L}_{WGAN-GP}, \tag{6.2}$$

where *BCE* denotes binary cross-entropy reconstruction cost between the original data samples $|S|$ (i.e., Mel spectrograms in this study) and their reconstructions $|\hat{S}|$ as:

$$BCE(S, \hat{S}) = -[S \log \hat{S} + (1 - S) \log(1 - \hat{S})]; |S| < 1. \tag{6.3}$$

The second term in Equation (6.2) is the WGAN with gradient penalty (WGAN-GP) loss with weighting β from Gulrajani et al. (2017), proposed as an improved solution to gradient clipping in adversarial training of the discriminator, computed as:

$$\mathcal{L}_{WGAN-GP} = \mathcal{L}_{D_z} + \lambda \mathbb{E}_{\hat{x} \sim p_{\hat{x}}}[(||\nabla_{\hat{x}} D_z(E(x))|| - 1)^2], \tag{6.4}$$

where $\hat{x}$ represents a randomly weighted average between real and generated samples. Following Gulrajani et al. (2017), $\lambda = 10$. During the regularisation phase, this term imposes regularisation on latent variables z and can be trained end-to-end with gradient descent.

6.1.2 Implementation

Details for the proposed adversarial autoencoder with Gaussian mixture prior (`AAE-GM`) are presented in this section. All neural network models are implemented using the TensorFlow Python library.[26]

6.1.2.1 Input Features

Following the approach in Bitton et al. (2019), input audio (16-bit 22.05 kHz mono WAV files) is transformed with short-time Fourier transform (STFT) using a Hann window with a window length of 2048 samples and a hop size of 324 samples to facilitate the desired temporal resolution of the network input. To satisfy the symmetric structure of networks E and G, Mel spectrograms are created from audio inputs with a fixed length of 41344 samples corresponding to a bar segment of 1.87s duration at 128 beats per minute (BPM). Every bar is normalised to this duration by a time-stretching algorithm to ensure fixed bar-length spectral representations. Section 6.2.1 presents further discussion on the chosen input representation of the utilised datasets. The resulting features represent the network input S of size 512 bins by 128 STFT frames. Magnitudes of S are floored to $1\text{e}{-3}$ and log-scaled in [0,1] according to the *BCE* range. Rhythmic pattern styles $\xi = 11$ (i.e., classes as clustered attributes of S) are used in supervised training as labels φ and are defined in Section 6.1.3.

[26] `https://www.tensorflow.org/`

6.1.2.2 Architecture

The architectural details and parameters were inspired by the methodologies outlined in Bitton et al. (2019), Engel et al. (2019), and Donahue et al. (2018). These choices are geared towards optimal performance in generating audio spectrograms with the frequency resolution outlined in Section 6.1.2.1 and have been adapted to facilitate the symmetric autoencoder architecture for modelling of bar-length patterns at 128 BPM. All convolution layers in the proposed architecture are 2D, leveraging square kernels to maintain the aspect ratio of the feature maps. Zero-padding is strategically set to half the kernel size, ensuring the output feature map size is the same as the input size (i.e., *same* padding), which is a common practice in convolutional neural networks to preserve border information (Goodfellow et al., 2014). Feature normalisation in these layers employs the widely-adopted batch normalisation technique introduced by Ioffe and Szegedy (2015), which stabilises learning by reducing internal covariate shift. We utilise leaky rectified linear units (LeakyReLU) with a slope of 0.2, to introduce non-linearity while mitigating the vanishing gradient problem often associated with standard ReLU activations. The deterministic encoder consists of five convolution layers with output channels configured as [16, 32, 64, 128, 256]. This design follows a progressive downsampling strategy that doubles the number of feature detectors after each layer, allowing the network to learn a hierarchy of features from simple to complex (Simonyan and Zisserman, 2014). The chosen kernel size of 7 and stride of 2 have been optimised to reduce the dimensionality effectively while capturing the relevant features in the input spectrograms. The encoder's output is a downsampled representation that is then flattened and passed through a sequence of fully-connected layers forming a bottleneck. This structure, consisting of [2048, 1024, 512] units, is designed to compress the input into a lower-dimensional latent space, as suggested by Hinton and Salakhutdinov (2006) for efficient data encoding. Two fully-connected layers μ and σ are used for sampling z with the reparametrisation trick (Kingma and Welling, 2013) (see Section 3.2.3), thus mapping the input to the latent space $z \in \mathbb{R}^{N_z}$, where $N_z = 64$. This technique is crucial for variational autoencoder models to approximate probability distributions for generative processes. The decoder architecture is a reflection of the encoder with 3 linear layers of output sizes [512, 1024, 2048] and a layer reshaping the vector into 256 feature maps, expanding the compressed representation back to the original input dimension. It employs linear layers followed by reshaping to recover spatial dimensions from the flattened latent representations. To address the issue of upsampling, we use nearest-neighbour interpolation as a computationally efficient alternative to transposed convolutions, which have been shown to produce undesirable checkerboard patterns in the output (Odena et al., 2016). In the context of audio processing, particularly when dealing with spectrograms, checkerboard artifacts can manifest as spurious or unevenly distributed features across the frequency-time bins. These maps are processed through 5 layers with an upsampling factor set to 4 and convolution layers with [128, 64, 32, 16, 1] output channels, kernel sizes [7, 7, 7, 9, 9] and stride 2. The last layer reconstructs the input shape of S (128×512) followed by a sigmoid activation function bounding the output to the *BCE* range. A sigmoid activation function is employed to confine the outputs to the [0,1] range, suitable for binary cross-entropy (BCE) loss computation, a standard approach in autoencoder frameworks for input reconstruction

(Kingma and Welling, 2013). The adversarial discriminator is composed of three fully-connected layers with a substantial number of output channels [2048, 2048, 1]. The architecture concludes with a linear output layer that outputs a scalar score, assessing the degree to which the latent representation matches the target distribution, a concept foundational to adversarial training as discussed by Goodfellow et al. (2014) and Makhzani et al. (2015).

6.1.2.3 Representation of Prior Distribution

The authors in Makhzani et al. (2015) observed that VAEs are largely limited by the Gaussian prior, and thus relaxed this constraint by allowing $p(z)$ to be any distribution by replacing the KL divergence with an adversarial loss imposed on the encoder output. Thus the latent variable z is required to have the same aggregated posterior distribution as the prior $p(z)$. The AAE framework makes it possible to leverage any prior knowledge that may be specific to the studied application. Here, an isotropic Gaussian with 0 mean and I variance is used as a baseline, and compared against a prior distribution that is a mixture of ξ N_z-dimensional Gaussians, where ξ denotes the number of rhythmic pattern styles as defined in Section 6.1.3. This distribution can be depicted with a 2D flower-like shape and allows modelling of a variety of similar rhythmic styles by pushing their latent codes to the centre of the N_z-dimensional distribution. Following the notation by authors in Valenti et al. (2020), the means of ξ Gaussians are placed on a 2D circle as:

$$\mu_i = \left[cos\left(\frac{2\pi i}{\xi}\right), sin\left(\frac{2\pi i}{\xi}\right), 0, ..., 0 \right], \tag{6.5}$$

where μ_i has a total of N_z dimensions. The covariance matrix Σ_i is calculated as:

$$\upsilon_i = \begin{bmatrix} cos(\frac{2\pi i}{\xi}) & sin(\frac{2\pi i}{\xi}), \\ -sin(\frac{2\pi i}{\xi}) & cos(\frac{2\pi i}{\xi}) \end{bmatrix}, \tag{6.6}$$

$$U_i = \begin{bmatrix} \upsilon_i^T & 0 \\ 0 & I \end{bmatrix}, \tag{6.7}$$

$$\Lambda = \begin{bmatrix} a_1 & 0 \\ 0 & \mathrm{diag}(a_2) \end{bmatrix}, \tag{6.8}$$

$$\Sigma_i = U_i \Lambda U_i^{-1}, \tag{6.9}$$

where U_i and Λ are $N_z \times N_z$ matrices. The variance $a_1 = 0.1$ for the radial (i.e., center-to-outer) dimension, and variance $a_2 = 0.001$ for the remaining dimensions. The matrix U_i is used to rotate Λ with respect to the position of the specific Gaussian. Thus, each of the ξ rhythmic pattern styles is associated with a separate Gaussian where patterns that are more similar are still able to be organised closer to each other.

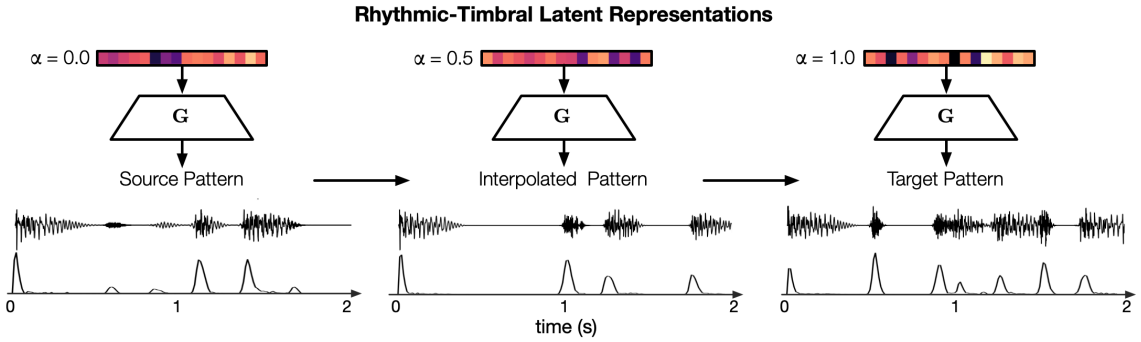

Figure 6.2: Rhythmic transformation of source (left) with intermediate pattern (middle) and resulting output transformation (right). Rhythmic envelopes (bottom) show changes to the rhythmic pattern as the latent code is manipulated via parameter α.

6.1.2.4 Training and Signal Reconstruction

The model is trained using the Adam optimiser (Kingma and Ba, 2014) with an initial learning rate of $1e{-}4$. All model weights use Xavier uniform initialisation (Glorot and Bengio, 2010). The model is trained for around 100000 iterations with a total batch size of 128 for approximately 2 days using 4 Tesla V100 GPUs on AI Bridging Cloud Infrastructure (ABCI) kindly facilitated by the National Institute of Advanced Industrial Science and Technology.[27] During training, the β parameter is gradually increased by 0.1 every 5000 iterations. Mel spectrograms generated by the trained model are approximated back to the linear frequency scale and iteratively inverted with the Griffin-Lim algorithm (Griffin and Lim, 1984) for 150 iterations.

6.1.3 Rhythmic Transformation

An overview of the rhythmic transformation is shown in Figure 6.2. A source recording is reduced to rhythmic-timbral representation output from a deterministic encoder and is passed to the generator together with a target pattern label. This latent code can be used to manipulate metrically relevant positions of drum instruments within a bar with mixing parameter α.

6.1.3.1 Representation of Rhythmic Patterns

Information related to rhythmic patterns is introduced during model training in order to guide the output generations towards particular target patterns. Audio tracks are first separated into a drums component and music parts (e.g., vocals, bass, other) with the Spleeter source separation library (Hennequin et al., 2020).[28] Next, audio tracks are segmented into bars b using the state-of-the-art beat and downbeat tracking algorithm (Böck et al., 2016b) included in the madmom Python library (Böck et al., 2016a).[29] Rhythmic patterns are represented with rhythmic envelope features processed with *LogFiltSpecFlux* from madmom, which performed well in onset detection function comparisons conducted in Böck et al. (2012b), for N ($N = 3$) frequency bands representing *low* (lowpass: 120 Hz), *mid* (bandpass:

[27] https://docs.abci.ai/en/
[28] https://github.com/deezer/spleeter
[29] https://github.com/CPJKU/madmom

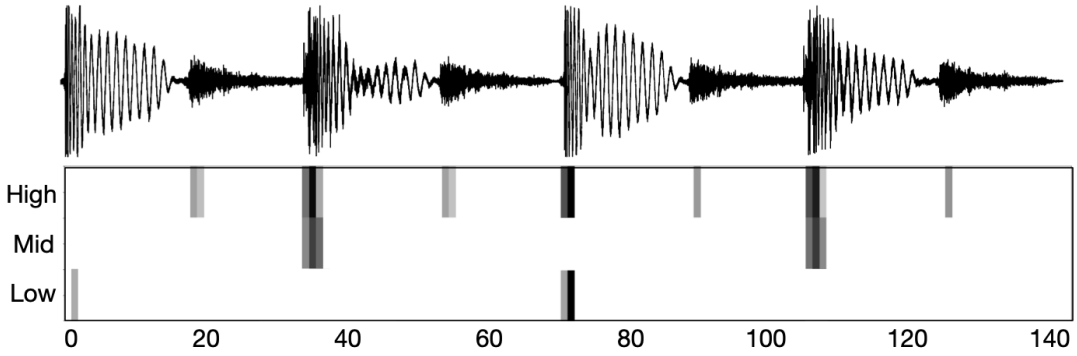

Figure 6.3: Bar-length drum pattern definition using three frequency bands (low, mid and high).

120–2500 Hz) and *high* (highpass: 2500 Hz) parts of drum recordings in each bar b. Following the authors in Dixon et al. (2004) and Hockman et al. (2008), b rhythmic envelopes are resampled to a length of 144 time steps t and normalised to ranges between 0 and 1. The resulting M number of patterns is represented by a template matrix $\tau \in \mathbb{R}^{M \times N \times t}$. Figure 6.3 shows an example bar-length drum recording with the proposed representation of three rhythmic envelopes plotted together.

6.1.3.2 Clustering of Rhythmic Pattern Styles

Building on past research for rhythmic pattern modelling (Dixon et al., 2003; Krebs et al., 2013; Peeters, 2005), an unsupervised clustering strategy via X-means (Pelleg and Moore, 2000) algorithm is proposed in this work. X-means is an unsupervised extension of the popular K-means algorithm, which does not require the predetermined K number of clusters prior to classification. The framework requires specification of the range within which K reasonably lies, and then jointly outputs the number of centroids together with a value for K that scores best by a model selection criterion such as Bayesian information criterion (BIC). All clustering experiments are implemented using PyClustering Python library (Novikov, 2019).

Centroid initialisation is known to influence clustering results in both K- and X-means algorithms, and as such results can be improved through informed initialisation. All experiments in this study incorporate K-means++ initialisation (Arthur and Vassilvitskii, 2006) with prior knowledge of rhythmic patterns extracted from transcriptions of the 50 most frequent kick, snare and hi-hats patterns from over 4.8 million bar-length drum patterns (Mauch and Dixon, 2012).[30] Patterns are resampled to satisfy the structure of rhythmic template matrix $\tau \in \mathbb{R}^{50 \times 3 \times 144}$.

6.1.3.3 Pattern Conditioning and Interpolation

In order to introduce conditioning based on rhythmic pattern styles, each input feature S used during training is assigned a categorical variable taking one of the ξ number of style states found through X-means clustering. During the reconstruction phase, one-hot encoded conditioning vectors for ξ rhythmic styles are concatenated with inputs to the generator G. The basis for a suitable ξ number of rhythmic pattern styles is presented in Section 6.1.3.4. Mixing of two different drum patterns can

[30] `http://isophonics.net/ndrum`

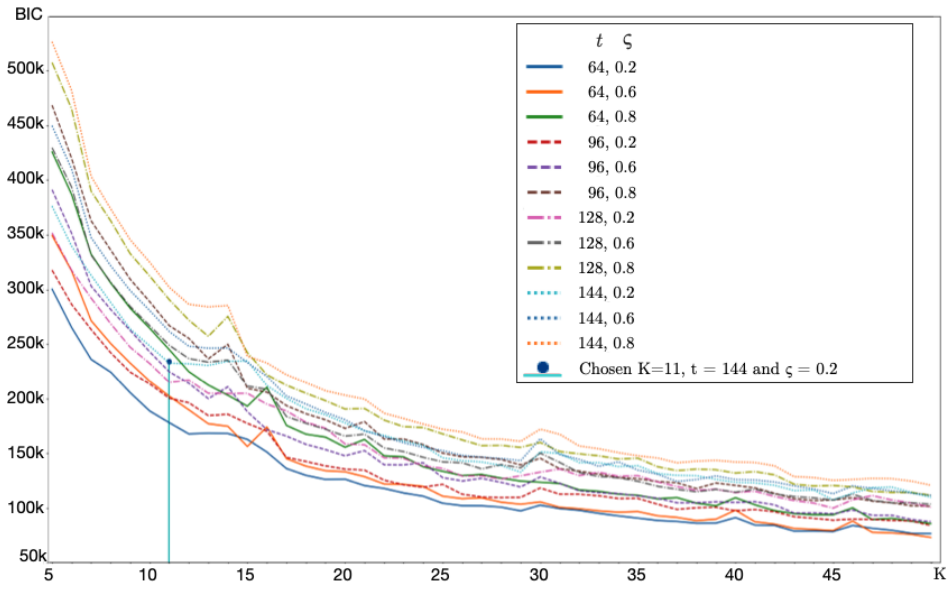

Figure 6.4: Determination of a suitable K with X-Means algorithm. The plot shows sums of mean squared errors for $K = [5 - 50]$ using Bayesian information criterion (BIC).

Rhythmic patterns ξ	3	9	2	4	1	10	7	6	0	5	8
Number of bars	21177	25496	33695	35815	38335	48453	50982	54612	58396	66939	76959

Table 6.1: Numbers of bars ordered from the smallest to the largest (left to right) present in each rhythmic pattern ξ.

be achieved through interpolation transformation in the latent space of the trained model. Since the transformation is continuous, a gradual change can be achieved from the source rhythmic pattern to the target pattern by interpolating the intermediate z values before inputting them into the trained generator. These intermediate latent codes can be produced using a linear interpolation between source and target latent codes such that:

$$\tilde{z} = \alpha z_{target} + (1 - \alpha) z_{source} \tag{6.10}$$

where α is an interval between [0,1]. The interpolated codes $\tilde{z}$ are fed into the generator, which outputs the mixed bar-length drum performances.

6.1.3.4 Pattern Style Definition via X-means

Determination of a suitable number of rhythmic patterns ξ is achieved through the X-means algorithm using BIC scores calculated across $K = [5, 50]$ with a maximum number of clusters set to 100. As in Dixon et al. (2004), a pattern resolution of $t = 144$ is used. Rhythmic envelopes are smoothed for different standard deviations $\varsigma = [0.2, 0.6, 0.8]$ covering a range of 4 timesteps at a time. Convergence was most frequently observed at $K = 11$ with $\varsigma = 0.2$. Hence, ξ consists of 11 patterns ranged from ξ_0 to ξ_{10}. Using the proposed clustering parameter configuration, mean rhythmic pattern representations are calculated for three frequency bands (i.e., high, mid, and low). Figure 6.5 shows the mean rhythmic

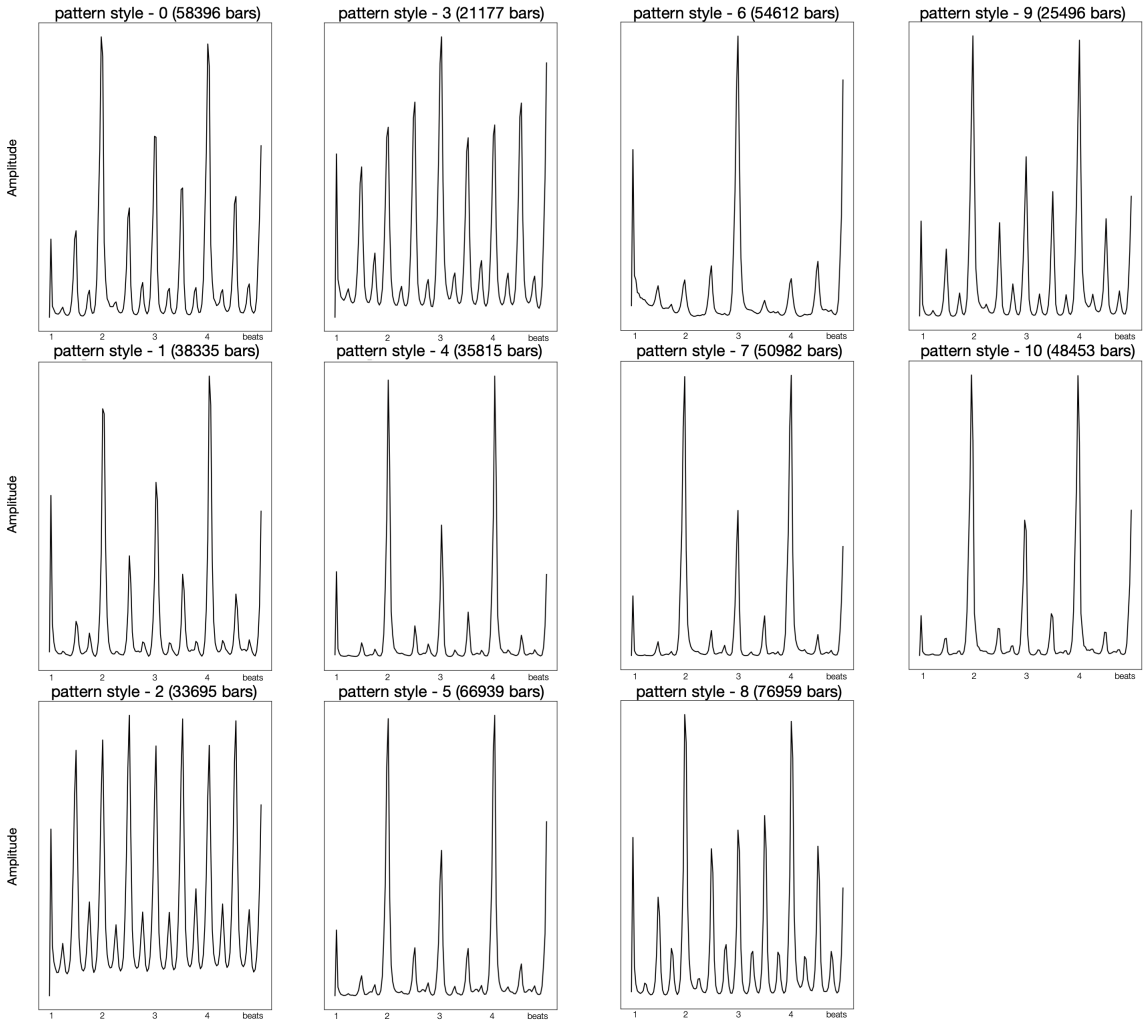

Figure 6.5: Rhythmic pattern style representations for patterns labelled from 0 to 11 extracted for the mid range (120–2500 Hz).

patterns for the mid frequency band to illustrate the differences in different pattern representations. The numbers of bars for each of the pattern clusters are shown in order from the least to highest number in Table 6.1 as well as above each rhythmic envelope representation in Figure 6.5. The clusters sizes range from 21177 bars (ξ_3) to 76959 (ξ_8) with a mean of 46442 bars and standard deviation of 17332 bars.

6.2 Evaluations

The proposed system described in Section 6.1 is assessed through an experiment to determine (1) the rhythmic pattern organisation in the *latent space structure*, (2) an evaluation of the audio *reconstruction performance* compared with similar AE models, and (3) an evaluation of the transformation quality between source and target patterns through *latent space interpolation*. In this section the dataset, experimental methodology and baseline systems under evaluation are presented.

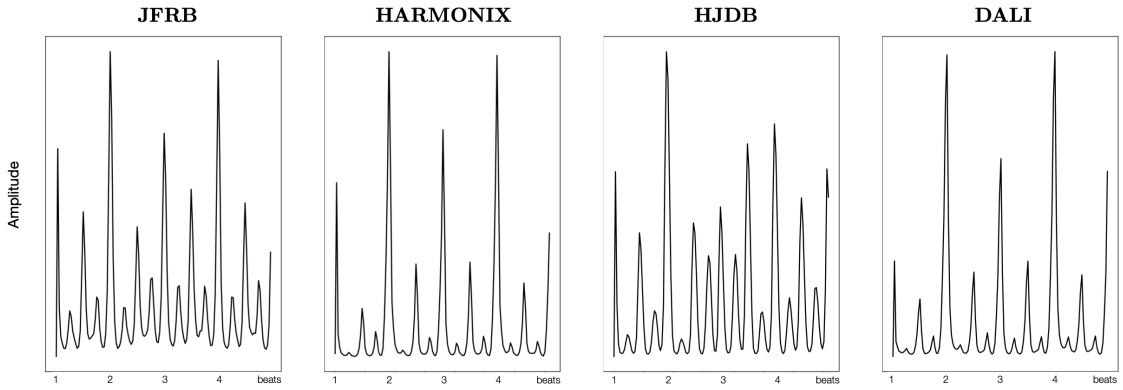

Figure 6.6: Mean bar-length pattern representations for different datasets (columns): `JFRB` (Tomczak et al., 2020), `HMX` (Nieto et al., 2019), `HJDB` (Hockman et al., 2012), and `DALI` (Meseguer-Brocal et al., 2018). Rhythmic envelopes are averaged over patterns extracted from recordings filtered using high (2500–11025 Hz), mid (120–2500 Hz), and low (40–120 Hz) frequency bands (rows).

6.2.1 Dataset

This project makes use of three publicly available datasets: (1) `DALI` (4116 tracks) (Meseguer-Brocal et al., 2018), (2) Harmonix (`HMX` 807 tracks) (Nieto et al., 2019), and (3) `HJDB` (227 tracks) (Hockman et al., 2012), as well as a private collection of 268 jazz, funk and R&B (`JFRB`) recordings. For reproducibility, all track filenames from all datasets are available on the supporting website.[31] Mean rhythmic pattern representations for four datasets are plotted in Figure 6.6. While `HMX` and `DALI` rhythms are quite representative of Western popular music with heavy accents on beat 2 and 4, the patterns in `HJDB` and `JFRB` show syncopated sixteenth note patterns. The resulting dataset contains 5418 musical pieces of polyphonic sound mixtures having various kinds of instruments and represents a wide variety of genres and rhythmic patterns. All audio recordings are in 16-bit mono WAV format and resampled to 22.05 kHz. To facilitate modelling of rhythmic patterns, those tracks are segmented into bars using the downbeat tracking algorithm by Böck et al. (2016b).

In order to model rhythmic patterns from percussion instruments present in the dataset, source separation is performed with the pre-trained *4stems* model provided in the Spleeter library Hennequin et al. (2020) to extract drum sounds from music sound mixtures. The resultant drum parts are used in two ways: (1) as training inputs described in Section 6.1.2.1, and (2) for rhythmic pattern modelling described in Section 6.1.3. In both scenarios, tracks with time signatures other than $\frac{4}{4}$ or with a peak amplitude < 0.2—due either to empty bars or poor source separation—are excluded. After filtering, the data is represented by 5418 tracks with a total of 510859 bars. Assessment of the dataset tempi results in a median tempo of 128 BPM. To facilitate appropriate representation of a wide range of rhythmic patterns, all bar-length segments are time-stretched to a fixed tempo of 128 BPM with the Rubberband library.[32] To reduce the overfitting or underfitting (see Section 3.1.4) during training, the

[31] `https://maciek-tomczak.github.io/acm2020/`
[32] `https://breakfastquay.com/rubberband/`

dataset samples are distributed among training (80%), validation (10%) and test sets (10%) with an equal distribution of bars per ξ rhythmic pattern styles throughout all sets.

6.2.2 Experimental Methodology

In order to view the organisation of the learned latent space, its structure is visualised with 2D and 3D plots for each of the rhythmic classes ξ with principal component analysis (PCA) portraying differences between two different prior distributions. The ability of the proposed model to generate spectrograms is evaluated using both timbral and temporal reconstruction metrics: root-mean squared error (RMSE), log-spectral distance (LSD) and cosine similarity (CS). The LSD is calculated as follows:

$$LSD = \sqrt{\sum \Big[10 \log_{10}(|S|) - 10 \log_{10}(|\hat{S}|)\Big]^2}. \tag{6.11}$$

Following Davies et al. (2014a), temporal reconstruction of the generations is evaluated with rhythmic cosine similarity (RCS) between rhythmic envelopes R (see Section 6.1.3) extracted from source χ and generated ν recording. The reported CS_{R_χ, R_ν} is computed between rhythmic envelopes R_χ and R_ν following the calculation presented in Equation (4.17). RCS will be close to unity for very similar patterns and nearer to zero for dissimilar patterns. To evaluate the continuity of the transformations, latent space interpolations between rhythmic patterns are performed using Equation (6.10). Scores for each metric are calculated between the source recording and the resulting rhythmic transformation. Reconstruction scores for all examples are scaled to range [0,1] and averaged for different α across three folds. Three-fold cross validation is performed to establish that the validation and testing subsets portray a true representation of the training dataset. For example, if the test or validation subsets were biased in some way towards a particular rhythmic pattern, then the system performance could be inaccurately presented. Hence different biases in the dataset can be reduced as well as the reliability of the reported results can be improved. Evaluations in the following sections use 1000 patterns from each ξ rhythmic pattern style, resulting in a total of 11000 evaluation audio examples per fold.

6.2.3 Baseline Systems

In addition to the proposed Gaussian mixture AAE (`AAE-GM`) architecture, three additional models are implemented for comparisons: (1) AAE using isotropic Gaussian prior distribution (`AAE-ISO`), (2) variational autoencoder (`VAE`), (3) a Wasserstein autoencoder with maximum mean discrepancy (`WAE-MMD`) regularisation. All models share the same architecture implementations and are trained in a supervised manner. The proposed `AAE-GM` uses a regularisation based on a WGAN adversarial framework—including a gradient penalty with Guassian mixture prior for conditional disentanglement of rhythmic pattern styles—for the transformation of rhythmic and timbral qualities of drum recordings. As a comparison for the rhythmic transformation capabilities of the presented `AAE-GM` model, the audio synthesis framework using `WAE-MMD` (Bitton et al., 2019) is here modified to act on longer timescales

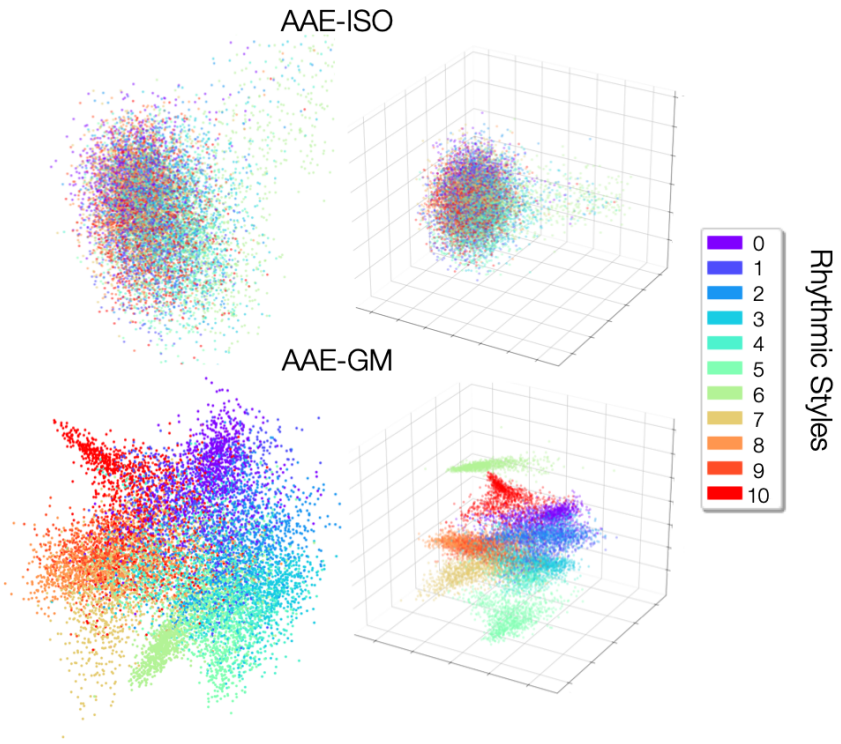

Figure 6.7: PCA visualisations of the baseline `AAE-ISO` (top) and the proposed `AAE-GM` (bottom) with 2 PCs (left) and 3 PCs (right) for 11 rhythmic styles.

as present in bar-length patterns. The `WAE-MMD` implementation is followed without the conditioning module proposed by the authors in (Bitton et al., 2019). In the case of `VAE`, the model minimises the evidence lower bound objective (Kingma and Welling, 2013) with isotropic Gaussian latent distribution. The `WAE-MMD` uses *BCE* reconstruction loss where the regularisation from Equation (6.2) is replaced with MMD (see Section 3.2). MMD represents a distance measure between the samples of the distributions $x \sim p(x)$ and $y \sim q(y)$ and was proposed as a more flexible regularisation to Kullback–Leibler divergence used in a baseline `VAE` (Bitton et al., 2019). MMD defines a differentiable divergence and was developed as a distance between probabilistic moments that map to a general reproducing kernel Hilbert space as defined in Equation (3.33). The kernel function used in computation of MMD is the radial basis kernel defined in Equation (3.34).

6.3 Results and Discussion

Audio examples and additional experiments are available on a supporting website.[33]

6.3.1 Latent Space Structure

The 64-dimensional latent spaces for `AAE-GM` and `AAE-ISO` are visualised in 2D and 3D in Figure 6.7 using PCA. PCA ensures that the visualisation is a linear transform of the original space, and thus preserves the real distances inside the latent space. As can be seen, it is not possible to distinguish

[33] `https://maciek-tomczak.github.io/maciek.github.io/Drum-Synthesis-and-Rhythmic-Transformation/`

	RCS				LSD				RMSE			
Pattern	AAE-GM	AAE-ISO	VAE	WAE-MMD	AAE-GM	AAE-ISO	VAE	WAE-MMD	AAE-GM	AAE-ISO	VAE	WAE-MMD
0	0.789	0.758	0.694	0.757	9.825	11.014	11.777	10.912	0.309	0.330	0.376	0.326
1	0.788	0.755	0.704	0.757	9.716	10.998	11.729	10.935	0.307	0.329	0.375	0.329
2	0.786	0.755	0.697	0.747	9.805	11.120	11.851	11.039	0.311	0.333	0.378	0.330
3	0.682	0.653	0.581	0.654	10.020	11.144	11.843	11.030	0.411	0.435	0.478	0.431
4	0.790	0.762	0.709	0.756	9.751	11.015	11.714	10.891	0.308	0.329	0.373	0.325
5	0.787	0.758	0.710	0.760	9.742	10.965	11.686	10.858	0.307	0.329	0.374	0.326
6	0.790	0.761	0.712	0.760	9.751	11.025	11.751	10.920	0.307	0.330	0.374	0.326
7	0.789	0.762	0.710	0.757	9.738	10.960	11.683	10.827	0.306	0.328	0.372	0.323
8	0.738	0.712	0.658	0.703	9.778	11.034	11.734	10.912	0.308	0.330	0.374	0.326
9	0.731	0.708	0.657	0.704	9.908	11.139	11.830	11.018	0.413	0.435	0.471	0.431
10	0.792	0.762	0.706	0.755	9.744	11.024	11.770	10.908	0.309	0.330	0.376	0.327
Means	0.769	0.741	0.685	0.737	9.798	11.040	11.761	10.932	0.327	0.349	0.393	0.345
σ	0.036	0.035	0.040	0.035	0.091	0.065	0.060	0.069	0.042	0.043	0.040	0.042

Table 6.2: Reconstruction results using rhythmic cosine similarity (RCS), log-spectral difference (LSD) and root-mean squared error (RMSE) presented per rhythmic pattern for the proposed AAE-GM and baseline systems AAE-ISO, VAE, and WAE-MMD. Colour strengths correspond to the results calculated across all rhythmic patterns (0–10) for each metric and system (i.e., per column). Higher RCS values and lower LSD and RMSE values indicate better reconstruction performance shown with green shading whereas lower performance is shown with red shading.

between the different rhythmic pattern styles in AAE-ISO without the Gaussian mixture prior. The effect of the proposed AAE-GM with Gaussian mixture prior can be clearly seen with more visibly organised clusters in both 2D and 3D PCA representations. When analysing mean rhythmic pattern representations as clustered by the X-means algorithm, pattern types ξ_0 (purple) and ξ_6 (green) represent disparate rhythmic styles—style ξ_0 is typified by a clear 16th-note pattern and style ξ_6 is an 8th-note pattern with an accent on the second beat in the bar (see Figure 6.5).

6.3.2 Reconstruction Performance

The reconstruction performance scores of the proposed and baseline models for each pattern style are shown in Table 6.2. The mean LSD and RMSE scores describe the spectral reconstruction quality of generated audio spectrograms with regard to the original. The results for mean LSD and RMSE indicate that the proposed AAE-GM model achieves a higher level of reconstruction quality than the other approaches. The mean differences between AAE-GM, AAE-ISO, and WAE-MMD are larger across all patterns when compared to the VAE system. This indicates that the generations produced by the baseline VAE are more noisy in comparison to the input examples.

The best reconstructed pattern across all systems is ξ_7, as indicated by lowest mean values for LSD (10.802) and RMSE (0.332) metrics. Conversely, the most challenging pattern to generate across systems is ξ_3 with highest means in LSD (11.00) and RMSE (0.439). The second most challenging pattern across systems in both metrics is ξ_9 with mean values in LSD (10.974) and RMSE (0.438). While the AAE-GM exhibits higher variability in its LSD reconstruction results across different patterns, it consistently outperforms the average reconstruction capabilities of other systems. This indicates that the AAE-GM adapts to more challenging rhythmic patterns, making it a more versatile model in diverse contexts. In contrast, the VAE shows the least variability in its LSD reconstructions, as indicated

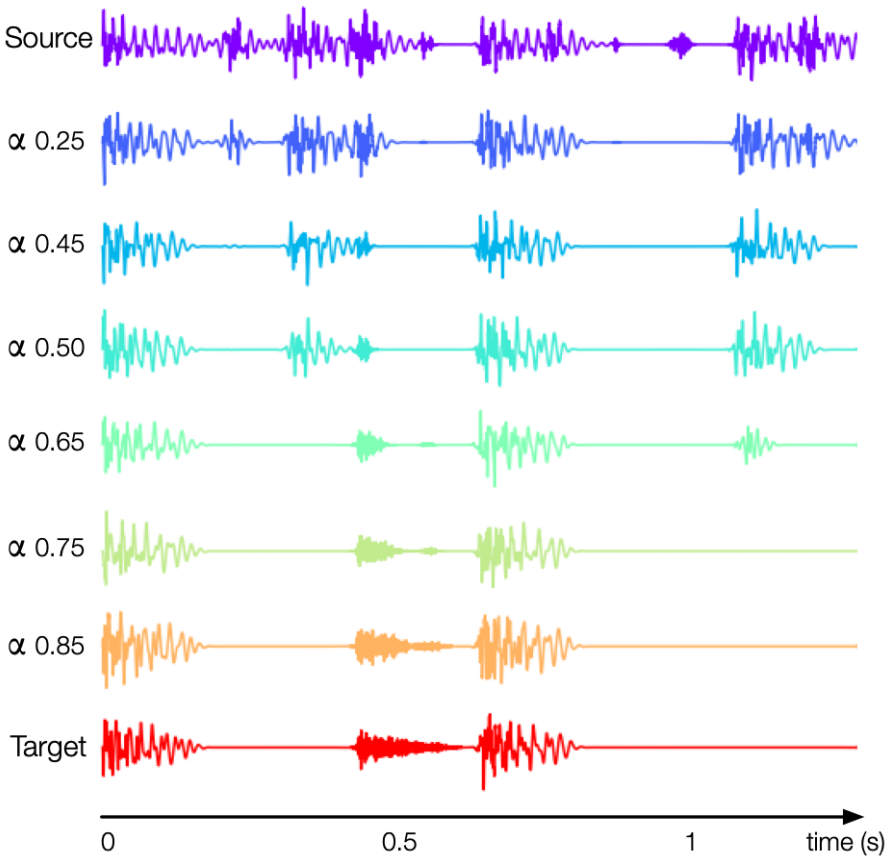

Figure 6.8: Example of interpolation between two rhythmic patterns.

by a standard deviation (σ) of 0.06. However, its adaptability is the poorest among the evaluated models, resulting in consistently subpar reconstruction performance across all patterns. The poor performance of all systems on ξ_3 and ξ_9 is correlated with the lowest numbers of bars in these two rhythmic styles (see Table 6.1). This indicates that the model does not perform as well with lower quantities of training examples. A similar finding is present in the rhythmic RCS results, where the highest mean performance across systems can be observed in ξ_6 (0.756) and ξ_7 (0.755). Both correspond to rhythmic styles with a larger number of bars than in ξ_3 and ξ_9. The RCS score quantifies how similar are the rhythmic envelopes of the newly synthesised audio in comparison to the original and the proposed `AAE-GM` produces higher RCS results than `AAE-ISO` for all pattern styles. Results from t-tests computed across transformations and folds demonstrate that the improvement for rhythmic RCS is significant ($\rho < 0.05$). This demonstrates the advantage of the Gaussian mixture prior in the `AAE-GM` system when compared to the baseline `AAE-ISO` systems. Although the reconstructions from the `AAE-GM`, `WAE-MMD` and `AAE-ISO` all contain a degree of noise, the results of these three systems achieve comparable RCS (> 0.7). The RCS for the `VAE` is considerably lower, likely due to the reconstructions being generated with a more substantial amount of noise. While all systems show similar variability in RCS scores, the `VAE` has a slightly higher standard deviation, indicating more inconsistency in its rhythmic pattern reconstructions. To mitigate the artifacts caused by the time-stretching effect—which contribute to the overall added noise in the transformations, as evidenced by the comparable variance of

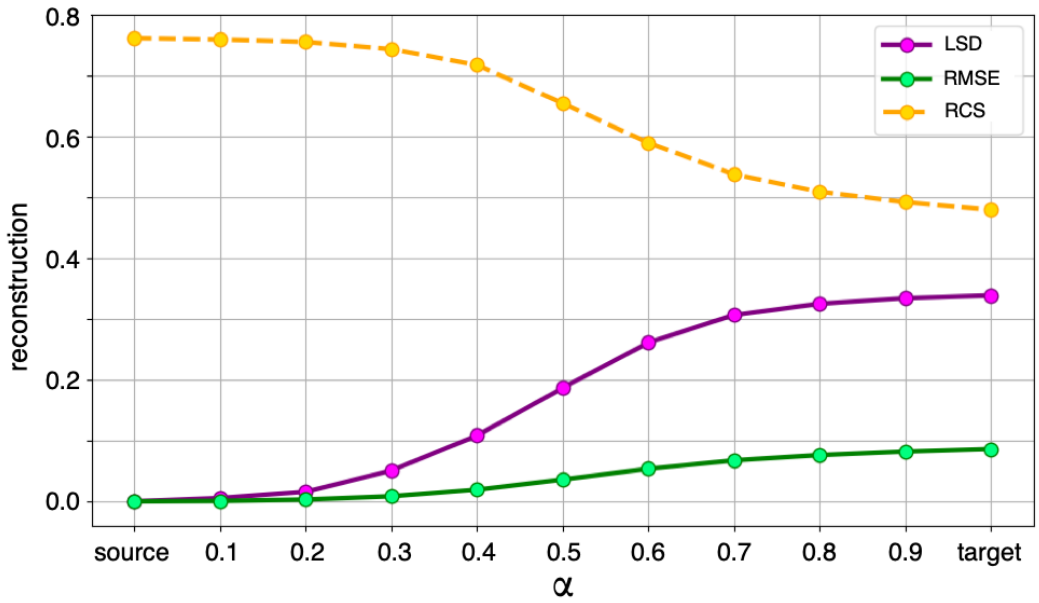

Figure 6.9: Reconstruction scores for interpolations between source and target rhythmic patterns.

RMSE reconstructions—it would be beneficial to explore the use of variable-length features in neural network training. Furthermore, the transformation would also benefit from additional drum placement information provided by accurate automatic drum transcription.

6.3.3 Latent Space Interpolation

One of the chief characteristics of a well-trained latent representation is its ability to generate accurately represented samples based on embeddings created by performing linear interpolation (Section 6.1.3.3). Smooth transitions in the latent space are desired in a user-controlled sound transformation (Bitton et al., 2019, 2018). Figure 6.8 demonstrates a transformation between two different types of rhythmic patterns and different instruments (e.g., purple kick drum transforms into a red snare drum at around 0.5s). Notably, the temporal positions of the last two events in the source audio (i.e., purple kicks after 1.0s) are gradually shifted in time as they are morphed into a single softer and higher pitched sound event at $\alpha = 0.5$ before it disappears completely at $\alpha = 0.75$.

To analyse the effect of the rhythmic transformation for the intermediate α values, all audio examples in the test set are interpolated to randomly chosen target patterns. Figure 6.9 depicts the reconstruction scores calculated as means from all folds and transformations for each interpolated value for α. As expected, the RCS decreases as the transformation moves output audio further away from the source and highlights an upper bound of the RCS reconstruction performance of the system at approximately 0.77. Results from t-tests computed across transformations and folds demonstrate that the change of the RCS of all patterns is significant ($\rho < 0.05$), which shows that the synthesised patterns are significantly different from the source patterns. Moreover, as the transformation operates on audio containing percussion only, the intention is not to adjust the spectral content by a large margin. The scores for RMSE reflect that characteristic by not varying considerably throughout the interpolation, indicating

that the spectrogram reconstruction quality remains similar. On the other hand, LSD mirrors the behaviour of RCS indicating change in spectral content moving towards a novel target spectrogram after transformation. This can be equivalent to moving and removing an event in one position or transforming it into another instrument.

6.3.4 Conclusions

The visual analysis of the 64-dimensional latent spaces using PCA has revealed significant structural differences between the `AAE-GM` and `AAE-ISO` models. With the Gaussian mixture prior integrated into `AAE-GM`, the latent space exhibited distinct, well-organised clusters, a feature not observable in the `AAE-ISO` model. This clustering pattern aligns with the delineation of rhythmic styles, such as the 16th-note patterns represented by ξ_0 and the accented 8th-note patterns denoted by ξ_6. A comprehensive comparison of reconstruction performance between the proposed `AAE-GM` model and baseline systems (`AAE-ISO`, `VAE`, and `WAE-MMD`) was presented using three metrics: rhythmic cosine similarity (RCS), log-spectral difference (LSD), and root-mean squared error (RMSE). These metrics collectively suggest that `AAE-GM` surpasses the baseline systems in maintaining the spectral characteristics of the original audio and suggests its potential for superior noise reduction in audio reconstructions. The variability in the `AAE-GM`'s performance across different rhythmic patterns underscores its adaptability and versatility in dealing with diverse musical contexts. While pattern ξ_7 was the most successfully reconstructed across all models, patterns ξ_3 and ξ_9 presented significant challenges, which is correlated with the smaller number of training examples for these styles. This highlights a crucial dependence of model performance on the volume of training data. Interestingly, even within these limitations, the `AAE-GM`: model exhibited a consistently higher RCS across all pattern types, underscoring the effectiveness of the Gaussian mixture prior in capturing the rhythmic nuances of the original samples. The statistical significance of these improvements, as confirmed by t-tests, validates the proposed model's enhancements over the baseline systems.

6.4 Chapter Summary

This chapter introduced a novel deep generative system that employs Gaussian mixture adversarial autoencoders (AAEGM) for a joint drum synthesis and pattern modification, akin to the popular task of redrumming. This approach offers users the flexibility to dynamically navigate through a trained latent space of rhythmic variations by manipulating patterns within a bar-length sequence, without needing to segment notes discretely. The system was rigorously trained and tested on a dataset of more than 500,000 bars derived from 5,418 tracks across various music genres, highlighting its ability to adapt to diverse musical contexts. The `AAE-GM` model has demonstrated its superiority over baseline models (`AAE-ISO`, `VAE`, and `WAE-MMD`) through extensive evaluation, outperforming them in maintaining the original audio's spectral integrity. This is reflected in the metrics of rhythmic cosine similarity (RCS), log-spectral difference (LSD), and root-mean squared error (RMSE). The novel Gaussian mixture model exhibits

capability in capturing the intricate rhythmic styles within the music, as evidenced by the clear clustering of styles in its 64-dimensional latent space—a feature that was not present in the `AAE-ISO` model. In the next chapter, the key takeaways from this and preceding chapters are synthesized, along with a discussion on the existing limitations and potential avenues for future research in this field. This is followed by a discussion of limitations that still exist and future directions that could be explored.

Chapter 7

Conclusions

This thesis has focused on rhythmic and timbral transformation of drum recordings using deep learning techniques. While deep generative modelling for musical audio synthesis has been subject of previous works, relatively few such works have explored generation and evaluation techniques for percussion recordings as well as transformation of the underlying rhythmic patterns. One of the main contributions of this thesis is the development of automated systems for joint rhythmic and timbral transformation which provide controllable transformations learned by a generative model. Results from evaluations of the proposed systems demonstrate that deep generative models (DGMs) are well suited for transformation of rhythmic patterns as well as neural synthesis of drum sounds within a variety of musical genres. However, the current systems may still produce unintended errors, indicating opportunities for improvement within the broader field of neural audio synthesis.

The modes of rhythmic transformation proposed in this thesis are presented in three main systems. Firstly, an audio style transfer (AST) approach with rhythmic constraints for mixing and mashing of two or more audio inputs was introduced, to determine to what extent can the system be used to control the rhythmic and timbral characteristics of drum recordings based on drum synthesis of beat-length segments. Secondly, a mode of transformation that relies on drum translation was implemented and evaluated with regard to how accurately can such system successively generate drum segments of varying lengths containing individual drum types. The high-level evaluations of this system highlighted potential improvement strategies which were implemented and resulted in improved rhythmic transformations with higher quality of the generated drum sounds. Lastly, these systems were combined to jointly perform neural drum synthesis and rhythmic transformation as part of a continuous transformation which alleviated the need for discretised note detection or pattern matching present in the previous approaches. The following reiterate the contents, contributions and conclusions of the systems presented in the main chapters of this thesis.

Chapter 4 introduced an exploration into rhythmically-constrained audio style transfer, offering transformative insights and robust evaluations. This chapter has made a significant contribution to the overarching goal of the thesis, which is to advance methodologies in automated rhythmic transformation of drum recordings. At the core of this chapter is the introduction of an innovative model for raw audio generation and rhythmic transformation. This system forms the basis for an exploration of the rhythmic transformation of drum recordings, underpinning the chapter's contributions. In the rhythmically-constrained AST system, a source recording of arbitrary length is automatically segmented and modified through the AST mode of transformation. A pivotal contribution is the introduction of three novel AST loss objectives: $\mathcal{L}3$, $\mathcal{L}4$, and $\mathcal{L}_5$. These objectives enhance the traditional AST formulation by facilitating the interplay between rhythmic and timbral characteristics, effectively creating a rhythmic-timbral mashup functionality. This extension introduces a new level of control into the transformation process. This extension introduces control into the transformation process. Through the proposed rhythmic evaluation, it becomes evident that the mashup objective $\mathcal{L}3$ and the augmented mashup objectives $\mathcal{L}4, 5$ generate entirely new rhythmic patterns. These patterns emerge as distinct entities, significantly diverging from the original drum recordings, and underscore the system's capability to inject novel rhythmic elements into audio transformations. Investigations into timbral reconstruction capabilities have unveiled a compelling consistency. All newly introduced mashup loss objectives preserve the quality across the frequency spectrum while simultaneously generating outputs that differ from the standard AST formulation—a desirable feature for timbral transformations. Another set of evaluations explored the extent of rhythmic modifications generated by the system, using an automatic onset detection system. These evaluations further demonstrated that the proposed AST system for rhythmic transformation created new events in objectives $\mathcal{L}3, 4, 5$ that were not present in the original inputs. Moreover, the results highlighted that a more controlled and rhythmically varied transformation could be achieved by extending the $\mathcal{L}3$ mashup objective with the augmented mashup objectives. The augmented objectives proved their suitability for modifying the rhythmic patterns to become more closely aligned with the chosen input, thereby opening up new possibilities for rhythmic transformation. These can be further refined through additional fine-tuning of the proportion parameters within the loss formulations. This chapter paves the way for more practical applications in the realm of automated rhythmic transformation. Such systems can serve as valuable tools for music producers, facilitating the generation of drum tracks that blend the rhythmic patterns of one composition with the sonic characteristics of another. The implications for music education are also significant, with the potential for developing innovative educational tools in music theory. Such tools could demonstrate how various rhythmic constraints influence the characteristics of drum loops, enhancing the learning experience for aspiring musicians and music enthusiasts alike.

Chapter 5 introduced a granular approach to neural drum synthesis through drum translation. This method autoregressively generates audio samples of percussion sounds, exploring the rhythmic and timbral capabilities of generative audio synthesis with WaveNet autoencoders. Initially, the chapter explored the model architecture and training specifications. The system utilised a dataset comprising

individual drum samples, successively synthesising drum segments of varying lengths, including individual drums and their combinations, referred to as drum domains. These initial evaluations were instrumental in shaping our understanding of the system's capabilities. Notably, this system fosters creativity in musical composition and addresses the challenges of rhythmic transformation in audio synthesis. A unique method for visually fine-tuning large audio datasets was proposed and implemented. This process aids in identifying and excluding outliers that introduce noise. Chapter 5 also introduced a novel evaluation technique based on automatic drum transcription. This method ascertained the efficacy of the drum translation system, enabling an assessment of the accuracy with which it can reproduce drum sounds. This evaluation provides a deeper understanding of the system's performance, especially in translating complex sounds like those of snare drums and hi-hats, known for their intricate timbral characteristics. Moreover, the chapter expands on the reconstruction metrics established in Chapter 4, providing a comprehensive framework for assessing the transformative capabilities of the system. These metrics are essential for understanding the fidelity and the transformative aspects of the generated audio. The comparative analysis of the translated outputs has revealed significant issues, such as waveform smearing and the occasional omission of drum sounds, which underscore the challenges within certain drum domains. The evaluation of reconstruction quality using Pearson correlation coefficients presents a nuanced perspective on the system's performance. High correlation values in low-frequency bands indicate the system's adeptness in recreating kick drum sounds, which are foundational to drum tracks. Conversely, the varied performance in mid-range and higher frequencies points to the challenges in accurately reproducing the complex harmonics of snares and hi-hats. The current model's memory constraints, which limit the system's residual channels and layer count, suggest that more expansive architectures, such as parallel WaveNet, might provide a solution (Oord et al., 2018). These findings offer valuable insights into the capabilities of the drum translation system. However, its direct applicability as a music production tool remains an open question. The objective evaluations conducted represent a significant step in this research field, but transitioning to real-world music production requires further exploration and refinement. Chapter 5 contributes to the field of neural audio synthesis, particularly in advancing the understanding and application of drum translation and redrumming processes. This research has unveiled a system capable of transforming drum sounds and has established a comprehensive set of evaluation metrics for rigorously assessing the quality of such transformations. The advancements presented here open up a multitude of potential use cases, each with the possibility of revolutionising aspects of sound design and musical creation. The proposed system can provide music producers with the ability to tailor drum sounds to fit diverse musical contexts, thus enhancing sound design across various genres. Furthermore, it allows for the augmentation of drum libraries by diversifying existing samples, thereby enriching the resources available to producers and composers. For artists, the drum translation system can facilitate the integration and morphing of drum sounds into compositions, setting a new precedent for personalised drum synthesis and transformative drum transformations that can be extended with personal drum sample collections.

Chapter 6 introduces a system for the continuous transformation of drum recordings. The principal contribution of this chapter is the development of a model capable of fluidly transitioning between different rhythmic patterns and styles, guided by user-defined parameters. This transformation offers continuous control, allowing users to navigate through the complex rhythmic landscape of bar-length patterns via the interpolation of a low-dimensional latent space. By integrating Gaussian mixture (GM) latent distributions for rhythmic pattern conditioning with advanced adversarial autoencoders, the system achieves a transformation process similar to the popular task of redrumming. Experimental assessments have showcased the system's ability to organise rhythmic patterns within its latent space effectively, surpassing other autoencoder models in terms of both rhythmic and timbral fidelity. The architecture of the latent space, especially when utilising the disentangled distributions of the proposed adversarial autoencoder with Gaussian mixtures, has been crucial to this success. High cosine similarity scores highlight the system's proficiency in generating temporal domains, while the log-spectral distance and RMSE metrics point to the inherent challenges of synthesising realistic drum audio with neural networks. It was noted that classes with fewer bars encountered more significant difficulties. Additionally, the chapter delved into the latent space interpolations between source and target patterns, demonstrating the system's ability to facilitate smooth transitions across its learned latent space with varying degrees of transformation. These insights validate the system's design and shed light on potential avenues for future research, particularly concerning the optimisation of generative models for audio synthesis that take into account the scale and diversity of datasets.

7.1 Contributions

As highlighted in Chapter 1, automated rhythmic transformation of drums refers to the automated processes for manipulation (e.g., resequencing, time-stretching) of rhythmic patterns of drum samples or recordings. Deep generative models can be used to learn patterns and timbres in existing drum examples and then generate new drum sounds and rhythms. The aim of this thesis was to develop deep learning-based computational models for automated transformation of rhythmic patterns of percussion instruments. The four most apparent contributions of this work are: formulation and description of modes of transformation for drums using DGMs, development of modular deep learning-based systems for rhythmic transformation, creation of new loss functions for AST, and implementation of comprehensive evaluation methodologies, encompassing similarity metrics, onset detection and automatic drum transcription to rigorously assess and validate the capabilities and efficacy of the proposed systems. The modes of transformation (Section 3.3) are explained in the context of algorithms developed in the field of deep learning, along with details of manipulation possibilities of the rhythmic and timbral characteristics of different drum recordings. These descriptions can be used for differentiating between transformations of patterns and timbres originating from the source and target recordings as well as from the latent space of a trained DGM. To date, neural drum synthesis approaches for the transformation of rhythmic patterns of percussion instruments were not described.

The systems presented in Chapters 4–6 were created to facilitate rhythmic transformation of drum recordings. Each of these modular systems was developed with design attributes that were specific to percussion instruments under analysis (e.g., kicks, snares, hi-hats) and optimised with respect to neural network architecture considerations as well as training hyperparameters. These systems can be used to transform rhythmic patterns through neural drum synthesis of an entire drum kit in a bar-length segment or shorter beat-length and individual drum segments. These transformations can be achieved through control over a reduced parameter space of a deep generative model or through manipulation of the chosen audio input content. In addition, new audio style transfer loss functions were created to facilitate mashup-oriented drum recording transformations. These mashup loss objectives reduced the computation requirements for the AST algorithm and offered rhythmic transformation capabilities which adhere to larger rhythmic structure of the input to generate music that is both creative and realistic. The audio style transfer system from Chapter 4 with the proposed mashup loss objectives was implemented as an open source command line audio effect incorporating user-adjustable parameters to make it more accessible to musicians and producers.[34] It is hoped that these systems may also help uncover musical relationships of familiar audio samples that might otherwise have never been conceptualised.

Additionally, datasets with drum recordings were created to assist the training and evaluations of the proposed systems for drum synthesis and rhythmic transformation. The dataset of individual kick, snare and hi-hat samples introduced in Chapter 5 was refined to improve the quality of the generated examples through automatic and visual filtering[35] of noisy audio samples. A dataset of drum loops presented in Chapter 6 was compiled from 5,418 tracks present in publicly available datasets (e.g., `HMX`, `DALI`, `HJDB`) covering various musical genres. To facilitate evaluation of simple and more complex percussion contexts as well as reproducibility of the results in this thesis, the audio examples and references to other dataset filenames were provided in the online supplementary materials.[36,37,38,39]

7.2 Future Work

There exists a large scope of additional research in different complementary aspects neural drum synthesis and rhythmic transformation presented in this thesis. This section discusses the possible directions this work could take in the future.

Training Data

As the efficacy of generative models relies significantly on the quality and diversity of the training data, there is a continual need to refine and expand datasets to drive advancements in drum synthesis and rhythmic modifications. The custom datasets used in Chapters 4–6 were specifically constructed for

[34] `https://github.com/maciek-tomczak/audio-style-transfer-with-rhythmic-constraints`
[35] `https://tdsdne.vercel.app/`
[36] `https://maciek-tomczak.github.io/rppw2017/`
[37] `https://maciek-tomczak.github.io/dafx2018/`
[38] `https://maciek-tomczak.github.io/dafx2019/`
[39] `https://maciek-tomczak.github.io/acm2020/`

this research. The analysis in Chapter 5 highlighted the direct impact of dataset quality and size on drum translation model performance. Similarly, the system in Chapter 6 demands substantial training data to capture intricate real-life audio patterns. To enhance these systems, future work could include improvement of the source separation models to facilitate access to more real-world percussion examples used for training. Another challenge lies in amassing and authenticating drum sample categories and rhythmic pattern style labels. A possible direction is crowdsourcing of drum samples and loops. This approach can serve as a potent strategy for gathering extensive high-quality data essential for neural drum synthesis models. Crowdsourcing allows for the collection of a wide variety of audio data from many different sources, which can help to increase the diversity and representativeness of the training examples. Such data curation strategies can significantly improve the generalisation capabilities and overall efficacy of neural drum synthesis and rhythmic transformation models. As datasets expand, the need for automated label verification, especially for rhythmic patterns, becomes paramount. Manual checks become less practical, highlighting the importance of automatic evaluations. Future work should assess the robustness and adaptability of models trained on crowdsourced datasets, gauging their real-world applicability. It is crucial that these models avoid overfitting to dataset nuances, ensuring their reliability in drum synthesis and transformation tasks.

Disentangled Representations

Disentanglement remains an unsolved challenge and is an active area of research, particularly important for improving the synthesis control of rhythmic patterns and drum types. Enhancing disentanglement in this domain represents a chief direction for future work. It allows for more fine-grained control over the generated audio, allowing the model to synthesise specific rhythmic styles, instruments, and to adjust certain attributes such as timbre or pitch. It can lead to models that are more robust to variations in the data and can generalise better to new audio examples as well as allow for better understanding of how the model works. Chapter 6 highlighted how the introduced Gaussian mixture prior enhanced rhythmic style disentanglement compared to baseline VAE models. Future research might explore different distributions tailored for diverse datasets and specific musical production needs as well as diffusion-based models (Tschannen et al., 2018; Yang et al., 2023).

Improved Evaluation Metrics for DGMs

Evaluating generative models for drum synthesis and rhythmic modifications poses inherent challenges. As new models are developed to excel on specific evaluation metrics, there exists a risk they might overfit to those metrics, potentially failing to produce audio that is perceptually improved or enhanced in any other dimension. To address this, future research could focus on developing new metrics that accurately capture the diverse characteristics of audio synthesis, consider the subjectivity in auditory perception, and provide an overview of the fidelity, coherence, and variability of the generated outputs. While the evaluation metrics discussed in Chapters 4–6 are considered standard in the field, they might not capture the nuanced rhythmic and timbral changes influenced by the training process. Future

research could prioritise crafting metrics that delve deeper into specific audio attributes, such as semantic content and the microtiming deviations of output transformations, especially in relation to the quantised representations of target patterns. These metrics could also incorporate text-based data to provide semantic insights about the desired transformation, for example, through the use of transformer-based and larger GAN-based models or their combinations (Dubey and Singh, 2023). Unlike the image domain where pixel-wise differences are straightforwardly quantified, the creation of new metrics for audio to gauge continuity within the latent space could offer valuable insights into smoothness and disentanglement. While a metric exists for images (Karras et al., 2021), its adaptation for audio would require considerable modifications.

User Studies

The evaluations detailed in Chapters 4–6 have highlighted the strengths and constraints of the proposed deep learning models for rhythmic transformation and neural drum synthesis. These insights serve as a foundation for the further development and refinement of these systems. Future research would benefit from exploring methodologies for conducting user studies. Specifically, focusing on audio plugins and embedded audio systems could offer users simpler access to the evaluated algorithms compared to command line tools. Such studies could concentrate on discerning how users interact with the deep learning systems, highlighting any challenges or obstacles they encounter, and collecting feedback on their overall experience. The results from user studies could then be used to refine the systems and better integrate them into user workflows. By aligning the deep generative applications with the needs and preferences of users, engagement, interaction, and overall system performance could be significantly enhanced.

Other Percussion Instruments

Future work could also explore other percussion such as more varied cymbals and tom drums. Additionally, Chapter 5 highlighted the challenges related to the generation of audio with multiple overlapping percussion instruments. Incorporating additional modalities, such as MIDI files and textual inputs as well as extending the number of studied percussion instruments beyond kicks, snares and hi-hats, can lead to generating more diverse and realistic audio outputs. Beyond the common hi-hats, there exists a large variety of cymbals – from standard crash and ride to china cymbals, each with its distinct timbre and sonic footprint. They can produce a wide array of overlapping sounds, which pose challenges for rhythmic transformation over longer rhythmic patterns spanning multiple bars. Capturing the nuances of a variety of cymbals can significantly enhance the depth and richness of generated audio.

Audio Plugins and Embedded Audio Processing

As highlighted through Chapters 4 to 6, another area of research is to improve the efficiency for audio generation and transformation. State-of-the-art neural audio synthesis models are incredibly resource-intensive, making their deployment for real-time applications challenging. However, as technology

progresses and optimisation methods improve, there is an increasing potential for these models to be adapted and streamlined for broader uses. This direction of research is pivotal in pushing the boundaries of current applications, especially when it comes to platforms without the luxury of high-powered graphics processing units. Audio plugins, whether they are virtual studio technology (VST), audio units (AU), or other formats, form the backbone of contemporary digital audio workstations. Given their widespread usage in sound design and music production environments, ensuring these plugins can handle advanced audio synthesis without any considerable latency is crucial. Current advancements in this field are focused on reducing computational overhead, improving parallel processing capabilities, and making the most of available hardware resources. Moreover, developing more efficient models or methods for optimising existing models could make them more practical for a wider range of use cases which could enable them to run on resource-constrained devices such as mobile phones and embedded systems (e.g., inside Eurorack modules).

Computational Creativity

To date, the majority of research in deep generative modelling for neural audio synthesis and rhythmic transformation has focused on mimicking human creative traits akin to developing content proposal generators, an approach that potentially underutilises the almost unlimited computational power of machines. The relevance of this research has been highlighted through the emergence of nuanced perspectives that might consider a greater emphasis on examining representation spaces (e.g., learning disentangled representations of rhythm and timbre) within these models, in order to redefine the relationship between humans and machines in the music production process. In considering future research paths, Esling and Devis (2020) outline two directions which emerge as particularly compelling. On the a technical front, there exists an important potential future work in discerning the mathematical properties underpinning deep generative models for music creation. The future work could also delve into the sociological facets of creativity, with the aim of creating a novel category of creatively intelligent music production systems which would reflect our current understanding of creativity in the era of deep generative systems for music creation.

7.3 Final Thoughts

As it was revealed to me during this research, the topic of neural audio synthesis, amasses extremely creative musicians and researchers who articulate their passion for music in a range of artistic and scientific disciplines. Neural composers and researchers in this field are constantly developing new methods and models to improve the quality and realism of the generated audio, and there are many contributions being made in this area every year. I, optimistically, hope that the work undertaken in this thesis can serve as an inspiration for further research in neural drum synthesis and a small step towards the comprehension of deep learning-based techniques for rhythmic transformation of drum recordings. I am excited to see how deep learning techniques advance in the coming years and I hope that this work will inspire the development of new contributions in the future.

References

Adamo, M. (2010). *The Breakbeat Bible*. Hudson Music, New York, USA (cit. on p. 13).

Alén, O. (1995). "Rhythm as Duration of Sounds in Tumba Francesa". In: *Ethnomusicology, 39(1), Special Issue: Participatory Discrepancies*, pp. 55–71 (cit. on p. 12).

Allan, H. (2004). *Bar Lines and Beyond - Metre Tracking in Digital Audio*. Master's thesis, University of Edinburgh, p. 28 (cit. on p. 14).

Amatriain, X., J. Bonada, L. Loscos, J. L. Arcos, and V. Verfaille (2003). "Content-based Transformations". In: *Journal of New Music Research, 32(1)*, pp. 95–114 (cit. on pp. 1, 3, 23, 24).

Arjovsky, M. and L. Bottou (2017). "Towards Principled Methods for Training Generative Adversarial Networks". In: *CoRR abs/1701.04862* (cit. on p. 49).

Arjovsky, M., S. Chintala, and L. Bottou (2017). "Wasserstein Generative Adversarial Networks". In: *Proceedings of the International Conference on Machine Learning (PMLR), Sydney, Australia*, pp. 214–223 (cit. on pp. 49, 99).

Arthur, D. and S. Vassilvitskii (2006). *k-means++: The Advantages of Careful Seeding*. Tech. rep. Stanford University (cit. on p. 104).

Aucouturier, J.-J., F. Pachet, and M. Sandler (2005). "The way it Sounds: Timbre Models for Analysis and Retrieval of Music Signals". In: *IEEE Transactions on Multimedia, 7(6)*, pp. 1028–1035 (cit. on p. 2).

Barry, S. and Y. Kim (2018). *Style Transfer for Musical Audio Using Multiple Time-Frequency Representations*. Accessed 15 February 2018, OpenReview (cit. on pp. 29, 53, 56–58, 61, 64, 73, 74).

Battenberg, E., V. Huang, and D. Wessel (2012). "Toward Live Drum Separation Using Probabilistic Spectral Clustering Based on the Itakura-Saito Divergence". In: *Proceedings of the Audio Engineering Society Conference (AES) Conference on Time-Frequency Processing in Audio. Helsinki, Finland*, pp. 1–10 (cit. on p. 21).

Bello, J. P., C. Duxbury, M. E. Davies, and M. Sandler (2004). "On the Use of Phase and Energy for Musical Onset Detection in the Complex Domain". In: *IEEE Signal Processing Letters, 11(6)*, pp. 553–556 (cit. on pp. 17, 18).

Bello, J. P., E. Ravelli, and M. B. Sandler (2006). "Drum Sound Analysis for the Manipulation of Rhythm in Drum Loops". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toulouse, France*, pp. 233–236 (cit. on p. 21).

Bello, J. P., L. Daudet, S. Abdallah, C. Duxbury, M. E. Davies, and M. B. Sandler (2005). "A Tutorial on Onset Detection in Music Signals". In: *IEEE Transactions on Speech and Audio Processing, 13(5)*, pp. 1035–1047 (cit. on pp. 15–17, 67).

Bello, J. P. and M. Sandler (2003). "Phase-based Note Onset Detection for Music Signals". In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Hong Kong*, pp. 441–444 (cit. on p. 17).

Bengio, Y., R. Ducharme, and P. Vincent (2000). "A Neural Probabilistic Language Model". In: *Proceedings of the Neural Information Processing System (NIPS), Denver, USA*, pp. 932–938 (cit. on p. 46).

Bilmes, J. A. (1993). *Timing is of the Essence*. Master's thesis, Massachusetts Institute Of Technology (cit. on pp. 12, 14, 22, 25).

Bittner, R. M., M. Gu, G. Hernandez, E. J. Humphrey, T. Jehan, H. McCurry, and N. Montecchio (2017). "Automatic Playlist Sequencing and Transitions." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Suzhou, China*, pp. 442–448 (cit. on p. 3).

Bitton, A., P. Esling, A. Caillon, and M. Fouilleul (2019). "Assisted Sound Sample Generation with Musical Conditioning in Adversarial Auto-encoders". In: *Proceedings of the International Conference on Digital Audio Effects (DAFx), Birmingham, UK* (cit. on pp. 52, 97, 98, 100, 101, 108, 109, 112).

Bitton, A., P. Esling, and A. Chemla-Romeu-Santos (2018). "Modulated Variational Auto-encoders for Many-to-many Musical Timbre Transfer". In: *CoRR abs/1810.00222* (cit. on p. 112).

Böck, S. (2016). "Event Detection in Musical Audio". PhD thesis. Johannes Kelper University Linz (cit. on pp. 2, 18, 19).

Böck, S., A. Arzt, F. Krebs, and M. Schedl (2012a). "Online Real-time Onset Detection with Recurrent Neural Networks". In: *Proceedings of the International Conference on Digital Audio Effects (DAFx), York, UK* (cit. on p. 16).

Böck, S. and M. E. P. Davies (2020). "Deconstruct, Analyse, Reconstruct: How to improve Tempo, Beat, and Downbeat Estimation". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Montréal, Canada*, pp. 574–582 (cit. on p. 20).

Böck, S., M. E. P. Davies, and P. Knees (2019). "Multi-Task Learning of Tempo and Beat: Learning One to Improve the Other". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Delft, Netherlands*, pp. 486–493 (cit. on p. 20).

Böck, S., F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer (2016a). "madmom: A New Python Audio and Music Signal Processing Library". In: *Proceedings of the ACM International Conference on Multimedia (ACM-MM), Amsterdam, Netherlands*, pp. 1174–1178 (cit. on pp. 57, 103).

Böck, S., F. Krebs, and M. Schedl (2012b). "Evaluating the Online Capabilities of Onset Detection Methods". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Porto, Portugal*, pp. 49–54 (cit. on pp. 16, 18, 23, 68, 103).

Böck, S., F. Krebs, and G. Widmer (2016b). "Joint Beat and Downbeat Tracking with Recurrent Neural Networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), New York City, USA*, pp. 255–261 (cit. on pp. 20, 57, 103, 107).

Böck, S. and M. Schedl (2011). "Enhanced Beat Tracking with Context-aware Neural Networks". In: *Proceedings of the International Conference on Digital Audio Effects (DAFx), Paris, France* (cit. on p. 19).

Böck, S. and G. Widmer (2013a). "Local Group Delay Based Vibrato and Tremolo Suppression for Onset Detection". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Curitiba, Brazil*, pp. 361–366 (cit. on p. 17).

— (2013b). "Maximum Filter Vibrato Suppression for Onset Detection". In: *Proceedings of the International Conference on Digital Audio Effects (DAFx), Maynooth, Ireland* (cit. on p. 17).

Briot, J.-P. and F. Pachet (2017). "Music Generation by Deep Learning-Challenges and Directions". In: *CoRR abs/1712.04371* (cit. on pp. 27, 74).

Butler, M. J. (2006). *Unlocking the Groove: Rhythm, Meter, and Musical Design in Electronic Dance Music*. Indiana University Press, Indiana, USA (cit. on p. 13).

Carney, M., C. Li, E. Toh, N. Zada, P. Yu, and J. Engel (2021). "Tone Transfer: In-Browser Interactive Neural Audio Synthesis". In: *Conference on Intelligent User Interfaces* (cit. on p. 2).

Chang, J. (2007). *Can't Stop Won't Stop: A History of the Hip-Hop Generation*. St. Martin's Press, New York, USA (cit. on p. 2).

Chen, T. Q. (2017). "Deep Kernel Mean Embeddings for Generative Modeling and Feedforward Style Transfer". PhD thesis. University of British Columbia (cit. on p. 45).

Child, R., S. Gray, A. Radford, and I. Sutskever (2019). "Generating Long Sequences with Sparse Transformers". In: *CoRR abs/1904.10509* (cit. on pp. 78, 79).

Choi, K., D. Joo, and J. Kim (2017). "Kapre: On-GPU Audio Preprocessing Layers for a Quick Implementation of Deep Neural Network Models with Keras". In: *Machine Learning for Music Discovery Workshop at the International Conference on Machine Learning (ICML)* (cit. on p. 58).

Chung, J., K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio (2015). "A Recurrent Latent Variable Model for Sequential Data". In: *Proceedings of the Neural Information Processing System (NIPS), Montréal, Canada* (cit. on p. 29).

Clevert, D.-A., T. Unterthiner, and S. Hochreiter (2015). "Fast and Accurate Deep Network Learning by Exponential Linear Unit". In: *CoRR abs/1511.07289* (cit. on p. 81).

Cocharro, D., G. Sioros, M. Caetano, and M. E. P. Davies (2014). "Real-time Manipulation of Syncopation in Audio Loops". In: *Proceedings of the International Computer Music Conference (ICMC) joint with the Sound and Music Computing Conference (SMC), Athens, Greece* (cit. on p. 26).

Collins, N. (2001). "Algorithmic Composition Methods for Breakbeat Science". In: *Proceedings of the International Conference for Music without Walls and Instruments, Leicester, UK* (cit. on p. 9).

Collins, N. and B. L. Sturm (2011). "Sound Cross-synthesis and Morphing using Dictionary-based Methods". In: *Proceedings of the International Computer Music Conference (ICMC)* (cit. on p. 27).

Cooper, G. and L. B. Meyer (1963). *The Rhythmic Structure of Music*. University of Chicago Press, Chicago, USA (cit. on p. 14).

Crocker, M. J. (1998). *Handbook of Acoustics*. John Wiley & Sons, New York, USA (cit. on p. 16).

Davies, M. E. P. (2007). "Towards Automatic Rhythmic Accompaniment". PhD thesis. Department of Electronic Engineering, Queen Mary University of London (cit. on p. 19).

Davies, M. E. P., P. Hamel, K. Yoshii, and M. Goto (2013). "AutoMashUpper: An Automatic Multi-Song Mashup System". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Curitiba, Brazil*, pp. 575–580 (cit. on pp. 27, 66).

— (2014a). "AutoMashUpper: Automatic Creation of Multi-song Music Mashups". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP), 22(12)*, pp. 1726–1737 (cit. on pp. 3, 19, 27, 66, 108).

Davies, M. E. P. and M. D. Plumbley (2006). "A Spectral Difference Approach to Extracting Downbeats in Musical Audio". In: *Proceedings of the European Signal Processing Conference (EUSIPCO), Florence, Italy* (cit. on p. 20).

Davies, M. E. P., A. M. Stark, F. Gouyon, and M. Goto (2014b). "Improvasher: A Real-time Mashup System for Live Musical Input". In: *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME), London, UK*, pp. 541–544 (cit. on p. 27).

Davy, M. and S. Godsill (2002). "Detection of Abrupt Spectral Changes Using Support Vector Machines an Application to Audio Signal Segmentation". In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Orlando, USA*, pp. 1313–1316 (cit. on p. 17).

Desain, P. and H. Honing (1989). "The Quantization of Musical Time: A Connectionist Approach". In: *Computer Music Journal, 13(3)*, pp. 56–66 (cit. on p. 12).

— (1991). "Towards a Calculus for Expressive Timing in Music". In: *Computers in Music Research, 3(1)*, pp. 43–120 (cit. on p. 12).

Dhariwal, P., H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever (2020). "Jukebox: A Generative Model for Music". In: *CoRR abs/2005.00341* (cit. on pp. 2, 3, 30, 78, 79).

Dieleman, S., A. v. d. Oord, and K. Simonyan (2018). "The Challenge of Realistic Music Generation: Modelling Raw Audio at Scale". In: *Proceedings of the Neural Information Processing System (NIPS), Montréal, Canada*, pp. 8000–8010 (cit. on p. 29).

Dixon, S. (2001). "Automatic Extraction of Tempo and Beat from Expressive Performances". In: *Journal of New Music Research, 30(1)*, pp. 39–58 (cit. on p. 19).

— (2006). "Onset Detection Revisited". In: *Proceedings of the International Conference on Digital Audio Effects (DAFx), Montréal, Canada*, pp. 133–137 (cit. on pp. 17, 19, 66).

— (2007). "Evaluation of the Audio Beat Tracking System Beatroot". In: *Journal of New Music Research, 36(1)*, pp. 39–50 (cit. on p. 19).

Dixon, S., F. Gouyon, and G. Widmer (2004). "Towards Characterisation of Music via Rhythmic Patterns". In: *Proceedings of the International Conference on Music Information Retrieval (ISMIR), Barcelona, Spain*, pp. 509–516 (cit. on pp. 22, 104, 105).

Dixon, S., E. Pampalk, and G. Widmer (2003). "Classification of Dance Music by Periodicity Patterns". In: *Proceedings of the International Conference on Music Information Retrieval (ISMIR), Baltimore, USA*, pp. 159–165 (cit. on p. 104).

Doersch, C. (2016). "Tutorial on Variational Autoencoders". In: *CoRR abs/1606.05908* (cit. on p. 47).

Donahue, C., J. McAuley, and M. Puckette (2018). "Adversarial Audio Synthesis". In: *CoRR abs/1802.04208* (cit. on pp. 98, 101).

Drysdale, J., M. Tomczak, and J. Hockman (2020). "Adversarial Synthesis of Drum Sounds". In: *Proceedings of the International Conference on Digital Audio Effects (DAFx), Vienna, Austria*, pp. 167–172 (cit. on p. 85).

Dubey, S. R. and S. K. Singh (2023). "Transformer-based Generative Adversarial Networks in Computer Vision: A Comprehensive Survey". In: *CoRR abs/2302.08641* (cit. on p. 121).

Duchi, J., E. Hazan, and Y. Singer (2011). "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research (JMLR), 12(7)* (cit. on p. 39).

Durand, S., J. P. Bello, B. David, and G. Richard (2016). "Feature Adapted Convolutional Neural Networks for Downbeat Tracking". In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Shanghai, China*, pp. 296–300 (cit. on p. 20).

Duxbury, C., M. Sandler, and M. E. Davies (2002). "A Hybrid Approach to Musical Note Onset Detection". In: *Proceedings of the International Conference on Digital Audio Effects (DAFx), Hamburg, Germany*, pp. 33–38 (cit. on p. 16).

Duxbury, C. (2005). "Signal Models for Polyphonic Music". PhD thesis. Centre for Digital Music, Queen Mary University of London (cit. on p. 17).

Dziugaite, G. K., D. M. Roy, and Z. Ghahramani (2015). "Training Generative Neural Networks via Maximum Mean Discrepancy Optimization". In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), Amsterdam, Netherlands*, pp. 258–267 (cit. on p. 45).

Ellis, D. P. W. (2007). "Beat Tracking by Dynamic Programming". In: *Journal of New Music Research, 36(1)*, pp. 51–60 (cit. on p. 19).

Ellis, D. P. W. and J. Arroyo (2004). "Eigenrhythms: Drum Pattern Basis Sets for Classification and Generation". In: *Proceedings of the International Conference on Music Information Retrieval (ISMIR), Barcelona, Spain*, pp. 101–106 (cit. on pp. 22, 23).

Engel, J., K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts (2019). "GANSynth: Adversarial Neural Audio Synthesis". In: *Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, USA* (cit. on pp. 98, 101).

Engel, J., L. Hantrakul, C. Gu, and A. Roberts (2020). "DDSP: Differentiable Digital Signal Processing". In: *CoRR abs/2001.04643* (cit. on p. 3).

Engel, J., C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan (2017). "Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders". In: *Proceedings of the International Conference on Machine Learning (ICML), Sydney, Australia*, pp. 1068–1077 (cit. on pp. 29, 78–80).

Esling, P. and N. Devis (2020). "Creativity in the Era of Artificial Intelligence". In: *CoRR abs/2008.05959* (cit. on p. 122).

Eyben, F., S. Böck, B. Schuller, and A. Graves (2010). "Universal Onset Detection with Bidirectional Long Short-Term Memory Neural Networks". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Utrecht, Netherlands*, pp. 589–594 (cit. on pp. 16, 18).

Fletcher, N. H. and T. D. Rossing (1998). *The Physics of Musical Instruments*. Springer, New York, USA, pp. 583–622 (cit. on p. 11).

Fonseca, J., M. Fuentes, F. Bonini Baraldi, and M. E. P. Davies (2021). "On the use of automatic onset detection for the analysis of maracatu de baque solto". In: *Perspectives on Music, Sound and Musicology: Research, Education and Practice*, pp. 209–225 (cit. on p. 18).

Foote, D., D. Yang, and M. Rohaninejad (2016). *Do Androids Dream of Electric Beats?* Accessed 2 September 2017, AudioStyleTransfer.wordpress.com (cit. on pp. 28, 53).

Foote, J. (2000). "Automatic Audio Segmentation Using a Measure of Audio Novelty". In: *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), New York, USA*, pp. 452–455 (cit. on p. 16).

Foroughmand, H. and G. Peeters (2019). "Deep-rhythm for Tempo Estimation and Rhythm Pattern Recognition". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Delft, Netherlands* (cit. on p. 23).

Fraisse, P. (1982). "Rhythm and Tempo". In: *The Psychology of Music, 1(1)*, pp. 149–180 (cit. on p. 14).

— (1984). "Perception and Estimation of Time". In: *Annual Review of Psychology, 35(1)*, pp. 1–37 (cit. on p. 14).

Frane, A. V. (2017). "Swing Rhythm in Classic Drum Breaks From Hip-Hop's Breakbeat Canon". In: *Music Perception: An Interdisciplinary Journal, 34(3)*, pp. 291–302 (cit. on p. 13).

Freeman, P. and L. Lacey (2002). "Swing and Groove: Contextual Rhythmic Nuance in Live Performance". In: *Proceedings of the International Conference on Music Perception and Cognition (ICMPC), Sydney, Australia*, pp. 548–550 (cit. on p. 12).

Friberg, A. and A. Sundström (1999). "Jazz Drummers' Swing Ratio in Relation to Tempo". In: *Journal of the Acoustical Society of America (JASA), 105(2)*, pp. 1330–1330 (cit. on p. 12).

— (2002). "Swing Ratios and Ensemble Timing in Jazz Performance: Evidence for a Common Rhythmic Pattern". In: *Music Perception: An Interdisciplinary Journal, 19(3)*, pp. 333–349 (cit. on p. 12).

Gabrielsson, A. (1973a). "Adjective Ratings and Dimension Analyses of Auditory Rhythm Patterns. I". In: *Scandinavian Journal of Psychology, 14(1)*, pp. 244–260 (cit. on p. 22).

— (1973b). "Similarity Ratings and Dimension Analyses of Auditory Rhythm Patterns. II". In: *Scandinavian Journal of Psychology, 14(1)*, pp. 161–176 (cit. on p. 22).

Ganin, Y., E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky (2016). "Domain-Adversarial Training of Neural Networks". In: *Journal of Machine Learning Research (JMLR), 17(1)*, pp. 2030–2096 (cit. on pp. 79, 81).

Gärdenfors, P. (2004). *Conceptual Spaces: The Geometry of Thought*. MIT Press, Cambridge, USA (cit. on p. 22).

Gatys, L. A., A. S. Ecker, and M. Bethge (2015). "A Neural Algorithm of Artistic Style". In: *CoRR abs/1508.06576* (cit. on pp. 28, 33, 51, 53, 59, 61, 64).

Gillet, O. and G. Richard (2008). "Transcription and Separation of Drum Signals from Polyphonic Music". In: *IEEE Transactions on Audio, Speech, and Language Processing (TASLP), 16(3)*, pp. 529–540 (cit. on p. 21).

Gillick, J., A. Roberts, J. Engel, D. Eck, and D. Bamman (2019). "Learning to Groove with Inverse Sequence Transformations". In: *Proceedings of the International Conference on Machine Learning Research (PMLR), Long Beach, USA* (cit. on p. 27).

Glorot, X. and Y. Bengio (2010). "Understanding the Difficulty of Training Deep Feedforward Neural Networks". In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), Sardinia, Italy*, pp. 249–256 (cit. on pp. 83, 103).

Gonzalez, R. C. and P. Wintz (1987). *Digital Image Processing, Addison-Wesley, Boston, USA* (cit. on p. 67).

Goodfellow, I. J., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). "Generative Adversarial Nets". In: *Proceedings of the Neural Information Processing System (NIPS), Montréal, Canada*, pp. 2672–2680 (cit. on pp. 29, 48, 99, 101, 102).

Gordon, J. W. (1985). "Perception of Attack Transients in Musical Tones (Timing, Rhythm)". PhD thesis. Stanford University (cit. on p. 17).

Goto, M. (2001). "An Audio-based Real-time Beat Tracking System for Music with or Without Drum-sounds". In: *Journal of New Music Research, 30(2)*, pp. 159–171 (cit. on pp. 15, 20).

Goto, M. and Y. Muraoka (1994). "A Beat Tracking System for Acoustic Signals of Music". In: *Proceedings of the ACM International Conference on Multimedia (ACM-MM), San Francisco, USA*, pp. 365–372 (cit. on pp. 15, 19).

Gouyon, F. (2003). *Towards Automatic Rhythm Description of Musical Audio Signals. Representations, Computational Models and Applications*. PhD pre-thesis work (cit. on pp. 1, 15, 24).

— (2007). "Microtiming in "Samba de Roda"—Preliminary Experiments with Polyphonic Audio". In: *Simpósio da Sociedade Brasileira de Computação Musical São Paulo, Brazil* (cit. on p. 13).

Gouyon, F., L. Fabig, and J. Bonada (2003). "Rhythmic Expressiveness Transformations of Audio Recordings: Swing Modifications". In: *Proceedings of the International Conference on Digital Audio Effects (DAFx), Hamburg, Germany*, pp. 8–11 (cit. on pp. 13, 25).

Gouyon, F., F. Pachet, O. Delerue, et al. (2000). "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), Verona, Italy* (cit. on p. 21).

Gretton, A., K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola (2012). "A Kernel Two-sample Test". In: *Journal of Machine Learning Research (JMLR), 13(1)*, pp. 723–773 (cit. on p. 45).

Griffin, D. and J. Lim (1984). "Signal Estimation from Modified Short-time Fourier Transform". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing (TASSP), 32(2)*, pp. 236–243 (cit. on p. 103).

Griffin, G., Y. E. Kim, and D. Turnbull (2010). "Beat-Sync-Mash-Coder: A Web Application for Real-time Creation of Beat-synchronous Music Mashups". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Dallas, USA*, pp. 437–440 (cit. on p. 27).

Grinstein, E., N. Q. K. Duong, A. Ozerov, and P. Pérez (2017). "Audio Style Transfer". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, USA* (cit. on pp. 28, 57, 58, 64, 71, 74).

Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville (2017). "Improved Training of Wasserstein GANs". In: *Proceedings of the Neural Information Processing System (NIPS), Long Beach, USA*, pp. 5767–5777 (cit. on pp. 49, 100).

Gwern (2020). *This Waifu Does Not Exist*. Accessed 1 July 2020, https://www.gwern.net/TWDNE (cit. on p. 85).

Hawthorne, C., I. Simon, R. Swavely, E. Manilow, and J. Engel (2021). "Sequence-to-Sequence Piano Transcription with Transformers". In: *CoRR abs/2107.09142* (cit. on p. 18).

He, K., X. Zhang, S. Ren, and J. Sun (2016a). "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR), Las Vegas, USA* (cit. on p. 80).

He, K., Y. Wang, and J. Hopcroft (2016b). "A Powerful Generative Model Using Random Weights for the Deep Image Representation". In: *Proceedings of the Neural Information Processing System (NIPS), Barcelona, Spain* (cit. on p. 61).

Hennequin, R., A. Khlif, F. Voituret, and M. Moussallam (2020). "Spleeter: A Fast And State-of-the Art Music Source Separation Tool With Pre-trained Models". In: *Journal of Open Source Software (JOSS), 50(5)*, pp. 2154–2157 (cit. on pp. 103, 107).

Herrera, P., A. Yeterian, and F. Gouyon (2002). "Automatic Classification of Drum Sounds: A Comparison of Feature Selection Methods and Classification Techniques". In: *Proceedings of the International Conference on Music and Artificial Intelligence (ICMAI), Edinburgh, Scotland*, pp. 69–80 (cit. on p. 21).

Hewitt, M. (2009). *Composition for Computer Musicians*. Delmar-Publishing, Huntington Beach, USA (cit. on pp. 10, 12, 13).

Hinton, G. E. and R. R. Salakhutdinov (2006). "Reducing the Dimensionality of Data with Neural Networks". In: *Science, 313(5786)*, pp. 504–507 (cit. on p. 101).

Hochreiter, S. and J. Schmidhuber (1997). "Long Short-term Memory". In: *Neural Computation, 9(8)*, pp. 1735–1780 (cit. on p. 46).

Hockman, J. A. (2007). *Automatic Timbre Mutation of Drum Loops*. Master's Thesis, New York University, USA (cit. on pp. 9, 27).

Hockman, J. A., J. P. Bello, M. E. P. Davies, and M. D. Plumbley (2008). "Automated Rhythmic Transformation of Musical Audio". In: *Proceedings of the International Conference on Digital Audio Effects (DAFx), Helsinki, Finland*, pp. 177–180 (cit. on pp. 13, 19, 24–26, 104).

Hockman, J. A. and M. E. P. Davies (2015). "Computational Strategies for Breakbeat Classification and Resequencing in Hardcore, Jungle and Drum & Bass". In: *Proceedings of the International Conference on Digital Audio Effects (DAFx), Trondheim, Norway* (cit. on pp. xii, 25).

Hockman, J. (2014). "An Ethnographic and Technological Study of Breakbeats in Hardcore, Jungle and Drum & Bass". PhD thesis. Department of Music Research, Schulich School of Music, McGill University (cit. on pp. 2, 9, 20, 66).

Hockman, J., M. E. P. Davies, and I. Fujinaga (2012). "One in the Jungle: Downbeat Detection in Hardcore, Jungle, and Drum and Bass". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Porto, Portugal*, pp. 169–174 (cit. on pp. xvi, 107).

Huang, S., Q. Li, C. Anil, X. Bao, S. Oore, and R. B. Grosse (2018). "TimbreTron: A WaveNet (CycleGAN (CQT (Audio))) Pipeline for Musical Timbre Transfer". In: *Proceedings of the Neural Information Processing System (NIPS), Montréal, Canada* (cit. on p. 29).

ITU (1988). *Pulse Code Modulation (PCM) of Voice Frequencies*. International Telecommunication Union (ITU) (cit. on p. 82).

Ioffe, S. and C. Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the International Conference on Machine Learning (ICML), Lille, France*, pp. 448–456 (cit. on pp. 41, 101).

Ishizaki, H., K. Hoashi, and Y. Takishima (2009). "Full-Automatic DJ Mixing System with Optimal Tempo Adjustment based on Measurement Function of User Discomfort". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Kobe, Japan*, pp. 135–140 (cit. on pp. 27, 63).

Isola, P., J.-Y. Zhu, T. Zhou, and A. A. Efros (2017). "Image-to-image Translation with Conditional Adversarial Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, Hawaii*, pp. 1125–1134 (cit. on pp. 33, 52).

Janer, J., J. Bonada, and S. Jordà (2006). "Groovator - An Implementation of Real-Time Rhythm Transformations". In: *Proceedings of the Audio Engineering Society Conference (AES), San Francisco, USA* (cit. on p. 25).

Jehan, T. (1997). "Musical Signal Parameter Estimation". In: *The UC Berkeley Center for New Music and Audio Technologies (CNMAT)* (cit. on p. 17).

Ji, S., J. Luo, and X. Yang (2020). "A Comprehensive Survey on Deep Music Generation: Multi-level Representations, Algorithms, Evaluations, and Future Directions". In: *CoRR abs/2011.06801* (cit. on p. 27).

Jing, Y., Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song (2019). "Neural Style Transfer: A Review". In: *IEEE Transactions on Visualization and Computer Graphics, 26(11)*, pp. 3365–3385 (cit. on p. 64).

Jones, M. R. and M. Boltz (1989). "Dynamic Attending and Responses to Time". In: *Psychological Review, 96(3)*, pp. 459–491 (cit. on p. 14).

Jordan, M. I., Z. Ghahramani, T. S. Jaakkola, and L. K. Saul (1999). "An Introduction to Variational Methods for Graphical Models". In: *Machine learning, 37(2)*, pp. 183–233 (cit. on p. 47).

Kalchbrenner, N. et al. (2018). "Efficient Neural Audio Synthesis". In: *Proceedings of the International Conference on Machine Learning Research (PMLR), Baltimore, USA*, pp. 2410–2419 (cit. on p. 29).

Karras, T., M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila (2021). "Alias-free Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems (NIPS), 34(1)*, pp. 852–863 (cit. on p. 121).

Kim, J. W., R. Bittner, A. Kumar, and J. P. Bello (2019). "Neural Music Synthesis for Flexible Timbre Control". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK*, pp. 176–180 (cit. on pp. 29, 67, 88, 92).

Kingma, D. P. and J. Ba (2014). "Adam: A Method for Stochastic Optimization". In: *CoRR abs/1412.6980* (cit. on pp. 39, 103).

Kingma, D. P. and M. Welling (2013). "Auto-encoding Variational Bayes". In: *CoRR abs/1312.6114* (cit. on pp. 29, 47–49, 98, 101, 102, 109).

Kingma, D. P., S. Mohamed, D. Jimenez Rezende, and M. Welling (2014). "Semi-supervised Learning with Deep Generative Models". In: *Advances in Neural Information Processing Systems (NIPS), 27(1)* (cit. on p. 33).

Klambauer, G., T. Unterthiner, A. Mayr, and S. Hochreiter (2017). "Self-normalizing Neural Networks". In: *Advances in Neural Information Processing Systems (NIPS), 30(1)* (cit. on p. 35).

Klapuri, A. P., A. J. Eronen, and J. T. Astola (2006). "Analysis of the Meter of Acoustic Musical Signals". In: *IEEE Transactions on Audio, Speech, and Language Processing (TASLP), 14(1)*, pp. 342–355 (cit. on p. 20).

Klapuri, A. and M. Davy (2007). "Signal Processing Methods for Music Transcription". In: (cit. on p. 2).

Knees, P., K. Andersen, S. Jordà, M. Hlatky, A. Bucci, W. Gaebele, and R. Kaurson (2016). "The Giantsteps Project: A Second-year Intermediate Report". In: *Proceedings of the International Computer Music Conference (ICMC), Utrecht, Netherlands* (cit. on pp. 9, 24).

Knees, P., K. Andersen, S. Jordà, M. Hlatky, G. Geiger, W. Gaebele, and R. Kaurson (2015). "Giantsteps - Progress Towards Developing Intelligent and Collaborative Interfaces for Music Production and Performance". In: *Proceedings of the IEEE International Conference on Multimedia & Expo Workshops (ICMEW)* (cit. on pp. 3, 9, 23).

Krebs, F., S. Böck, and G. Widmer (2013). "Rhythmic Pattern Modeling for Beat and Downbeat Tracking in Musical Audio". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Curitiba, Brazil*, pp. 227–232 (cit. on pp. 23, 104).

Krebs, F., S. Böck, and G. Widmer (2015). "An Efficient State-Space Model for Joint Tempo and Meter Tracking". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Málaga, Spain*, pp. 72–78 (cit. on p. 20).

Krebs, F., F. Korzeniowski, M. Grachten, and G. Widmer (2014). "Unsupervised Learning and Refinement of Rhythmic Patterns for Beat and Downbeat Tracking". In: *Proceedings of the IEEE European Signal Processing Conference (EUSIPCO), Lisbon, Portugal*, pp. 611–615 (cit. on pp. 15, 20).

Lacoste, A. and D. Eck (2006). "A Supervised Classification Algorithm for Note Onset Detection". In: *EURASIP Journal on Advances in Signal Processing, 2007(1)*, pp. 1–13 (cit. on pp. 16, 18).

Lattner, S. and M. Grachten (2019). "High-Level Control of Drum Track Generation Using Learned Patterns of Rhythmic Interaction". In: *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, USA*, pp. 35–39 (cit. on p. 27).

Lerdahl, F. and R. Jackendoff (1983). *A Generative Theory of Tonal Music*. MIT Press, Cambridge, USA (cit. on p. 14).

Li, Y., N. Wang, J. Liu, and X. Hou (2017). "Demystifying Neural Style Transfer". In: *CoRR abs/1701.01036* (cit. on p. 60).

Li, Y., K. Swersky, and R. Zemel (2015). "Generative Moment Matching Networks". In: *Proceedings of the International Conference on Machine Learning Research (PMLR), San Diego, USA*, pp. 1718–1727 (cit. on p. 45).

Liu, D. C. and J. Nocedal (1989). "On the Limited Memory BFGS Method for Large Scale Optimization". In: *Mathematical Programming, 45(1)*, pp. 503–528 (cit. on p. 40).

London, J. (2012). *Hearing in Time: Psychological Aspects of Musical Meter*. Oxford University Press, Oxford, UK (cit. on p. 14).

MIREX (2016). *Music Information Retrieval Evaluation eXchange (MIREX): Audio Downbeat Estimation Evaluation*. Accessed 18 September 2019, `/2016:Audio_Downbeat_Estimation_Results` (cit. on p. 20).

— (2018a). *MIREX: Audio Onset Detection Evaluation*. Accessed 21 August 2018, `/mirex2018/results/aod/` (cit. on pp. 18, 68).

— (2018b). *MIREX: Automatic Drum Transcription Evaluation*. Accessed 21 August 2018, `/2018:Drum_Transcription_Results` (cit. on p. 21).

— (2019). *MIREX: Audio Beat Tracking Evaluation*. Accessed 18 September 2019, `/mirex2019/results/abt/smc/` (cit. on p. 19).

Maas, A. L., A. Y. Hannun, A. Y. Ng, et al. (2013). "Rectifier Nonlinearities Improve Neural Network Acoustic Models". In: *Proceedings of the International Conference on Machine Learning (ICML), Atlanta, USA* (cit. on p. 35).

Major, M. (2014). *Recording Drums: The Complete Guide*. Nelson Education, Toronto, Canada (cit. on pp. 11, 92).

Makhzani, A., J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey (2015). "Adversarial Autoencoders". In: *CoRR abs/1511.05644* (cit. on pp. 49, 97, 98, 102).

Manzelli, R., V. Thakkar, A. Siahkamari, and B. Kulis (2018). "Conditioning Deep Generative Raw Audio Models for Structured Automatic Music". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Paris, France* (cit. on p. 29).

Marín, D. G. (2018). "Similarity and Style in Electronic Dance Music Drum Rhythms". PhD thesis. Department of Information and Communication Technologies, Pompeu Fabra University (cit. on p. 10).

Masri, P. (1996). "Computer Modelling of Sound for Transformation and Synthesis of Musical Signals." PhD thesis. Electrical & Electronic Engineering, University of Bristol (cit. on p. 16).

Masri, P. and A. Bateman (1996). "Improved Modelling of Attack Transients in Music Analysis-Resynthesis". In: *Proceedings of the International Computer Music Conference (ICMC), Hong Kong* (cit. on p. 17).

Mauch, M. and S. Dixon (2012). "A Corpus-based Study of Rhythm Patterns". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Porto, Portugal*, pp. 163–168 (cit. on p. 104).

McAulay, R. and T. Quatieri (1986). "Speech Analysis/Synthesis Based on a Sinusoidal Representation". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing (TASSP), 34(4)*, pp. 744–754 (cit. on p. 16).

McFee, B., C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto (2015). "librosa: Audio and Music Signal Analysis in Python". In: *Proceedings of the Python in Science Conference (SciPy), Texas, USA* (cit. on pp. 82, 89).

Mehri, S., K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio (2016). "SampleRNN: An Unconditional End-to-end Neural Audio Generation Model". In: *CoRR abs/1612.07837* (cit. on pp. 29, 52).

Meseguer-Brocal, G., A. Cohen-Hadria, and G. Peeters (2018). "DALI: A Large Dataset of Synchronized Audio, Lyrics and Notes, Automatically Created Using Teacher-Student Machine Learning Paradigm".

In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Paris, France* (cit. on pp. xvi, 107).

Mital, P. K. (2017). "Time Domain Neural Audio Style Transfer". In: *Proceedings of the Neural Information Processing System (NIPS), Long Beach, USA* (cit. on pp. 29, 57, 58, 64, 73, 74).

Moelants, D. (2002). "Preferred Tempo Reconsidered". In: *Proceedings of the International Conference on Music Perception and Cognition (ICMPC), Sydney, Australia*, pp. 1–4 (cit. on p. 63).

Mor, N., L. Wolf, A. Polyak, and Y. Taigman (2018). "A Universal Music Translation Network". In: *CoRR abs/1805.07848* (cit. on pp. 30, 78, 80).

— (2019). "A Universal Music Translation Network". In: *Proceedings of the International Conference on Learning Representations (ICLR), Vancouver, Canada* (cit. on pp. 30, 78–82).

Moriarty, D. E. and R. Mikkulainen (1996). "Efficient Reinforcement Learning Through Symbiotic Evolution". In: *Machine Learning, 22(1)*, pp. 11–32 (cit. on p. 40).

Muandet, K., K. Fukumizu, B. Sriperumbudur, and B. Schölkopf (2016). "Kernel Mean Embedding of Distributions: A Review and Beyond". In: *Foundations and Trends® in Machine Learning, 10(1–2)*, pp. 1–141 (cit. on pp. 44, 45).

Nagarajan, V. and J. Z. Kolter (2017). "Gradient Descent GAN Optimization is Locally Stable". In: *Proceedings of the Neural Information Processing System (NIPS), Long Beach, USA*, pp. 5585–5595 (cit. on p. 99).

Nakano, T., M. Goto, J. Ogata, and Y. Hiraga (2005). "Voice Drummer: A Music Notation Interface of Drum Sounds Using Voice Percussion Input". In: *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST), Seattle, USA*, pp. 49–50 (cit. on p. 21).

Nieto, O., M. McCallum, M. E. P. Davies, A. Robertson, A. Stark, and E. Egozy (2019). "The Harmonix Set: Beats, Downbeats, and Functional Segment Annotations of Western Popular Music". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Delft, Netherlands* (cit. on pp. xvi, 107).

Novikov, A. (2019). "PyClustering: Data Mining Library". In: *Journal of Open Source Software (JOSS), 4(36)*, pp. 1230–1234 (cit. on p. 104).

Nowozin, S., B. Cseke, and R. Tomioka (2016). "f-GAN: Training Generative Neural Samplers Using Variational Divergence Minimization". In: *Proceedings of the Neural Information Processing System (NIPS), Barcelona, Spain*, pp. 271–279 (cit. on pp. 44, 49).

Odena, A., V. Dumoulin, and C. Olah (2016). *Deconvolution and Checkerboard Artifacts*. Accessed December 2016, https://distill.pub/2016/deconv-checkerboard/ (cit. on p. 101).

Oord, A. v. d., S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu (2016a). "WaveNet: A Generative Model for Raw Audio". In: *CoRR abs/1609.03499* (cit. on pp. 29, 52, 78, 79, 81, 82).

Oord, A. v. d., N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu (2016b). "Conditional Image Generation with PixelCNN Decoders". In: *Proceedings of the Neural Information Processing System (NIPS), Barcelona, Spain* (cit. on p. 46).

Oord, A. v. d. et al. (2018). "Parallel WaveNet: Fast High-fidelity Speech Synthesis". In: *Proceedings of the International Conference on Machine Learning Research (PMLR), Baltimore, USA*, pp. 3918–3926 (cit. on p. 117).

Owsinski, B. (2017). *The Recording Engineer's Handbook*. Hal Leonard Corporation, Milwaukee, USA (cit. on p. 11).

Owsinski, B. and D. Moody (2009). *The Drum Recording Handbook*. Hal Leonard Corporation, Milwaukee, USA (cit. on p. 11).

Pampalk, E., P. Herrera, and M. Goto (2008). "Computational Models of Similarity for Drum Samples". In: *IEEE Transactions on Audio, Speech, and Language Processing (TASLP), 16(2)*, pp. 408–423 (cit. on p. 21).

Paulus, J. and A. Klapuri (2002). "Measuring the Similarity of Rhythmic Patterns". In: *Proceedings of the International Conference on Music Information Retrieval (ISMIR), Baltimore, USA* (cit. on pp. 10, 15, 22).

— (2009). "Drum Sound Detection in Polyphonic Music with Hidden Markov Models". In: *EURASIP Journal on Audio, Speech, and Music Processing, 2009(1)*, pp. 1–9 (cit. on p. 21).

Paulus, J. and T. Virtanen (2005). "Drum Transcription with Non-negative Spectrogram Factorisation". In: *Proceedings of the IEEE European Signal Processing Conference (EUSIPCO), Antalya, Turkey*, pp. 1–4 (cit. on p. 21).

Peeters, G. (2005). "Rhythm Classification Using Spectral Rhythm Patterns". In: *Proceedings of the International Conference on Music Information Retrieval (ISMIR), London, UK*, pp. 644–647 (cit. on pp. 22, 23, 104).

Peeters, G. and H. Papadopoulos (2011). "Simultaneous Beat and Downbeat-tracking Using a Probabilistic Framework: Theory and Large-scale Evaluation". In: *IEEE Transactions on Audio, Speech, and Language Processing (TASLP), 19(6)*, pp. 1754–1769 (cit. on p. 20).

Pelleg, D. and A. W. Moore (2000). "X-means: Extending K-means with Efficient Estimation of the Number of Clusters". In: *Proceedings of the International Conference on Machine Learning (ICML), Stanford, USA*, pp. 727–734 (cit. on p. 104).

Perez, A., C. Proctor, and A. Jain (2017). *Style Transfer for Prosodic Speech*. Tech. rep. Stanford University (cit. on pp. 29, 64, 74).

Pikrakis, A. (2013). "A Deep Learning Approach to Rhythm Modeling with Applications". In: *Proceedings of the International Workshop on Machine Learning and Music (MML), Prague, Czech Republic* (cit. on p. 22).

Pons, J., R. Gong, and X. Serra (2017). "Score-informed Syllable Segmentation for a Cappella Singing Voice with Convolutional Neural Networks". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Suzhou, China*, pp. 483–489 (cit. on p. 18).

Pressing, J. (2002). "Black Atlantic Rhythm: Its Computational and Transcultural Foundations". In: *Music Perception, 19(3)*, pp. 285–310 (cit. on p. 12).

Ramírez, M. A. M., O. Wang, P. Smaragdis, and N. J. Bryan (2021). "Differentiable Signal Processing with Black-Box Audio Effects". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, Canada*, pp. 66–70 (cit. on p. 67).

Ravelli, E., J. P. Bello, and M. Sandler (2007). "Automatic Rhythm Modification of Drum Loops". In: *IEEE Signal Processing Letters, 14(4)*, pp. 228–231 (cit. on pp. 13, 24, 25).

Razavi, A., A. v. d. Oord, and O. Vinyals (2019). "Generating Diverse High-fidelity Images with VQ-VAE-2". In: *Advances in Neural Information Processing Systems (NIPS), 32(1)* (cit. on p. 30).

Reiss, J. D. and Ø. Brandtsegg (2018). "Applications of Cross-adaptive Audio Effects: Automatic Mixing, Live Performance and Everything in Between". In: *Frontiers in Digital Humanities, 5(1)*, pp. 1–17 (cit. on p. 19).

Repp, B. H. (2005). "Sensorimotor Synchronization: A Review of the Tapping Literature". In: *Psychonomic Bulletin & Review, 12(6)*, pp. 969–992 (cit. on p. 14).

Reynolds, S. (2012). *Energy Flash: A Journey Through Rave Music and Dance Culture*. Soft Skull Press, New York, USA (cit. on pp. 2, 25).

Robertson, A. (2009). "Interactive Teal-time Musical Systems". PhD thesis. School of Electronic Engineering and Computer Science, Queen Mary University of London (cit. on p. 13).

Roma, G. (2008). "Freesound Radio: Supporting Collective Organization of Sounds". PhD thesis. Department of Information and Communication Technologies, Pompeu Fabra University (cit. on p. 16).

Rosenthal, D. F. (1992). "Machine Rhythm: Computer Emulation of Human Rhythm Perception". PhD thesis. Massachusetts Institute of Technology (cit. on p. 14).

Rossing, T. D. (2001). *Science of Percussion Instruments*. World Scientific Publishing, Singapore (cit. on p. 11).

Rossing, T. D., F. R. Moore, and P. A. Wheeler (2014). *The Science of Sound*. Pearson Publishing, London, UK (cit. on p. 11).

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). "Learning Representations by Back-propagating Errors". In: *Nature, 323(6088)*, pp. 533–536 (cit. on p. 39).

Salimans, T., I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen (2016). "Improved Techniques for Training GANs". In: *Advances in Neural Information Processing Systems (NIPS), 29(1)*, pp. 2234–2242 (cit. on p. 49).

Sawada, S., S. Fukayama, M. Goto, and K. Hirata (2019). "TransDrums: A Drum Pattern Transfer System Preserving Global Pattern Structure". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK*, pp. 391–395 (cit. on p. 26).

Scheirer, E. D. (1998). "Tempo and Beat Analysis of Acoustic Musical Signals". In: *Journal of the Acoustical Society of America (JASA), 103(1)*, pp. 588–601 (cit. on p. 19).

Schloss, W. A. (1985). "On the Automatic Transcription of Percussive Music from Acoustic Signal to High-level Analysis". PhD thesis. Center for Computer Research in Music and Acoustics (CCRMA), Stanford University (cit. on pp. 17, 21).

Schlüter, J. and S. Böck (2013). "Musical Onset Detection with Convolutional Neural Networks". In: *Proceedings of the International Workshop on Machine Learning and Music (MML) held in conjunction with the European Conference on Machine Learning and Principles (ECML) and Practice of Knowledge Discovery in Databases (PKDD) Prague, Czech Republic* (cit. on p. 18).

— (2014). "Improved Musical Onset Detection with Convolutional Neural Networks". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy*, pp. 6979–6983 (cit. on p. 16).

Schörkhuber, C. and A. Klapuri (2010). "Constant-Q Transform Toolbox for Music Processing". In: *Proceedings of the Sound and Music Computing Conference (SMC), Barcelona, Spain*, pp. 1–8 (cit. on p. 88).

Schrader, B. (1982). *Introduction to Electro-acoustic Music*. Prentice Hall, Hoboken, USA (cit. on pp. 2, 9).

Serra, X. (1989). "A System for Sound Analysis/Transformation/Synthesis Based on a Deterministic Plus Stochastic Decomposition". PhD thesis. Center for Computer Research in Music and Acoustics (CCRMA), Stanford University (cit. on p. 9).

Shiga, J. (2007). "Copy-and-persist: The Logic of Mash-up Culture". In: *Critical Studies in Media Communication (CSMC), 24(2)*, pp. 93–114 (cit. on p. 27).

Simonyan, K. and A. Zisserman (2014). "Very Deep Convolutional Networks for Large-scale Image Recognition". In: *CoRR abs/1409.1556* (cit. on p. 101).

Snoman, R. (2012). *The Dance Music Manual: Tools, Toys and Techniques*. CRC Press, Boca Raton, USA (cit. on pp. 10, 13).

Southall, C. (2019). "Automatic Drum Transcription Using Deep Learning". PhD thesis. Sound and Music Analysis (SOMA) Group, Digital Media Technology (DMT) Lab, Birmingham City University (cit. on pp. 10, 20, 67).

Southall, C., R. Stables, and J. Hockman (2016). "Automatic Drum Transcription Using Bi-Directional Recurrent Neural Networks". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), New York City, USA* (cit. on pp. 21, 67).

— (2017). "Automatic Drum Transcription for Polyphonic Recordings Using Soft Attention Mechanisms and Convolutional Neural Networks". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Suzhou, China*, pp. 606–612 (cit. on pp. 21, 88).

— (2018). "Player Vs Transcriber: A Game Approach To Data Manipulation For Automatic Drum Transcription". In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Paris, France*, pp. 58–65 (cit. on p. 21).

Sriperumbudur, B. K., K. Fukumizu, A. Gretton, B. Schölkopf, and G. R. G. Lanckriet (2009). "On Integral Probability Metrics, ϕ-Divergences and Binary Classification". In: *CoRR abs/0901.2698* (cit. on p. 44).

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research (JMLR), 15(1)*, pp. 1929–1958 (cit. on p. 42).

Steinwart, I. and A. Christmann (2008). *Support Vector Machines*. Springer Science & Business Media, Berlin, Germany (cit. on p. 44).

Stevens, S. S. and J. Volkmann (1940). "The Relation of Pitch to Frequency: A Revised Scale". In: *The American Journal of Psychology (AJP), 53(3)*, pp. 329–353 (cit. on p. 66).

Stewart, A. (2000). "'Funky Drummer': New Orleans, James Brown and the Rhythmic Transformation of American Popular Music". In: *Popular Music, 19(3)*, pp. 293–318 (cit. on p. 12).

Sutton, R. (1986). "Two Problems with Back Propagation and Other Steepest Descent Learning Procedures for Networks". In: *Proceedings of the Conference of the Cognitive Science Society (CogSci), Amherst, USA*, pp. 823–832 (cit. on p. 39).

Taylor, G., R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein (2016). "Training Neural Networks Without Gradients: A Scalable ADMM Approach". In: *Proceedings of the International Conference on Machine Learning Research (PMLR), New York, USA*, pp. 2722–2731 (cit. on p. 40).

Temperley, D. (1999). "Syncopation in Rock: A Perceptual Perspective". In: *Popular Music, 18(1)*, pp. 19–40 (cit. on p. 13).

— (2004). *The Cognition of Basic Musical Structures*. MIT Press, Cambridge, USA (cit. on p. 14).

Tenney, J. and L. Polansky (1980). "Temporal Gestalt Perception in Music". In: *Journal of Music Theory, 24(2)*, pp. 205–241 (cit. on p. 14).

Termens, E. G. (2004). "New Approaches for Rhythmic Description of Audio Signals". PhD thesis. Department of Information and Communication Technologies, Pompeu Fabra University (cit. on p. 15).

Thul, E. and G. T. Toussaint (2008). "Rhythm Complexity Measures: A Comparison of Mathematical Models of Human Perception and Performance". In: *Proceedings of the International Conference on Music Information Retrieval (ISMIR), Philadelphia, USA*, pp. 663–668 (cit. on p. 22).

Tieleman, T., G. Hinton, et al. (2012). "Lecture 6.5-RMSProp: Divide the Gradient by a Running Average of Its Recent Magnitude". In: *COURSERA: Neural Networks for Machine Learning, 4(2)*, pp. 26–31 (cit. on p. 39).

Tingle, D., Y. E. Kim, and D. Turnbull (2010). "Exploring Automatic Music Annotation with 'Acoustically-Objective' Tags". In: *Proceedings of the International Conference on Multimedia Information Retrieval, New York, USA*, pp. 55–62 (cit. on p. 3).

Toh, C.-C., B. Zhang, and Y. Wang (2008). "Multiple-Feature Fusion Based Onset Detection for Solo Singing Voice". In: *Proceedings of the International Conference on Music Information Retrieval (ISMIR), Philadelphia, USA*, pp. 515–520 (cit. on p. 18).

Tokui, N. (2008). "Massh!: A Web-based Collective Music Mashup System". In: *Proceedings of the International Conference on Digital Interactive Media in Entertainment and Arts (DIMEA), Athens, Greece*, pp. 526–527 (cit. on p. 27).

Tomczak, M., J. Drysdale, and J. Hockman (2019). "Drum Translation for Timbral and Rhythmic Transformation". In: *Proceedings of the International Conference on Digital Audio Effects (DAFx), Birmingham, UK*, pp. 340–346 (cit. on p. 54).

Tomczak, M., M. Goto, and J. Hockman (2020). "Drum Synthesis and Rhythmic Transformation with Adversarial Autoencoders". In: *Proceedings of the ACM International Conference on Multimedia (ACM-MM), Seattle, USA*, pp. 2427–2435 (cit. on pp. xvi, 54, 107).

Tomczak, M., C. Southall, and J. Hockman (2018). "Audio Style Transfer with Rhythmic Constraints". In: *Proceedings of the International Conference on Digital Audio Effects (DAFx), Aveiro, Portugal*, pp. 45–50 (cit. on p. 53).

Toussaint, G. T. (2004). "A Comparison of Rhythmic Similarity Measures". In: *Proceedings of the International Conference on Music Information Retrieval (ISMIR), Barcelona, Spain* (cit. on p. 22).

Tschannen, M., O. Bachem, and M. Lucic (2018). "Recent Advances in Autoencoder-based Representation Learning". In: *CoRR abs/1812.05069* (cit. on p. 120).

Tzanetakis, G., A. Kapur, and R. I. McWalter (2005). "Subband-based Drum Transcription for Audio Signals". In: *Proceedings of the IEEE Workshop on Multimedia Signal Processing, Shanghai, China*, pp. 1–4 (cit. on p. 21).

UMMA (2017). *Matisse Drawings: Curated by Ellsworth Kelly from The Pierre and Tana Matisse Foundation Collection*. Accessed 12 December 2018, https://umma.umich.edu/news/2017/umma-presents-matisse-kelly-drawings (cit. on p. 59).

Ulyanov, D. and V. Lebedev (2016). *Audio Texture Synthesis and Style Transfer*. Accessed 3 December 2016, https://dmitryulyanov.github.io/ (cit. on pp. 28, 51, 53, 56, 58, 64, 73, 74).

Uria, B., I. Murray, and H. Larochelle (2014). "A Deep and Tractable Density Estimator". In: *Proceedings of the International Conference on Machine Learning (ICML), Beijing, China*, pp. 467–475 (cit. on p. 46).

Valenti, A., A. Carta, and D. Bacciu (2020). "Learning a Latent Space of Style-Aware Symbolic Music Representations by Adversarial Autoencoders". In: *CoRR abs/2001.05494* (cit. on pp. 27, 102).

Veire, L. V., T. De Bie, and J. Dambre (2019). "A CycleGAN for Style Transfer Between Drum and Bass Subgenres". In: *Proceedings of the Machine Learning for Music Discovery Workshop at the International Conference on Machine Learning (ICML), Long Beach, USA*, pp. 1–3 (cit. on p. 30).

Verfaille, V., U. Zölzer, and D. Arfib (2006). "Adaptive Digital Audio Effects (A-DAFx): A New Class of Sound Transformations". In: *IEEE Transactions on Audio, Speech, and Language Processing (TASLP), 14(5)*, pp. 1817–1831 (cit. on p. 24).

Verma, P. and J. O. Smith (2018). "Neural Style Transfer for Audio Spectrograms". In: *Proceedings of the Neural Information Processing System (NIPS), Montréal, Canada* (cit. on pp. 28, 29, 64, 74).

Verma, T. S., S. N. Levine, and T. H. Y. Meng (1997). "Transient Modeling Synthesis: A Flexible Analysis/Synthesis Tool for Transient Signals". In: *Proceedings of the International Computer Music Conference (ICMC), Thessaloniki, Greece* (cit. on p. 16).

Vogl, R., M. Dorfer, G. Widmer, and P. Knees (2017). "Drum Transcription via Joint Beat and Drum Modeling Using Convolutional Recurrent Neural Networks." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Suzhou, China*, pp. 150–157 (cit. on p. 21).

Wilmering, T., G. Fazekas, and M. B. Sandler (2013). "The Audio Effects Ontology." In: *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Curitiba, Brazil*, pp. 215–220 (cit. on p. 3).

Witek, M. A. G., E. F. Clarke, M. Wallentin, M. L. Kringelbach, and P. Vuust (2014). "Syncopation, Body-movement and Pleasure in Groove Music". In: *PLOS One: Public Library of Science, 9(4)*, pp. 1–12 (cit. on p. 14).

Wu, C.-W., C. Dittmar, C. Southall, R. Vogl, G. Widmer, J. Hockman, M. Müller, and A. Lerch (2018). "A Review of Automatic Drum Transcription". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP), 26(9)*, pp. 1457–1483 (cit. on pp. 2, 15, 20, 21, 67).

Wu, C.-W. and A. Lerch (2015). "Drum Transcription Using Partially Fixed Non-negative Matrix Factorization". In: *Proceedings of the IEEE European Signal Processing Conference (EUSIPCO), Nice, France*, pp. 1281–1285 (cit. on pp. 21, 67).

Wu, Y.-K., C.-Y. Chiu, and Y.-H. Yang (2022). "JukeDrummer: Conditional Beat-aware Audio-domain Drum Accompaniment Generation via Transformer VQ-VA". In: *CoRR abs/2210.06007* (cit. on p. 30).

Wyse, L. (2017). "Audio Spectrogram Representations for Processing with Convolutional Neural Networks". In: *Proceedings of the International Conference on Deep Learning and Music at the International Joint Conference on Neural Networks (IJCNN), Anchorage, USA* (cit. on pp. 28, 64).

Yang, T., Y. Wang, Y. Lv, and N. Zh (2023). "DisDiff: Unsupervised Disentanglement of Diffusion Probabilistic Models". In: *CoRR abs/2301.13721* (cit. on p. 120).

Yoshii, K., M. Goto, K. Komatani, T. Ogata, and H. G. Okuno (2007). "Drumix: An Audio Player with Real-time Drum-part Rearrangement Functions for Active Music Listening". In: *Information and Media Technologies, 2(2)*, pp. 601–611 (cit. on p. 25).

Yu, F. and V. Koltun (2016). "Multi-scale Context Aggregation by Dilated Convolutions". In: *CoRR abs/1511.07122* (cit. on p. 37).

Yu, R. (2020). "A Tutorial on VAEs: From Bayes' Rule to Lossless Compression". In: *CoRR abs/2006.10273* (cit. on p. 47).

Zeiler, M. D. (2012). "ADADELTA: An Adaptive Learning Rate Method". In: *CoRR abs/1212.5701* (cit. on p. 39).

Zeiler, M. D. and R. Fergus (2014). "Visualizing and Understanding Convolutional Networks". In: *European Conference on Computer Vision (ECCV), Zürich, Switzerland*, pp. 818–833 (cit. on p. 37).

Zhang, Z., Y. Song, and H. Qi (2017). "Age Progression/Pegression by Conditional Adversarial Autoencoder". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, Hawaii*, pp. 5810–5818 (cit. on p. 97).

Zhu, C., R. H. Byrd, P. Lu, and J. Nocedal (1997). "Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-scale Bound-constrained Optimization". In: *ACM Transactions on Mathematical Software, 23(4)*, pp. 550–560 (cit. on p. 61).

Zhu, J.-Y., T. Park, P. Isola, and A. A. Efros (2017). "Unpaired Image-to-image Translation Using Cycle-consistent Adversarial Networks". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy*, pp. 2223–2232 (cit. on pp. 29, 30, 52).

Zölzer, U. (2011). *DAFX: Digital Audio Effects*. John Wiley & Sons, New York, USA (cit. on pp. 9, 24).