Deep Learning Methods for Sample-based Electronic Music

Jake Drysdale

19th May 2023

A thesis submitted 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

Sample-based electronic music (SBEM) encompasses various genres centred around the practice of sampling—the act of repurposing existing audio to create new music. Contemporary SBEM production involves navigating digital collections of audio, which include both libraries of samples and recorded music. The proliferation of digital music access, sample libraries, and online resource services have introduced challenges in navigating and managing these extensive collections of musical material. Selecting suitable samples from these sources is a meticulous and time-consuming task, requiring music producers to employ aesthetic judgement. Despite technological advancements, many SBEM producers still utilise laborious methods—established decades ago—for obtaining and manipulating music samples. This thesis proposes deep learning, a subfield of machine learning that develops algorithms to decipher intricate data relationships without explicit programming, as a potential solution. This research primarily explores the potential of deep learning models in SBEM, with a specific focus on developing automated tools for the analysis and generation of electronic music samples, towards enriching the creative experience for music producers.

To this end, a novel deep learning system designed for automatic instrumentation role classification in SBEM is introduced. This system identifies samples based on their specific roles within a composition—such as melody, bass, and drums—and exhibits versatility across various SBEM production tasks. Through a series of experiments, the capacity of the system to automatically label unstructured sample collections, generate high-level summaries of SBEM arrangements, and retrieve samples with desired characteristics from existing recordings is demonstrated. Additionally, a neural audio synthesis system that facilitates the continuous exploration and interpolation of sounds generated from a collection of drum samples is presented. This system employs a generative adversarial network (GAN), further modified to interact with the generated outputs. The evaluations highlight the effectiveness of the proposed conditional style-based GAN in generating a diverse range of high-quality drum samples. Various systematic approaches for interacting with the network and navigating the generative space are investigated, demonstrating novel methods of sample manipulation. Collectively, these contributions aim to foster further exploration and advancements at the intersection of deep learning and SBEM.

Acknowledgements

I extend my profound gratitude to my supervisor, Jason Hockman, for presenting me with the immense opportunity to conduct research on a subject I am passionate about. His guidance, feedback, expert knowledge, unyielding belief in my capabilities, and inspiring words of motivation during challenging times have been invaluable. I would like to thank my supervisory team Ryan Stables and Cham Athwal for their support throughout my PhD journey, which has significantly helped develop my research. I am deeply grateful for the thoughtful review and recommendations for enhancements offered by my dissertation committee— Mick Grierson, Ian Williams, and Franco Cheung. A massive thank you to all the past and present members of the DMT lab and SoMA group—Jason Hockman, Ryan Stables, Cham Athwal, Matthew Cheshire, Maciej Tomczak, Mattia Colombo, Sam Smith, Sean Enderby, Nicholas Jillings, Niccolò Granieri, Carl Southall, Islah Ali-MacLachlan, Tychonas Michailidis, Alastair Jamieson, Christina Karpodini, Alan Dolhasz, William Wilson, Ben West, and Yonghao Wang. There have been moments when each of you has provided indispensable insights, inspiring conversations, and unwavering support.

I would like to thank all the researchers I have worked with—Jason Hockman, Maciej Tomczak, António Ramires, Matthew Cheshire, Patricio López-Serrano, and Xavier Serra—working with each one of you has been a pleasure. To Matthew Davies, Jordan Smith, Philippe Esling, Javier Nistal, Antoine Lavault, and Eduardo Fonseca for the fruitful research discussions, which have been a source of inspiration and have greatly enriched my understanding of the field.

This thesis would not have been possible without the emotional support and motivation from the family and friends, I am so lucky to have around me. To Harriet, my girlfriend, thank you for your patience, for putting up with me and for always being there when times have been tough. I am eternally grateful to my parents, Sarah and Paul Drysdale, whose love and support have been my backbone. A big thanks and love to my Nan and Grandparents. A special thanks to my brother Jimi, and the Saturn Sound collective—Isaac, Adam, Lewis, Levi, and Jacob, your friendship and support has made this journey more enjoyable.

Contents

List of Figures

List of Tables

Acronyms

AIRC	automatic instrumentation role classification
AdaIN	adaptive instance normalisation
ADT	automatic drum transcription
AI	artificial intelligence
AMT	automatic music tagging
AR	autoregressive
BCE	binary cross-entropy
BPM	beats per minute
CNN	convolutional neural network
DAW	digital audio workstation
DGM	deep generative model
DJ	disc jockey
EM	electronic music
EQ	equaliser
FSLD	Freesound loop dataset
GAN	generative adversarial network
GP	gradient penalty
GPU	graphics processing unit
HJDB	hardcore, jungle, and drum and bass
IRAM	instrumentation role activation map
LAT	loop activation transcription
MIREX	music information retrieval evaluation exchange
MIR	music information retrieval
MLP	multi-layer perceptron
MPC	music production center

MSE	mean square error
NMF	non-negative matrix factorisation
NTF	non-negative tensor factorisation
PCA	principal component analysis
PR-AUC	area under precision-recall curve
ROC-AUC	area under receiver operating characteristic curve
SBEM	sample-based electronic music
SOM	self-organising maps
STFT	short-time Fourier transform
t-SNE	student-t stochastic neighbour embedding
UMAP	uniform manifold approximation and projection
VAE	variational autoencoder
VDS	virtual drumming software
VST	virtual studio technology
WGAN	Wasserstein generative adversarial network
WGAN-GP	Wasserstein GAN with gradient penalty

Mathematical Notation

x	A vector
X	A matrix
$\|x\|$	The absolute value of x
x_i	The i-th element of vector x
X_{i*}	The i-th row of matrix X
X_{*j}	The j-th column of matrix X
X^T	the transpose of matrix X
X^{-1}	the inverse of matrix X
$x \in X$	Set membership, x is an element of X
$X \in \mathbb{R}^{m,n}$	X is an m by n matrix containing real numbers
$x \leftarrow x + 1$	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 of x and y
$[0, 1]$	Range with minimum value of 0 and maximum value of 1
$\min_k, \max_k$	Extrema operations with respect to an integer value k

Chapter 1

Introduction

The transformative practice of repurposing existing audio material to create new music——known as *sampling*——has revolutionised music production and continues to be an essential technique in the present day. Driven by advancements in digital audio technology, sampling has substantially influenced the development of modern music, sparking a cultural shift within the industry. Affordable tools for audio production and digital sampling have enabled individuals with varying musical expertise to explore music creation through the use of samples. As a result, sample-based electronic music (SBEM) has gained widespread popularity in recent years, emphasising the growing importance of sampling in the contemporary music landscape. One of the key attributes of SBEM, and a main justification for the relevance of this work, is its continual appropriation of new technology used for creative purposes. The creative possibilities in SBEM are virtually boundless, and deep learning models have emerged as potent tools for enhancing and streamlining the production process. In recent years, the fields of music information retrieval (MIR) and deep learning have made significant progress in providing solutions and methodologies to various music-related challenges. Nevertheless, there is untapped potential for developing specialised tools that simplify complex music creation paradigms and workflows in contemporary production, working towards enriching the creative experience for music producers.

The roots of sampling can be traced back to the 1940s with the pioneering tape recording experiments led by Pierre Schaeffer, a prominent figure in the evolution of modern music. Schaeffer is acclaimed for his contributions to musique concrète, an innovative approach that seamlessly merged science and engineering with artistic expression and compositional methods (Holmes, 2012). The techniques and practices rooted in musique concrète gained broader exposure through their integration into popular television. A prime example is Delia Derbyshire's adaptation and realisation of Ron Grainer's title theme for the television series Doctor Who in 1963, which brought musique concrète and tape manipulation to a wider audience (Butler, 2014). As early sample-based instrument technologies emerged, influential artists such as The Beatles and David Bowie began incorporating sampling into their work (Griffin, 2016; Heyman, 2021). In the early 1970s, sampling approaches were further pioneered through the development of *dub* music. This innovative method involved remixing existing recordings, creatively applying electronic effects, and transforming studio technologies into instruments for composition and real-time improvisation. Key figures in this movement included Jamaican recording engineers Osbourne Ruddock (known as "King Tubby"), Errol Thompson, and Lee "Scratch" Perry (Veal, 2013).

This laid the foundation for sampling to gain further prominence during the rise of hip-hop in the 1980s when digital sampling technologies became more affordable and accessible to home producers. This democratisation of technology, along with the willingness of artists to explore creative possibilities, significantly impacted the landscape of music. Sampling allowed music creators to traverse the timeline of pre-existing music, connecting and fusing musical ideas from across the globe in ways previously unattainable before the rise of this technology (Reynolds, 2011). From the inception of hip-hop music to the present day, percussion solos from 1960s and 1980s jazz and funk——known as *breakbeats*——have remained highly sought-after samples for many music producers (Ewoodzie Jr, 2017). Hardcore, jungle, and drum and bass (HJDB) are three interconnected genres that emerged in the early 1990s, distinguished by their employment of fast-paced breakbeats, manipulation, and rhythmic transformation techniques (Hockman, 2014). These genres expanded upon the foundations established by traditional hip-hop, further enriching the diverse landscape of music production. In the present day, sampling is prevalent across many genres of music, and modern producers have adapted to incorporate this technique into their creative processes. It has become an integral part of contemporary music production, with a significant proportion of today's popular music being composed with samples (Morey and McIntyre, 2014).

Initially, SBEM producers sourced samples from vinyl records through a process known as *crate digging*. However, they have since adapted their selection methods to accommodate the digital age, obtaining large corpora of sampled recordings from existing digital recordings or via royalty-free resources to circumvent copyright issues. Digital crate digging has emerged as the contemporary alternative to sampling music from vinyl records. Producers may now explore online platforms such as Youtube and Tracklib[1] to find music containing suitable sampling material for their productions. Additionally, online stores provide access to thousands of samples and loops——segments of music material prepared for seamless repetition. With the expansive assortment of music and audio sample libraries available today, methods for selecting, organising, and manipulating samples are still limited by the boundaries of current technology. Despite their origins dating back decades, labour-intensive techniques for acquiring and modifying music samples persist as common practices among many producers, particularly those striving to uncover rare and inventive samples within the vast array of existing music.

Deep learning, a subfield of machine learning research, focuses on the development and understanding of algorithms capable of learning intricate relationships from data without the need for explicit programming (Goodfellow et al., 2016). As a result of advancements in deep learning, our interactions with digital information and media have been transformed and will continue to evolve across various domains, including computer vision (Chai et al., 2021), speech recognition (Li, 2022), natural language processing (Lauriola et al., 2022), and medical diagnosis (Islam et al., 2021). Concurrently, significant progress has been made in developing deep learning models capable of generating highly realistic and diverse outputs in the domains of text (Radford et al., 2019), images (Karras et al., 2021), and speech (Yang et al., 2021). For instance, text-based generative models like GPT-3 (Brown et al., 2020) and BERT (Devlin et al., 2019) have demonstrated impressive capabilities in natural language understanding and generation, enabling them to perform tasks such as language translation, summarisation, and dialogue generation.

Within the realm of music-related applications, deep learning techniques can be broadly categorised into two areas: music information retrieval (MIR) and music generation. MIR is a multidisciplinary field that

[1] `https://www.tracklib.com/`

focuses on developing methods for the analysis and characterisation of meaningful information from music data. This can include tasks such as genre classification (Ndou et al., 2021), recommendation systems (Martín-Gutiérrez et al., 2020), automatic music transcription (Benetos et al., 2018; Wu et al., 2018), and music source separation (Hennequin et al., 2020). Deep learning can also be employed to generate new music content by analysing and extracting features from existing recordings, creating learned representations of these musical elements. Applications of this technique include the development of powerful neural audio synthesisers (Engel et al., 2019; Kong et al., 2020) and models capable of generating complete musical compositions (Dhariwal et al., 2020; Shen et al., 2022). These advanced tools not only lower the creative barriers imposed by traditional music production workflows, which often come with steep learning curves, but also provide valuable knowledge, education, and new insights into the production process.

The primary aim of this thesis is to explore the potential of deep learning models in the context SBEM, with a focus on developing automated tools that can facilitate producers in realising their creative concepts. The process of locating appropriate samples for novel compositions has been a long-standing challenge in SBEM (Rodgers, 2003), whether it pertains to uncovering samples embedded within existing music (Chang, 2009) or navigating through extensive sample libraries (Andersen and Knees, 2016). Building upon genre-specific knowledge of the unique characteristics of SBEM, this thesis aims to enhance the creative process within this genre by devising systems that automate various SBEM tasks. These tasks include sourcing suitable samples from existing music and sample libraries, creating high-level overviews of SBEM arrangements, and generating highly controllable sounds that can be manipulated to smoothly transition between existing elements within sample collections. Through a series of experiments, the thesis demonstrates the development and implementation of novel deep learning tools tailored specifically for SBEM production. The research outcomes encompass the thesis itself, open-source implementations and datasets for sample analysis and generation. Collectively, these contributions are designed to foster further exploration and advancements at the intersection of deep learning and SBEM.

1.1 Motivation

Creating music is a complex and subjective endeavour, grounded in structure, skill, and intention. It involves a series of informed decisions that weave together various combinations of vibrations, which can be harmonious or inharmonious, depending on the desired outcome. These decisions are rooted in a high-level understanding of the musical algorithms required to achieve a specific outcome. The creative process of producing new music often involves borrowing and reinterpreting ideas from the past, drawing inspiration from established patterns and techniques while adding a personal touch (Burkholder, 1994). Frequently occurring combinations of musical algorithms, influenced by instruments, playing styles, music theory, and cultural traditions, give rise to distinct music genres and sub-genres. Musicians can learn and reinterpret these practices and workflows to express themselves while preserving specific characteristics that listeners would find familiar. This delicate interplay between innovation and familiarity allows artists to create unique compositions that resonate with audiences, building upon established musical elements while pushing the boundaries of creativity.

SBEM comprises genres that predominantly adopt the practice of sampling, a contemporary form of borrowing that resides at the nexus of technology and artistry. The global success of SBEM has been driven by innovative music producers who utilise production software and techniques to creatively repurpose audio. The growing popularity of SBEM genres has also spurred increased attention on their analysis within the fields of MIR, computational musicology, and audio processing (López-Serrano, 2019; Hockman, 2014). Hockman and Davies (2015) emphasised the importance of genre-specific approaches in MIR research, positing that understanding how a particular type of music is created can inform the development of new technologies tailored to the specific needs of producers. Developing genre-specific knowledge about a particular type of music can contribute to the creation of systems better suited to the needs of music makers.

Driven by my passion as an electronic music producer, this thesis aims to explore innovative techniques that empower producers to discover, reinterpret, and seamlessly integrate music samples, revitalising the sounds of the past with a fresh and dynamic perspective. The process of creating music can be significantly impeded by laborious tasks that disrupt a seamless creative flow, thereby affecting sustained periods of focused concentration. In electronic music production, sample retrieval remains a critical challenge within the creative process (Andersen and Knees, 2016). While commercial companies like Splice[2] and Loopcloud[3] provide extensive databases of annotated sample libraries, the sheer volume of available content can be overwhelming. Additionally, producers often come across unstructured samples in their personal collections, particularly when sourced from existing recordings, and must rely on their own retrieval strategies, sometimes resorting to randomness.

Sampling from existing music rather than sample libraries presents unique creative opportunities, fostering inspiration and a personalised sound while honouring specific eras, artists, or musical styles. However, the current absence of efficient tools specifically designed for locating samples in existing recordings compels producers to rely on their critical listening skills as they navigate through extensive amounts of music. This process requires producers to identify captivating elements within a piece, extract them, and seamlessly integrate these samples into their own compositions. The automatic generation of a sample library, derived from a vast collection of existing records and systematically labelled according to

[2] https://splice.com/

[3] https://sounds.loopcloud.com

its musical function or role, has the potential to offer an invaluable resource for producers and stimulate innovation in the realm of electronic music production.

An important aspect of composing music with existing recordings is that the samples preserve their inherent acoustic properties, resulting in limited options for modification and precise control, particularly in comparison to synthesisers and other electronic instruments. Additionally, the techniques for transforming samples have largely remained stagnant since the early 1990s, constrained by the intrinsic transformational potential tied to the timbral and rhythmic qualities of the source material. In contrast, synthesis techniques afford finer control over sound characteristics, yet struggle to accurately emulate the nuanced and complex nature of real-world instruments. One potential solution to address these limitations could be the development of a synthesiser designed to effectively mimic a given collection of samples, offering both the authenticity of the original sounds and the flexibility of synthesised instruments.

Addressing these challenges is a critical area of inquiry in music technology research, as designing effective interfaces for music production data exploration holds the potential to significantly improve the listening experience and creative process. Furthermore, the development of efficient, and engaging methods for navigating and organising the wealth of musical content is essential not only for music producers but also for the broader audio research community. In recent years, substantial advancements have been made in the fields of MIR and deep learning, offering solutions and methodologies for a wide range of music-related challenges. However, there remains untapped potential in developing tailored tools for more specialised tasks, catering to the needs of both professional and amateur music creators.

The digital landscape has significantly broadened the accessibility of music-related services and resources, offering numerous advantages while simultaneously increasing the need for efficient navigation and organisation of extensive music collections. Leading music streaming platforms, such as Spotify and YouTube, employ artificial intelligence-driven music recommendation systems to help users explore the vast content available on their platforms. In these systems, the importance of MIR and deep learning methods, such as for content analysis, continues to grow. Simultaneously, deep learning-based music production tools are gaining traction, providing innovative means for users to engage with and create music. Numerous deep learning production tools are now commercially available, such as iZotope RX[4] and Spleeter (Hennequin et al., 2020), which facilitate the separation of tracks into individual stems such as vocals, bass, and percussion; neural audio synthesisers such as Magenta NSynth (Engel et al., 2017) and Bytedance Mawf[5] provide users with intuitive control over timbre and dynamics, empowering them to explore novel sounds that would be challenging to produce with traditional synthesisers; and cost-effective automatic mixing and mastering services such as MixGenius Landr[6]. Moreover, AI-based music contests (Huang et al., 2020) are emerging as a trend, further highlighting the growing influence of deep learning and artificial intelligence technologies in the world of music production and fostering a competitive environment that encourages innovation and creativity.

1.2 Aim and Objectives

The aim of this thesis is to develop novel deep learning-based systems specifically tailored towards assisting in SBEM production. This thesis is centred around three key objectives:

[4] https://www.izotope.com/en/products/rx.html
[5] https://mawf.io/
[6] https://www.landr.com

- To design and optimise a deep learning system that automatically labels samples based on their functional roles within SBEM, and extends its capabilities to generate high-level summaries of SBEM, ultimately enabling the detection of samples in existing music.
- To design a deep generative model for synthesising a collection of drum samples, enabling the continuous exploration of intermediate samples.
- To investigate and propose systematic approaches for interacting with the latent space of the generative model, facilitating novel and creative manipulation of samples.

1.3 Contributions

The objectives outlined above are supported by the following contributions:

- Design and optimisation of modular deep learning models for analysing the structural role a sample occupies within a musical composition. (Chapter 4)
- Comprehensive evaluation of the deep learning models through three experiments: automatic instrumentation role classification, loop activation transcription, and detecting samples in existing recordings, providing insights into the performance and applications within SBEM. (Chapter 4)
- Development of a deep generative model for conditional synthesis of drum samples, guided by a compact latent space representation. (Chapter 5)
- Application of dimensionality reduction techniques to facilitate exploration of the latent space in deep generative models for drum synthesis. (Chapter 5)
- Development of an encoder model for controlling drum synthesis with input audio. (Chapter 5)
- Open-source TensorFlow[7] implementation of the models for drum sample synthesis, complete with pre-trained weights, training data, a command-line interface for training the models on new data, and prototype GUI for controlling the model.

1.4 Open Source Implementations and Datasets

- **Neural drum sample synthesiser**: Open source implementation for training and inference. `https://github.com/SoMA-group/style-drumsynth`
- **Prototype graphical user interfaces**: Interfaces for controlling drum sample synthesis. `https://jake-drysdale.github.io/tools.html`
- **Drum sample dataset**: Dataset of kick, snare, and cymbals. `https://github.com/SoMA-group/ADS`
- **Instrumentation role dataset**: Dataset of annotated SBEM compositions. `https://jake-drysdale.github.io/research.html`

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

1.5 Publications

The following papers have been published as part of this thesis:

- **Drysdale, J.**, Tomczak, M., Hockman, J. (2020), Adversarial Synthesis of Drum Sounds, in *Proceedings of the International Conference on Digital Audio Effects (DAFX), Vienna, Austria*.
- **Drysdale, J.**, Tomczak, M., Hockman, J. (2021), Style-based Drum Synthesis with GAN Inversion, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*.
- **Drysdale, J.**, Ramires, A., Serra, X. and Hockman, J. (2022), Improved Automatic Instrumentation Role Classification and Loop Activation Transcription, in *Proceedings of the International Conference on Digital Audio Effects (DAFX), Vienna, Austria*.

It is important to emphasise the collaborative nature of the research that contributed to the relevant chapters of my work. As the lead author, I played a central role in the publications on drum synthesis, elements of which have been adapted and expanded upon in Chapter 5. My colleague, António Ramires, and I shared authorship equally on the publication focusing on automatic instrumentation role classification. This work has also been adapted and extensively elaborated upon in Chapter 4. Our mutual research interests and collaborative efforts ensured a rigorous and comprehensive data analysis, which led to a deeper understanding of the research subject.

The following publications are associated with deep generative models for drum sample processing but do not contribute directly to this thesis:

- Cheshire, M., **Drysdale, J.**, Enderby, S., Tomczak, M. and Hockman, J. (2022), Deep Audio Effects for Snare Drum Recording Transformations, in *Journal of the Audio Engineering Society (JAES), Special Issue: New Trends in Audio Effects*.
- Tomczak, M., **Drysdale, J.** and Hockman, J. (2019), Drum Translation for Timbral and Rhythmic Transformation, in *Proceedings of the International Conference on Digital Audio Effects (DAFX) Birmingham, United Kingdom*.

1.6 Thesis Structure

The remainder of this thesis is divided into five chapters and structured as follows: Chapters 2 and 3 provide background information and literature reviews, and Chapters 4 and 5 provide the main research projects. Chapter 6 summarises the main findings from Chapters 1 through to 5 and presents possible directions for future work. The following provides a more detailed explanation of each chapter.

Chapter 2 provides a brief overview of the history, fundamental concepts, and technologies related to SBEM production, as described in the literature. This is essential to elucidate existing limitations and provide context for the research discussed in this thesis. It begins by tracing the origins and evolution of sampling, along with the development of new technologies. Common issues faced in SBEM are discussed, as well as the diverse practices employed by producers when composing music with samples. An overview of the characteristics of drums and the various techniques that SBEM producers employ to integrate them into their compositions is also provided. Finally, a review of MIR and deep learning research related to the analysis and generation of electronic music samples is given. This comprehensive

examination of key topics lays the foundation for understanding the contributions of the thesis and sets the stage for further exploration in subsequent chapters.

Chapter 3 provides an overview of deep neural networks for analysis and generation, introducing commonly used architectures for this task, along with their advantages and disadvantages. Core principles related to neural networks, training procedures, deep generative models, and strategies for manipulating the latent space are examined. These preliminaries are crucial for understanding the performance and capabilities of the implemented systems, offering the necessary context for the investigations presented throughout the thesis.

Chapter 4 introduces a novel system for automatically labelling samples based on their functional roles within SBEM. The system is further enhanced to provide high-level summaries of SBEM arrangements, and to detect samples in existing recordings. The proposed system is evaluated through three experiments: instrumentation role classification, loop activation transcription, and the identification of potential samples in existing recordings. In the first experiment, several deep learning architectures are evaluated, with a particular focus on their ability to accurately label samples according to various instrumentation roles, such as bass, melody, and drums. The second experiment delves into the analysis of structural characteristics of SBEM, using the proposed system to generate high-level summaries. The performance of the system is compared to previous approaches for loop activation transcription, a related task that involves estimating the locations in which loops occur throughout a piece of music. The system is further evaluated on a dataset of full-length SBEM compositions with annotated structural roles, demonstrating the applicability to authentic music scenarios and ability to accurately estimate the annotations. In the third experiment, a novel approach for the automatic retrieval of samples from existing recordings is introduced. The proposed deep learning architectures are evaluated on their ability to identify the location of breakbeats within funk, soul, and jazz recordings.

Chapter 5 introduces a novel deep learning system for synthesising drum samples. The system serves as a method for creating new, highly controllable samples from a collection, offering the ability to seamlessly morph between generations of these samples. The chapter commences with details of the system implementation and training procedures. The proposed system is then evaluated against a baseline system, featuring an analysis of attributes from the generated audio and an exploration of potential applications and system limitations. Systematic approaches for interacting with the latent space are proposed, involving the identification of synthesis parameters through various dimensionality reduction techniques. To further demonstrate the versatility and potential for customisation of the system, the chapter also presents supplementary experiments involving the use of different conditioning techniques. Subsequently, a prototype graphic user interface is showcased, illustrating how the system and identified synthesis parameters could be seamlessly integrated into an audio plugin.

The thesis is concluded in **Chapter 6** with a summary of findings across Chapters 2 to 5, a critique of the methods used in this work, and suggestions for further improvement and further work in this area.

Chapter 2

Review of Sample-based Electronic Music Production Literature

Sample-based electronic music (SBEM) describes a variety of music genres that are centred around the practice of sampling, which is to select, record, edit and process existing recordings to create new compositions or performances. SBEM is predominantly constructed using samples, either exclusively or in combination with synthesised audio components (Rodgers, 2003). As opposed to traditional music composition, where a composer may think in terms of melody, harmony and lyrics, sample-based music producers create musical collages primarily using the pre-recorded material that is available to them (Morey and McIntyre, 2014). The process of composing music with samples can involve complex music theory; however, music producers make use of distinct skill sets, workflow and tools that are specifically tailored towards the production of SBEM. Although sampling is often linked to hip-hop, in which music makers often use short segments of funk, soul and jazz for building music compositions, sampling has become a prevalent practice in a large majority of popular music.

The proliferation of sampling, driven by advancements in audio production and digital technologies, has substantially shaped the evolution of modern music, sparking a cultural transformation within the industry. Affordable audio production tools have empowered individuals of varying musical expertise to explore music creation through the use of samples (Rose, 1994). Sampling enables music creators to traverse the timeline of pre-existing music, connecting and combining musical ideas from across the globe in ways that could not have been achieved prior to the advent of sampling (Reynolds, 2011). Considering the abundance of music and audio sample libraries available at present, techniques for identifying, organising and processing samples are bound by the limitations of current technology. Many of the laborious methods for obtaining and manipulating music samples originated decades ago and are still standard practice for many producers (Rodgers, 2003; Andersen and Knees, 2016).

This chapter offers a comprehensive synopsis of the fundamental concepts and systems related to SBEM production as described in the literature. The aim of this synopsis is to elucidate existing limitations and provide context for the research discussed in this thesis. The contents of this chapter can summarised as follows. Section 2.1 presents a concise history of sampling, technology, and typology. Section 2.2 delves into the core practices involved in SBEM, specifically focusing on sample sourcing, selection, manipulation, and arrangement. Section 2.3 outlines the characteristics of drums and the various techniques employed to integrate them into an SBEM composition. Section 2.4 reviews music

information retrieval (MIR) and deep learning research related to the analysis and generation of samples. A chapter summary is provided in Section 2.5.

2.1 Music Sampling: An Overview

The re-purposing of audio material to create new music—known as *sampling*—has revolutionised the development of popular electronic music genres, and remains an essential technique in music to this day. Fundamentally, digital sampling is a process that involves the digital analysis and recording of an analogue signal for the purposes of audio playback and manipulation (Johnson, 1987). Sampling, in the context of electronic music production, is a practice that utilises digital audio technology to isolate, manipulate and rearrange segments of audio—or *samples*—from existing recordings to create new musical compositions. Sampling is a cultural practice that involves adapting existing works to create new ones. It is a modern form of adaptation rooted in a long-standing artistic aesthetic of borrowing, quoting, and repurposing (Sanjek, 1992).

2.1.1 Origins

The creative process of producing new music often involves borrowing and reinterpreting ideas from the past, drawing inspiration from established patterns and techniques while adding a personal touch (Burkholder, 1994). Throughout history, eminent composers such as Bach, Mozart, and Beethoven have drawn upon the works of their predecessors to craft new compositions (Arewa, 2005; Brandes, 2007). They borrowed melodies, harmonies, and rhythms to advance musical development, with listeners enjoying the transformations of familiar motifs within evolving musical styles. In the same vein, Nirvana's *Come as You Are* (1992) is a clear adaptation of the lead riff from Killing Joke's *Eighties* (1984). Analogously, sampling entails extracting a component from an existing recording and ingeniously recontextualising it to create a new composition.

The idea of creating music with samples originated around the 1940s, when composers such as Pierre Schaeffer and John Cage began to expand on the harmonic, rhythmic and timbral restrictions imposed by the western orchestra by experimenting with magnetic tape to repurpose and modify recorded audio from sources such as radio broadcasts, field recordings and various mechanical noises (Holmes, 2012). Schaeffer laid the groundwork for SBEM production through the development of musique concrète, an early form of avant-garde electronic music that is characterised by the use of tapes, sound design, and looping to create new collages of music from existing sounds (Sinnreich, 2010). In contrast to other traditional music, musique concrète was not defined by the particular sounds used in a composition, but by a particular approach to listening and composing, in which any sound can be transformed into musical material through the application of the right techniques (Emmerson, 2018). During this era, composers discovered that, by physically cutting and joining magnetic tape through editing and splicing techniques, they could sequence pre-recorded sounds in space and time to create music without the need for human performers. By splicing sections of tape together, they could make samples recorded on magnetic tape repeatable in an end-to-end loop. Additionally, the attack and decay of recorded sounds could be manipulated by cutting tape with sloped angles. Holmes (2012) describes a three-step process for composing music with tape. First, the recording of raw material on to magnetic tape. Second, listening to the tapes, extracting sections to be used in the composition and storing them for ease of access.

Third, manipulating and arranging the chosen segments of tape into a desired sequence. Although the practice of composing with magnetic tape is almost obsolete today, there are many parallels with modern SBEM production. The continued use of fundamental concepts and terminology (e.g., splice, loop, record) in SBEM highlights the enduring relevance of these techniques and their contribution to the evolution of electronic music production.

During the mid-20th century, early forms of sampling technology were introduced such as the Mellotron, a keyboard instrument capable of triggering segments of audio recorded to tape with each key. The term sampling was coined by the manufacturers of the Fairlight Computer Music Instrument (CMI), the first commercially available digital synthesiser to adopt a sampling function (Davies, 1996). However, due to the expensive cost of early technology, sampling was exclusively available to only the wealthiest of producers and musicians. The advent of digital sampling technology in the mid-1980s, specifically affordable samplers like the Akai Music Production Center (MPC), revolutionised the landscape of music production. For the first time, professional-quality recordings could be incorporated into home studio productions. Later in the decade, the introduction of the Akai s1000 further democratised sampling technology, making it accessible to a wider range of musicians.

The practice of sampling achieved mainstream appreciation and became ubiquitous in popular music such as hip-hop, rock, and electronic (Chang, 2009). It allowed producers to create music from their homes without the need for expensive studio time or access to musicians. Pre-recorded audio could be loaded into a sampler and played back using a musical instrument digital interface (MIDI) keyboard, trigger pad, or sequencer. Instead of practising an instrument or performing with a band, sample-based producers would search for sounds from recorded music and arrange them into new compositions. Sampling opened up new creative possibilities for producers by allowing them to integrate diverse elements from various genres, time periods, and cultural traditions into their compositions. Although early sampling technologies were limited in terms of computer memory and audio processing potential, these constraints stimulated creativity, artistic choices, and new genres of music (Hockman, 2014).

One highly influential figure in hip-hop who had a significant impact on the sound of the genre in the 1990s and 2000s is James Dewitt Yancey, better known as J Dilla (Charnas, 2022). J Dilla was widely recognised for his innovative use of sampling and his signature soulful, melodic production style. He was a pioneer in the use of the Akai MPC, a digital sampling drum machine and sequencer that was popular in the 1990s and is still used today by many electronic music producers. The Akai MPC, a powerful and versatile tool for electronic music creation, has been employed in numerous classic hip-hop tracks and has been favoured by renowned producers such as Dr. Dre, Kanye West and Pete Rock (Magazine, 2004; Producer's Edge, 2008). Figure 2.1 displays the Akai MPC3000 (Linn, 1994), which features a built-in sampler, drum pads, and a variety of control knobs and buttons for manipulating samples and creating music. J Dilla used the MPC3000 extensively in his production work and was known for his innovative approach to finding unique samples and using them in unexpected ways to create complex and dynamic music (Charnas, 2022). One of his notable techniques was to program samples using slight imperfections to emulate a more humanised rhythm. His creative use of the Akai MPC played a significant role in shaping the sound of modern hip-hop and his influence can still be heard in the music of today (Charnas, 2022).

Figure 2.1: Akai MPC3000 digital sampling drum machine and sequencer.

2.1.2 Breakbeats

Since the birth of sample-based music to the present day, percussion solos—or *breakbeats*—are the most sought-after samples for many music producers. A breakbeat is a percussion-only passage that can appear anywhere in any music, although typically it occurs towards the end of a funk song (Katz, 2012). SBEM producers use breakbeats as an essential building block to form a rhythmic foundation when composing music. The usage of breakbeats in music was initiated around the birth of hip-hop and is often credited to Clive Campbell (better known as Kool Herc) who discovered that partygoers favoured breakbeat sections from funk records during the 1970s (Katz, 2012). By taking a drum solo out of its original context, Kool Herc pioneered a new style of music by repeating the breakbeat section of a record using a turntable. As this early style of hip-hop music began to grow in popularity, other disc jockeys (DJ) such as Grandmaster Flash went on to develop more sophisticated ways of looping breakbeats through the use of two turntables, a mixing interface and two copies of the same record (Katz, 2012). Through developments in digital sampling technology, it became possible to record and store breakbeats from vinyl records to digital memory. This facilitated the potential for more sophisticated manipulation of breakbeats such as layering and resequencing.

Primarily sourced from samples of funk and jazz recordings from the 1960s to 1980s, breakbeat usage provided the foundation for many musical genres. Several influential genres that developed from breakbeat-focused music and the use of sampling technology are hardcore, jungle, and drum and bass (HJDB). HJDB are three related genres that are characterised by fast tempi, breakbeats and distorted bass sounds. Hockman (2014) provides a comprehensive examination of the history, production methods, and computational analysis of these genres. The selection and creative usage of breakbeats are often-used attributes in the determination of an artist's ability within HJDB and its many sub-genres (Hockman and Davies, 2015). Breakbeat selection is the process of choosing one or more breakbeats from the vast amount of existing pre-recorded music to use within a new production. Music producers often dedicate a considerable amount of time searching through and listening to vinyl records or digital music to find unique and interesting breakbeats (Joyce, 2022). Once identified, breakbeats can be recorded and segmented using a digital sampler, then rearranged or looped using a sequencer or tracking software (Hockman, 2014). Other prevalent transformations include layering, distortion, pitch-shifting and time-stretching, which can be achieved using audio editing software and analogue effect units.

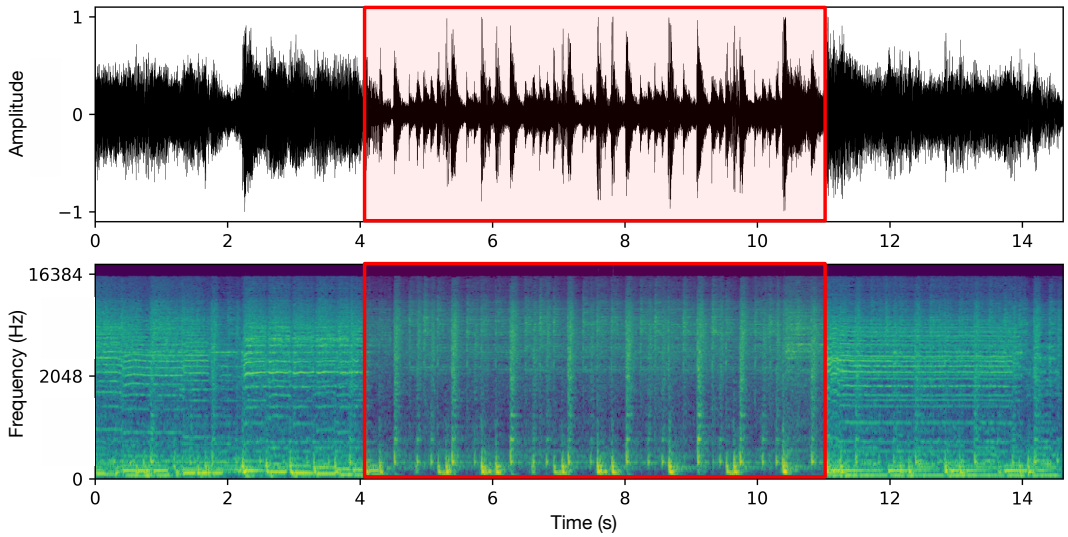

Figure 2.2: An excerpt from *Amen, Brother* (1969) by The Winstons. Above is an audio waveform and below a Mel-scaled log spectrogram. The area highlighted in red displays the location of the Amen break sample.

One of the most iconic breakbeats was originally sampled from the drum solo in The Winstons' *Amen, Brother* (1969) (The Economist, 2011). The drum solo in *Amen, Brother* (known as the *Amen break*) was performed by the drummer Gregory Coleman and lasted just under seven seconds. The Amen break became extremely popular in the late 1980s and early 1990s due to its widespread use in hip-hop and HJDB (Hockman, 2014). The sample from *Amen, Brother* has been described as one of the most widely used and influential breakbeats in popular music (Reynolds, 2012). It holds the distinction of being the most sampled record on the WhoSampled website (WhoSampled, 2023), a platform dedicated to cataloguing and documenting sampled music, which relies on user contributions to enrich its database. The Amen break has a strong cultural significance within the hip-hop and electronic dance music scenes, which has contributed to its enduring popularity (The Economist, 2011).

Figure 2.2 illustrates an excerpt of *Amen, Brother* in the form of an audio waveform and spectrogram, displaying the start and end locations to splice and isolate the breakbeat from the surrounding instrumentation. At a glance, it can be seen from the highlighted area of the waveform that the percussion is isolated; at this section the waveform is more sparse and the transient content produced by the percussion is more visible. Additionally, close inspection of the spectrogram reveals that, during the breakbeat section, there is an absence of harmonic frequencies (bright parallel lines) arising from other musical elements (e.g., horns and piano). These qualities make the *Amen break* a desirable sample for DJs and producers due to its ease of manipulation and looping. Furthermore, the breakbeat has a powerful and energetic sound that is well-suited for use in fast-paced, high-energy music (Hockman, 2014).

Due to the significant role that breakbeats play in popular music, a growing amount of academic research has been dedicated towards computational strategies for breakbeats analysis and manipulation. Nick Collins pioneered various research towards automating production techniques associated with the manipulation of breakbeats including segmentation (Collins, 2001b), resequencing (Collins, 2002) and algorithmic composition (Collins, 2001a). Hockman et al. (2012) present a genre-specific method for downbeat detection specifically designed for HJDB genres, which are characterised by fast tempo,

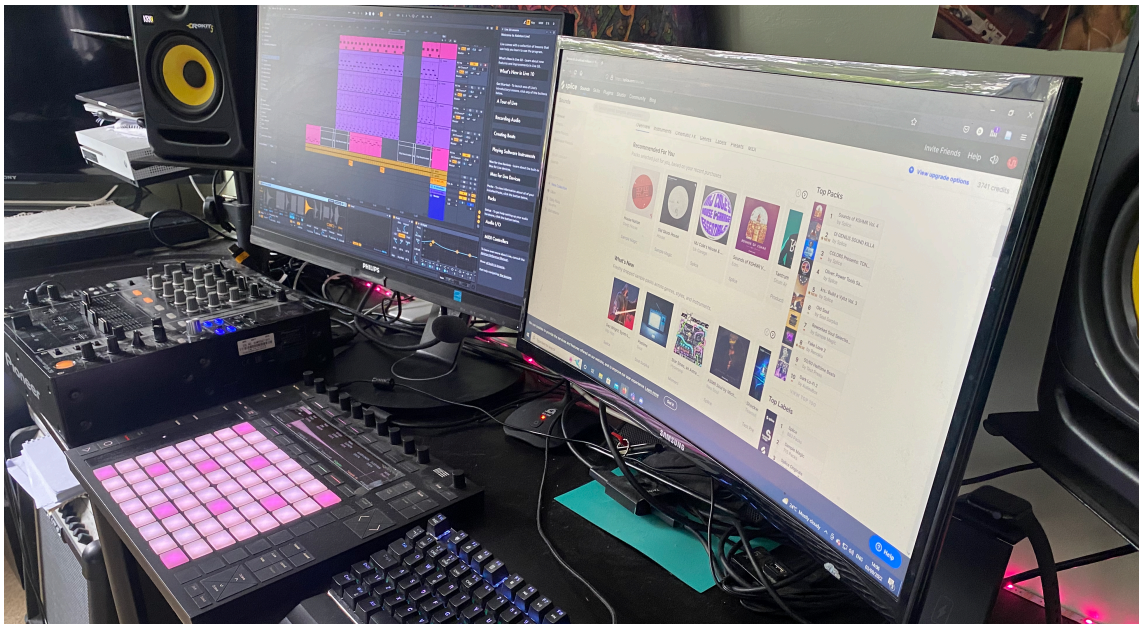

Figure 2.3: An example of a typical home music production setup.

breakbeat usage, high note density, and emphasis on offbeats. Hockman (2014) provides an ethnographic and technological study of breakbeats in HJDB genres, covering the history, track creation process, analysis techniques, and several interviews with established HJDB musicians and record label owners. Hockman and Davies (2015) propose the task of breakbeat classification, developing a system for identifying which breakbeats are sampled in a given composition as well as a specialised system for automatic segmentation and classification of the individual drum sounds. López-Serrano et al. (2017) established a first baseline for automatically identifying the location of breakbeats in digital music recordings by adapting an approach previously used for singing voice detection. Frane (2017) examined the microtiming of classic breakbeats focusing primarily on sixteenth-note swing—a systematic delay of even-numbered sixteenth-note divisions of the pulse. Furthermore, Dittmar and Müller (2016) proposed a method for reverse engineering breakbeats using score-informed source separation to extract and isolate individual drum hits from the drum sound mixture.

2.1.3 Modern Sampling Technology

In the present day, affordable music production technologies for incorporating and manipulating samples have democratised the SBEM creation process, allowing users with varying levels of musical knowledge to experiment in the creation of SBEM from the comfort of their homes (Reuter, 2022). Sampling has become ever more accessible to potential music makers; memory constraints are no longer an issue, there is an abundance of free and affordable music and sample libraries, and many producers have access to powerful digital audio workstations (DAW) such as Ableton Live[1] and Logic Pro.[2] Figure 2.3 shows an example of a home music production set-up consisting of a pair of monitors, a hardware controller, and a computer with a DAW. DAWs revolutionised music recording and production by transitioning from analogue and digital tapes to computer-based storage systems, such as hard disk drives (HDD) and solid-state drives (SSD), for managing and storing digital audio data (Marrington et al., 2017).

[1] `https://www.ableton.com`
[2] `https://www.apple.com/uk/logic-pro`

Larger storage capacities and faster access times offered by HDDs and SSDs enable more efficient music production workflows and greater creative freedom. These storage systems support complex projects, enhance performance, streamline collaboration, and help producers adapt to evolving audio technology.

Many SBEM producers create music with a DAW, using its grid-like interface to arrange, manipulate and mix samples. A DAW is a graphical environment on a computer screen that facilitates the manipulation of MIDI data and digital audio. The interface design influences creative decision-making and workflow, with various features such as the main sequencer interface, mixer, piano roll for MIDI editing, waveform display, and traditional score (Marrington et al., 2017). DAWs typically feature multiple channels, drawing inspiration from tape-based systems, with each channel representing an individual audio track. Each channel contains a stream of audio that can be edited, processed, and enhanced using a chain of audio effects, such as equalisation, compression, and reverb. These effects are used to sculpt and shape each sound, adding depth, character, and interest to the music. The sum of the individual channels, when combined, results in an overall mixture of audio that forms the final piece of music (Bartlett and Bartlett, 2009). DAWs often include built-in sampling capabilities, and many also offer a range of virtual instruments and effects that can be used to create and manipulate sounds.

Modern software samplers, such as Native Instruments Kontakt,[3] Steinberg HALion,[4] offer advanced sampling capabilities and features for producers. These samplers can be used to create digital instruments within a DAW using pre-recorded audio. They include a number of built-in effects, such as filters, pitch shifting, and amplitude envelope control, which allow producers to shape and transform the sound of the recordings within a plug-in. Software samplers often come with professionally designed multi-sampled libraries, comprising a vast collection of audio recordings of instruments played in various styles and captured using diverse microphone techniques. However, creating such extensive libraries demands considerable time, effort, and financial resources, often requiring expensive studios and professional musicians for high-quality results. Despite their versatility, these samplers are inherently limited in terms of control, as users can only trigger and manipulate the recorded samples with standard audio processing options. The techniques for transforming samples have largely remained stagnant since the early 1990s, constrained by the intrinsic transformational potential tied to the timbral and rhythmic qualities of the source material.

To facilitate interactive music creation and performance, various hardware digital sampling devices have been developed to work alongside DAWs. For example, as seen in Figure 2.3, the Ableton Push is a hardware device with a grid of velocity-sensitive pads, physical dials and buttons for triggering sounds, sequencing, and controlling parameters in Ableton Live.[5] Modern sampling hardware typically integrates a software platform with an array of hardware controls, providing users with a more intuitive approach to navigate and edit their music than the traditional computer, mouse, and keyboard setup. Hardware sampling devices are popular among electronic musicians as they facilitate music creation and performance, offering a hands-on, tactile approach using both the software and hardware components of the system (Kladder, 2016).

[3] https://www.native-instruments.com/en/products/komplete/samplers/kontakt-7
[4] https://www.steinberg.net/vst-instruments/halion
[5] https://www.ableton.com/en/push

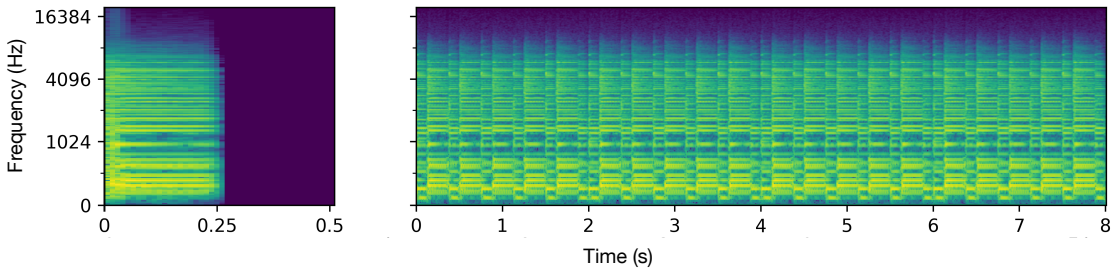

Figure 2.4: Mel-scaled log spectrograms for a piano one-shot (left) and a piano one-loop (right). The sample on the left is single piano chord played for 0.25 seconds. The sample on the right is a one-bar chord sequence repeated four times to create an eight second, four-bar piano loop.

2.1.4 Sample Typology

A typology for different types of music samples can serve as an invaluable tool for producers, musicians, and musicologists by offering a systematic framework for understanding, organising, and appreciating the diverse range of sampling techniques and their applications across various music genres. Categorising samples based on their musical elements, purposes, or characteristics, not only facilitates the effective management of extensive sample libraries but also enables producers to make informed decisions in selecting samples for specific creative goals. Frequently sampled musical elements include: drums, such as breakbeats and single percussive strikes; bass lines, for example, bass guitars or double bass; vocal or spoken word excerpts; and melodic elements, for instance, lead guitars and piano melodies (Hockman, 2014; Sewell, 2013). Other sounds that are sampled include field recordings, radio and television extracts, and industrial or mechanical sounds (Holmes, 2012; Landy, 2012). Moreover, the length of a sample can vary depending on its intended use, ranging from a single cycle of a waveform for the development of a synthetic instrument (Bristow-Johnson, 1996) to an entire phrase, such as the chorus of another piece of music (Sewell, 2013).

Several researchers have devoted their efforts to creating a typology of sampled material to enable the comparison and analysis of sampling trends. Through a combination of analysis and ethnography, Sewell (2013) developed a typology of sampling in hip-hop music to identify the sounds used in sample-based music, differentiate the treatments of sampled sounds and describe the sampling styles of music producers. Samples are divided into three main types: structural samples, surface samples, and lyric samples; structural samples create the rhythmic foundation, surface samples overlay the foundation, and lyric samples provide words or phrases. Ratcliffe (2014) proposed an alternative sampling for electronic dance music using a classification system to group sounds into one of four main categories: short isolated fragments (i.e., one-shots), loops and phrases, larger elements, and transformed material. In modern collections, samples are often categorised into *one-shots* and *loops*. Figure 2.4 displays spectrograms for a one-shot (left) and a loop (right).

One-shot: A one-shot is a type of audio sample that is associated with a single event sound, such as a strike of a drum, a piano note, or a guitar chord. One-shots can be sequenced using MIDI information to precisely adjust their pitch and timing, allowing for the creation of dynamic music phrases. One of the most prevalent usages of one-shots in SBEM is for creating drum patterns, typically using kick drums, snare drums and cymbals. One-shots can be triggered using the performance pads built into a sampler, or drum machines and virtual drumming software to simulate an acoustic drum kit, enabling

live performance and improvisation. Drum one-shots can be created by recording percussion instruments or using electronic instruments to generate percussive sounds through synthesis techniques (Section 2.3.1). SBEM producers may also segment longer samples into individual one-shots, providing more flexible control when creating original sequences and variations.

Loop: The term *loop* is often used to describe a type of sample, usually of short duration (between one to four bars in length), that has been prepared for seamless repetition (Stillar, 2005). A loop can be created by sampling a section from a pre-recorded piece of music, extracting a short phrase from a solo instrument recording, or through the use of MIDI and a selection of one-shots. To facilitate seamless repetition, the loop start and end points must be carefully selected to ensure a musical idea is resolved (Collins, 2014). These loops often serve as the primary material from which music makers can create compositions through various editing and combinatory processes (e.g., layering, splicing, rearranging). Additionally, loops are an essential structural component of SBEM, and are often associated with a particular role. Alongside the growing interest in creating music with loops, academic literature on assisting loop-based music production has increased. There are several methods for automated loop extraction from pre-existing music (Shi and Mysore, 2018; Smith and Goto, 2018; Smith et al., 2019), and loop creation (Cocharro et al., 2014; Alain et al., 2020; Chandna et al., 2021).

Many commercial companies, such as Splice[6] and Loopcloud,[7] have curated large databases of one-shots and loops, which can help both professional and amateur producers to create music without expert knowledge of music theory (e.g., chords, scales, notation). In contemporary music production, loops are often categorised based on the role they occupy in a piece of music (i.e., instrumentation roles) such as bass, melody, or effects, rather than specific instruments. This approach to categorisation is evident in established sample libraries, including Splice and Loopcloud. Given that SBEM production is significantly influenced by aesthetic preferences (Schloss, 2014), producers frequently select samples based on their functional roles within the musical arrangement. By categorising samples based on their instrumentation roles, it is easier to understand how they contribute to the composition and how they will interact with other elements within the piece. Moreover, producers can explore a broader range of traditional and synthesised timbres without being constrained by the instrument specificity, which may be difficult to determine when dealing with heavily processed or experimental samples. This focus on functional roles encourages creativity and experimentation, allowing producers to explore new combinations of sounds and instruments. For instance, when a producer seeks a sample to fulfil the bass role in their music, they can search for loops labelled accordingly and be provided with candidates from various instruments, such as bass guitars, pianos, or synthesisers.

2.2 Sample-based Electronic Music Production

SBEM producers typically engage in the process of locating and selecting a variety of samples, which are then skillfully combined and arranged to create a cohesive musical composition (Rodgers, 2003; Morey and McIntyre, 2014; López-Serrano, 2019). Samples can be sourced from a variety of different resources such as pre-recorded music and online sample stores (López-Serrano et al., 2017; Shelvock, 2020). Selecting samples that sound coherent together is a highly subjective endeavour, and often relies on the aesthetic judgement of a music producer to assess their suitability within the current musical

[6] `https://splice.com`
[7] `https://sounds.loopcloud.com`

Figure 2.5: Searching for records to sample (i.e., crate digging) in SJ Records, an independent record store located in Stratford-upon-Avon, UK.

context (Morey and McIntyre, 2014). After obtaining a collection of samples, producers employ various techniques to modify their characteristics, including pitch shifting, time stretching, and layering (Jackson, 2016). These samples are then typically arranged into a composition using a DAW. Another key task is the creation of a drum track, as it provides an underlying structure and rhythmic foundation for the composition. This section provides an overview of the techniques and tools that are typically used in the process of creating music using samples.

2.2.1 Sample Sourcing

Originally, SBEM producers sourced samples by visiting record stores, flea markets and thrift shops to seek vinyl records that may contain sampling material (Vályi, 2010). Upon acquiring a record, producers engage in a meticulous listening process, attentively listening to the entire content in search of an appealing section to sample (López-Serrano et al., 2017). This process is referred to as *crate digging* and involves physically searching through crates of vinyl records to find musical material that can be reused in a new composition (as seen in Figure 2.5). It has granted producers access to an array of global musical material that would have otherwise remained unreachable, fostering the ability to traverse and connect decades of sound and music (Reynolds, 2011). Crate digging can be a time-consuming and often frustrating process as it requires patience and persistence. However, for some individuals, the excitement of the search and the feeling of accomplishment that comes with discovering a rare record or track makes the effort worthwhile (Ahmed et al., 2012; Jackson, 2016).

> There's something on every record. You can make any record into a beat. I mean, even in the early days, cats made music out of anything. I collect music. Music that sounds good and I make samples with real music.
>
> - Pete Rock (Producer's Edge, 2008)

There are different strategies that crate diggers use when searching for records or tracks. For example, some producers may focus on a particular genre or era, while others look for records or tracks by specific

artists or labels (Vályi, 2010; Ahmed et al., 2012). Once an interesting record has been found, it can be recorded into a digital audio system and a sample can be extracted using a sampler or DAW. As an alternative to crate digging, companies such as Zero-G[8] produced sample packs consisting of a variety of royalty-free samples, including breakbeats, bass lines, melodies and sound effects (Snoman, 2012; Hockman, 2014). These samples can be used as a source of inspiration or to quickly find and incorporate new sounds into their music.

With the proliferation of music on the internet, music producers have adapted to take advantage of the abundance of digital music and samples available online. Digital crate digging is the modern alternative to sampling music from vinyl records and sample CDs; instead, producers search through websites such as YouTube and Beatport[9] in search of existing music to sample from. Producers can also obtain samples from cloud-based online services such as Splice and Loopcloud, or by using community-based sites like Looperman and Freesound, which offer thousands of free loops and samples (Shelvock, 2020). Online sample stores are curated collections of ready-made audio samples that enable producers to focus on other creative aspects of music production instead of the tedious task of sourcing samples from pre-recorded music and the technicality of sound design and recording. These stores offer a wide range of loops and one-shots from a variety of genres, such as hip-hop, house, and jazz. They also offer sample packs created by popular artists and producers, which can inspire creativity and accelerate the music production process. It is important to note that while some of these sources may offer high-quality, royalty-free samples created by professionals, others may be accessed illegally, which can result in legal and ethical issues. Tracklib is a recent music service that allows producers to browse a large catalogue of original music tracks and sample them legally for use in their own music productions.[10] It solves many legal and ethical issues surrounding sampling and music clearance by providing a streamlined process for obtaining the necessary licenses and permissions for the samples used in a composition.

2.2.2 Sample Selection

Sample selection is a meticulous retrieval task in which music producers use their aesthetic judgement to listen and identify samples from the collections available to them. This task typically involves searching for samples, from a potentially large library, to use in new musical compositions. This process requires practice, experimentation, and trial and error to identify samples with particular attributes, such as sparseness, instrumentation, and desirable timbral characteristics (Andersen and Knees, 2016; Hockman, 2014). Efficient organisation, indexing, and retrieval of audio libraries play a pivotal role in the selection processes of music composers and producers (Andersen and Knees, 2016; Jackson, 2016). In the context of online sample stores, the navigation process typically involves a semantic approach, whereby users have a particular structure of meaning in mind when searching for samples and associated metadata or tags. These music tags are descriptive keywords that provide information about the audio content from the perspective of the listener, such as instrument (e.g, guitar, piano, synthesiser), role (e.g., melody, bass, drums), genre (e.g., hip-hop, jungle, techno) and type (i.e., loop or one-shot). Sample libraries produced for commercial use are often structured and include descriptive tags that enable efficient retrieval of desired sounds. Splice is a widely recognised and utilised online sample store that offers music producers access to a broad selection of high-quality sounds and loops, drawn from an

[8] https://zero-g.co.uk
[9] https://www.beatport.com/
[10] https://www.tracklib.com

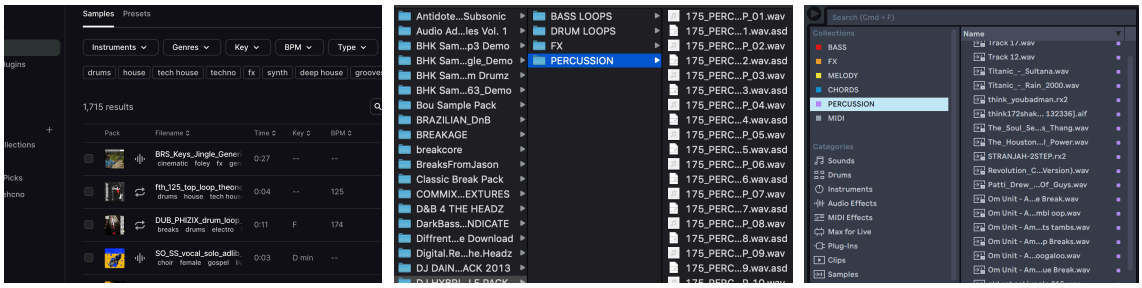

Figure 2.6: Examples of sampling browsing interfaces. Splice sample browser (left), hierarchically structured folders of samples (centre), Ableton Live's built-in browser (right).

extensive library featuring works from prominent artists and producers. As depicted in the left of Figure 2.6, the Splice user interface incorporates a tag-based browsing and filtering system, enabling users to explore and discover new sounds for use in their musical projects. However, not all sample stores provide this functionality, and producers may also encounter unstructured samples in their personal collections, especially if they have sourced them from pre-recorded music. Producers who frequently sample from various existing recordings and neglect to label or organise them appropriately may find themselves with a collection of unstructured samples. Without effective categorisation or metadata, locating specific sounds for use in future projects can become laborious. In such cases, producers often develop their own systems for organising and searching their samples (Andersen and Knees, 2016). In a series of interviews with accomplished music producers, Andersen and Knees (2016) identify the challenges these professionals encounter when navigating their personal sound collections.

> Sometimes, when you don't know what you are looking for, and you're just going randomly through your samples, that might be helpful, but most of the time I have something in mind that I am looking for, and I am just going through all these sound files, and I am just waiting for the sound which I had in mind to suddenly appear. Or what comes the closest to what I had in mind. So I think that most of the time, I know what I am looking for, and then it is just a matter of time before I find it. - TOK002 (Andersen and Knees, 2016)

The centre of Figure 2.6 shows an example of a hierarchical folder structure often used by producers to organise their sample packs. Manual annotation of samples for organisational purposes can be a challenging and time-consuming task, especially when dealing with large audio collections. DAWs often feature browsing interfaces that facilitate text-based retrieval of audio files by scanning the computer directory for matching filenames and allowing users to preview samples within the DAW. For example, the Ableton Live browsing system shown on the right side of Figure 2.6, enables users to locate samples using a text-based search, provided that the sample filenames are labelled with appropriate text identifiers. Moreover, the colour-coding system in Ableton Live enables users to tag and organise their own samples on the fly, simplifying the process of locating them in future sessions. Owing to the advantages conferred by music tags and metadata, numerous studies have focused on developing scalable systems capable of automatically generating tags for music (see Section 2.4.1).

Music tags, while useful in filtering and selecting samples through text-based queries, may be limited in their ability to encapsulate the intricate and subtle qualities of sound. This can make it difficult for users to align the predefined semantic structure with their own individual perception and understanding of the sounds (Andersen and Knees, 2016). Furthermore, utilising broadly semantic tags for searching can still lead to unwieldy selections, particularly when browsing large collections of samples. The challenges

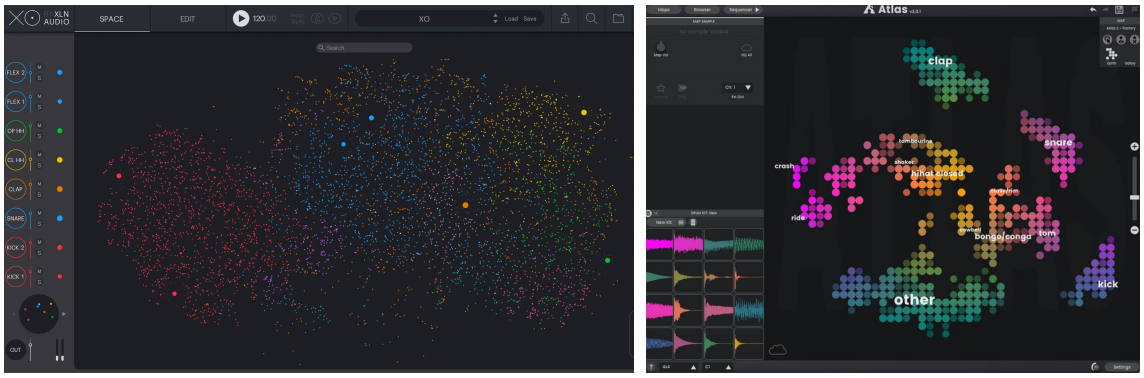

Figure 2.7: Examples of commercially available products for automatic organisation of drum samples, XLN XO (left) and Algonaut Atlas (right). Drum samples are mapped onto a two dimensional space and colour coded based on their type (e.g., kick, snare, hi-hat, and clap).

associated with sample selection have prompted research into more efficient methods of organising and navigating collections of samples (see Section 2.4.5). A number of commercial products and companies offer solutions for organising sample collections including, XLN XO[11] and Algonaught Atlas.[12] As shown in Figure 2.7, these tools provide a framework in which samples with similar attributes are located in proximity to one another on a two-dimensional interface, thus facilitating the search and selection process.

2.2.3 Sample Manipulation

Sample manipulation refers here to the various techniques that can be used to transform a sample, such as by changing its pitch, tempo, timbre, or other characteristics. Pitch transformation, for example, can be used to ensure that samples are harmonically compatible when combined in a composition. Adjusting the tempo can also help producers to achieve a desired rhythmic feel or fit samples into a specific tempo. The ability to manipulate samples opens up a vast range of creative possibilities for producers and helps them to push the boundaries of what is possible with sound. Sample *flipping* is a term often used within the SBEM community to describe this process of transforming previously existing material in new and innovative ways (Kruse, 2016). Some examples of transforming a sample include layering multiple samples to create a more complex and textured sound; applying audio effects such as equalisation, reverb, or delay to modify the timbre and dynamics; and using time-stretching or pitch-shifting to change the duration, tempo or pitch of a sample to correspond with other elements in the composition.

HJDB music brought about several innovations in the studio, and one of them was the introduction of a new sonic texture achieved by pushing the time-stretch algorithm of Akai S series sampler to produce audible artifacts (Hockman, 2014). The time-stretched effect has a very distinct and instantly recognisable sound, with a stuttering, metallic quality. The production of Dead Dred's *Dred Bass* (1994) is a widely recognised example of the advanced sample manipulation techniques that emerged from HJDB. The track showcases the use of several intricate methods, including time-stretched vocal samples, a skillfully manipulated *Amen* breakbeat with pitched individual drum hits, and a reversed and distorted synthesised bass sound (Hockman, 2014). These techniques were subsequently adopted and replicated

[11] https://www.xlnaudio.com/products/xo
[12] https://algonaut.audio/

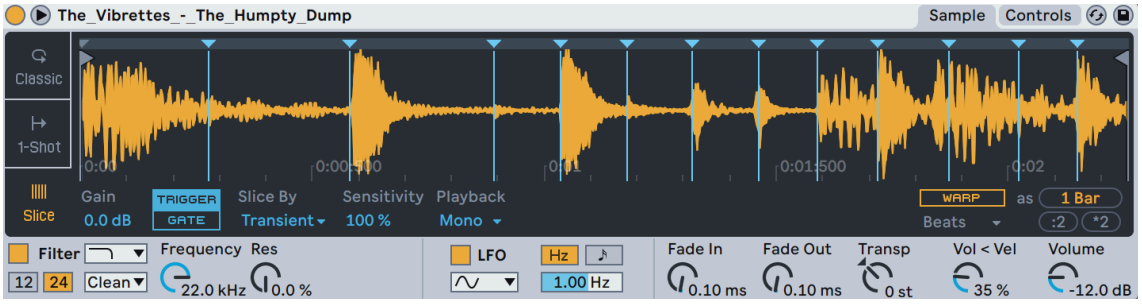

Figure 2.8: Segmenting a breakbeat sample using Ableton Simpler. Samples are segmented automatically using the transient information (vertical lines in waveform display). The sensitivity parameter can be used to adjust the number of segment locations and additional segments can be added manually. Once segments are acceptable to the producer, rhythmic modifications can be made by rearranging the segments using MIDI information.

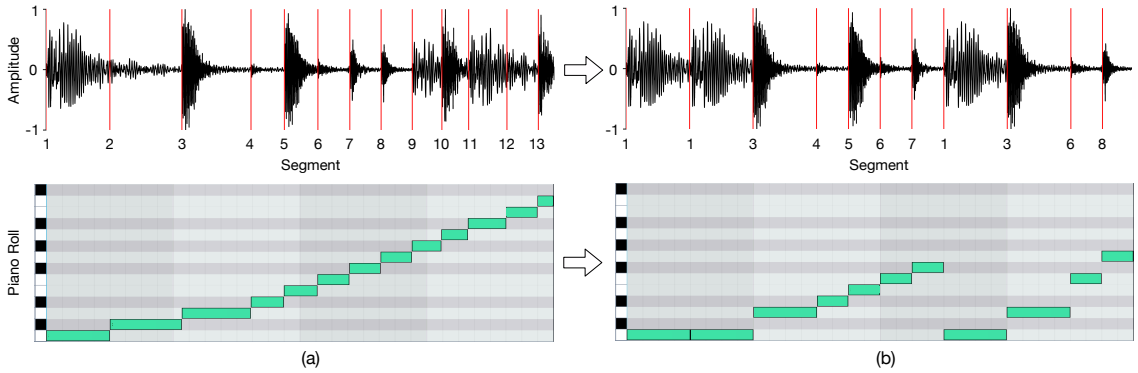

Figure 2.9: Rearranging a segmented breakbeat sample to create a rhythmic variation. The top left is the waveform of the original sample and corresponding piano roll MIDI data. The right side shows a transformed version, achieved by reordering the segments.

in numerous later productions within the genre. Another frequently used technique is to divide a sample into several segments (often termed *chopping*) and rearrange them into a new pattern (Morey and McIntyre, 2014). This technique is especially prevalent in HJDB, in which rhythmic modifications are a key aspect of the genre (Hockman, 2014; Joyce, 2022).

Figure 2.8 shows a breakbeat imported into Ableton Simpler, a sampling instrument that allows a user to playback and manipulate samples in a variety of ways. Similar to many other modern samplers, Simpler incorporates a semi-automated segmentation feature that detects segment boundaries by analysing the transient envelope of the waveform. A sensitivity parameter can be used to adjust the number segment locations and additional segments can be added by manually clicking on the waveform. Once a user has selected the desired segment boundaries using a software sampler, rhythmic modifications can be made by rearranging the segments using MIDI data. Figure 2.9 demonstrates the rearrangement of a breakbeat sampled from *Humpty Dump* (1973) by The Vibrettes. The top of Figure 2.9 (a) shows the original waveform and the segment boundaries identified using an automatic segmentation functionality. Below shows the Ableton piano roll, a graphical representation of MIDI data associated with the segment boundaries. Figure 2.9 (b) shows a rhythmic transformation created by reordering the MIDI data to create an alternative drum pattern. It is worth noting that the example shown uses

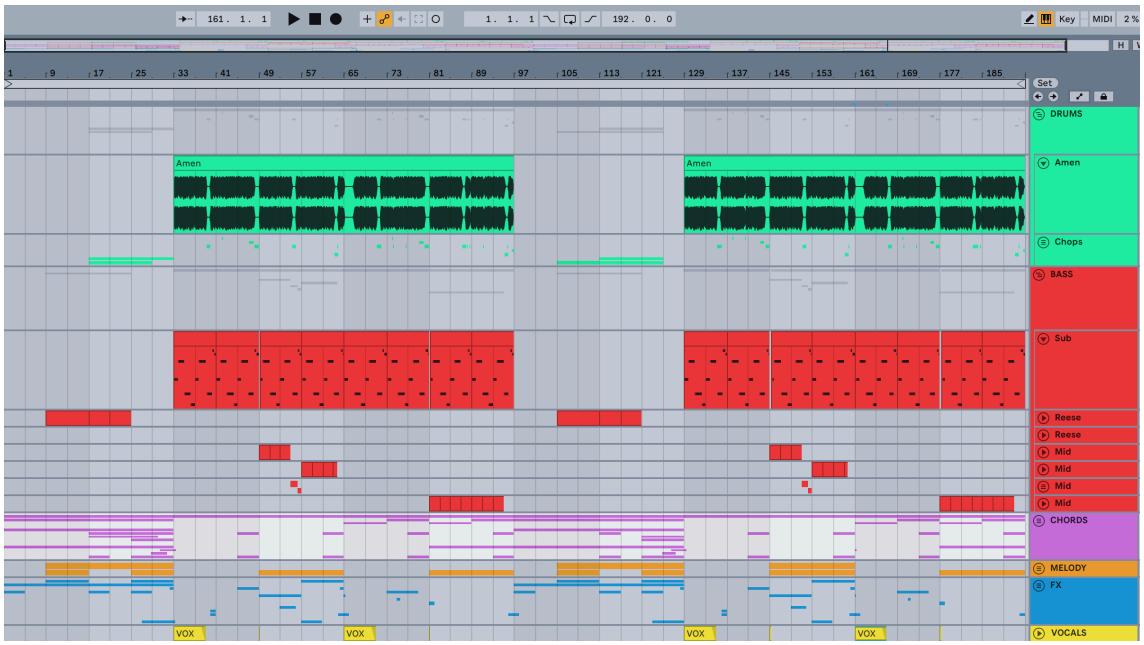

Figure 2.10: An example of music being arranged in a digital audio workstation.

a segmented breakbeat sample; however, this technique can be applied to any sample whether it is a chord progression, melody or bass line.

2.2.4 Arrangement and Structure

Arranging is a crucial aspect of SBEM, which involves organising samples into a coherent composition. SBEM producers most often compose musical arrangements in a DAW using a cut-and-paste method to repeatedly trigger and overlay samples over the course of the track (Morey and McIntyre, 2014). A prevalent technique involves the sequential introduction and removal of various samples at certain time intervals, resulting in a layered structure (Snoman, 2012). Through progressively layering different samples, applying effects (e.g., equalisation, reverb, delay), and automating their parameters, producers can create a sense of depth and movement in their tracks. The layered structure can be used to build tension and release, create dynamic contrasts, and add complexity to a track. This technique is widely used in electronic music genres such as hip-hop, HJDB, house, and techno (Schloss, 2014; Hockman, 2014; Butler, 2006).

Figure 2.10 presents an example of music being arranged in a DAW software, in which the horisontal coloured bars represent the tracks for audio and MIDI information. In this example, the different instruments are grouped and colour coded based on their specific role in the composition: drums, bass, chords, melody, fx, and vocals. For example, the bright green group of tracks comprise different drum and percussion samples, and it can be seen how they are activated over the course of the composition. Grouping tracks based on their instrumentation role provides several advantages in the music production process. This method facilitates a well-organised arrangement, makes it easier to comprehend instrument positions, and enables individual processing of groups which can help to create a balanced mix (Snoman, 2012). Furthermore, focusing on the roles rather than specific instruments simplifies the arrangement process, allowing producers to concentrate on the overall composition. Adopting such an approach helps maintain a clear and efficient workflow while working on various aspects of the arrangement.

Unlike many popular music genres, which typically follow a verse-chorus form, SBEM music often follows a loop-based structure that begins with a sparse introduction that progressively increases in complexity by adding and removing further music material. Furthermore, SBEM producers often structure their tracks in a way that is conducive to DJ performance and mixing, with the intention of facilitating a seamless transition between songs during a DJ set (Butler, 2006). This type of structure can be referred to as a *DJ-friendly* mix; however, producers may release a shorter version of the full track for commercial radio or television (Snoman, 2012). Butler (2006) describes the prototypical form of electronic music using five phases: introduction, buildup, core, breakdown, and outro. The introduction phase contains the fewest number of layers and is often the simplest part of a composition. Its purpose is to establish the primary musical theme and shape audience anticipation for what the track will subsequently provide. For DJ-friendly arrangements, the introduction often includes a simple drum loop that facilitates beat-matching—a skill which involves ensuring that two tracks are playing at the same tempo (or fraction or multiple thereof) and have their relevant metrical pulses (e.g., beats) aligned. Following the introduction, the buildup phase creates anticipation and musical tension by introducing new layers. The buildup phrase often includes sound effects, repetitive melodic motifs, and drum rolls—a technique in which the drum is played rapidly in succession. A prevalent technique used by producers to create tension during the buildup, is to automate the parameters of audio effects to gradually increase or decrease the volume, frequency, or pitch of the various layers and loops. The core phase (sometimes referred to as the *drop*), is the main section of a composition, in which the majority of layers are active. Depending on the genre, this phase can vary in terms of complexity; however, it is typically the most exciting and pleasurable moment for listeners. The core accommodates the main hook—the most memorable part of a composition—which can be represented by a melody, bassline or breakbeat depending on the genre. Solberg (2014) explored the correlations between the intense emotional experiences and production techniques used by EM producers to emphasise the *drop*. The breakdown phase is utilised to drastically reduce the energy through the removal of the main elements and layers. Its purpose is to provide a momentary pause and an opportunity to reset the energy level after a high-intensity core. The outro is the final phase of an arrangement, in which layers are progressively reduced to silence. Similar to the introduction, a DJ-friendly mix will include a simple drum loop to facilitate a transition into the next song.

2.3 Drum Creation in Sample-based Music

Drums and percussion play a significant role in shaping the rhythmic foundation of many music genres, particularly in SBEM (Greenwald, 2002; Morey and McIntyre, 2014). However, in contrast to genres such as pop and rock, where vocals or guitars are the most dominant elements, drums are typically at the forefront of many SBEM productions and play a crucial role in shaping the overall sound and quality of a composition (Hockman, 2014). Furthermore, many SBEM genres are differentiated by the choice of drum sounds, tempo, and rhythmic patterns, such as the use of the Roland TR-909 drum sounds at around 120 BPM and an isochronous kick drum rhythm in house music (Butler, 2006) or the use of fast-paced breakbeats in HJDB (Hockman, 2014). These distinct timbral qualities and rhythmic patterns are essential in defining the characteristics of each genre. Hence, music producers spend a great deal of time and effort into the creation of drum sounds and rhythmic patterns that are coherent and interesting (Hewitt, 2009). There are several methods for incorporating drums into an SBEM composition including

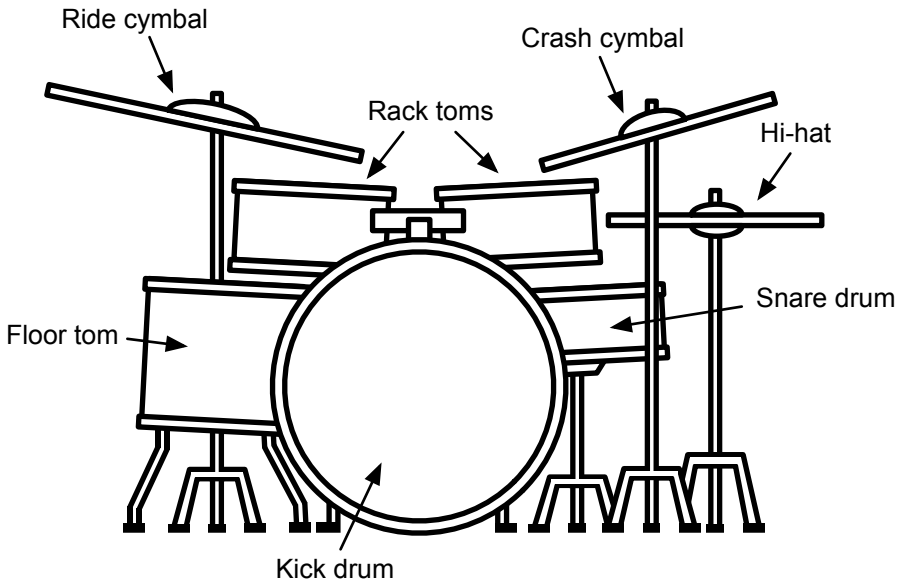

Figure 2.11: An illustration of the standard western drum kit.

recording a physical drum set, sampling percussion elements from existing recordings, and synthesising drum sounds from scratch.

2.3.1 The Drum Kit

The main types of drums that are sought after by SBEM producers are based around the modern drum kit. The drum kit is a fundamental component of the typical rhythm section and finds its application in various traditional music genres, spanning from rock, pop, blues and jazz. It is comprised of multiple instruments that are played independently by a single musician (i.e., a drummer). Figure 2.11 illustrates a standard drum kit consisting of a kick drum, a snare drum, cymbals and toms.

Kick drum: A kick drum—otherwise known as a bass drum—is the largest drum in the standard drum kit and one of the most important elements in modern dance music. It provides the essential low-frequency foundation and is often used to establish phrasing by accentuating the downbeat positions, giving a sense that a song is driving forward (Greenwald, 2002). The kick drum is composed of two drumheads encapsulated in a shell: the batter head, which is the striking side of the drum, and the resonant head, which responds to the impact of the batter head (Rossing et al., 2014). It is played using a foot-operated pedal, which is connected to a beater. When the drummer presses the pedal, the beater strikes the batter head, creating the initial transient of the drum sound which typically lasts for a few milliseconds. The impact of the hit causes the rest of the drum to resonate over a longer period. The sound of an acoustic kick drum is determined by various factors, including the tension of the drumheads, the diameter and construction of the shell, and the materials used for the shell, the drumheads and the beater (Rossing et al., 2014). These variables affect the pitch and timbral characteristics of the drum, allowing drummers to achieve a wide range of sounds to suit different musical styles. In addition, the decay time of an acoustic kick drum can be altered by using a blanket or duvet placed inside the drum to dampen the vibrations (Bartlett and Bartlett, 2009). This technique is often used in recording studios and live performances to achieve a more controlled sound.

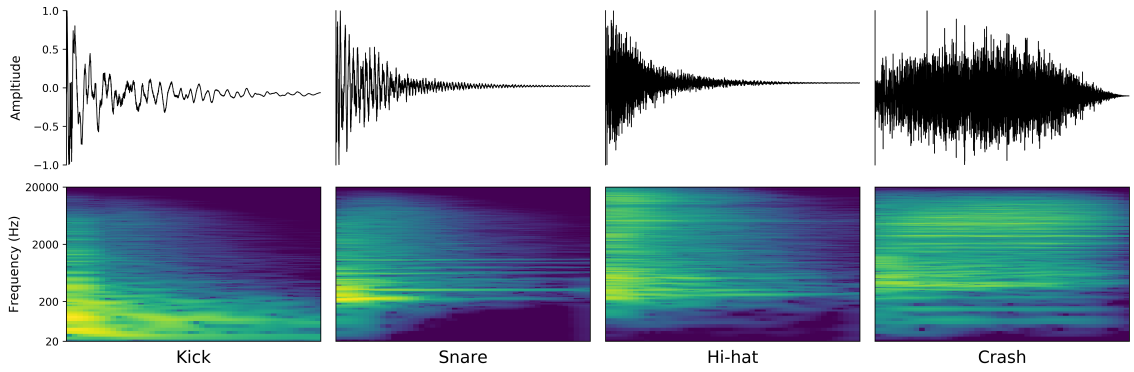

Figure 2.12: Waveform and spectrogram audio representations of common drum kit components—kick drum, snare drum, hi-hat, and crash cymbal, respectively.

Snare drum: The snare drum is made of a cylindrical drum shell, usually consisting of metal or wood, and features a series of snare wires that are stretched across the resonant head of the drum. When the batter head of the drum is stuck, these wires vibrate against the resonant head, producing a sharp staccato sound. The presence of snare wires causes the vibrational modes of the drum to become more complex and unpredictable, while also creating a unique, inharmonic noise that contributes to the overall character of the snare (Rossing et al., 1992). The snare drum can be struck in a variety of ways, each resulting in a unique sound. For example, the velocity of the strike plays a significant role in the sound produced, with a softer hit resulting in a quieter and more muted sound (referred to as ghost notes), while a harder strike produces a sharper and more vibrant tone, often exciting the snare wires to a greater degree. In addition, striking both the drum head and rim simultaneously (known as a rimshot) creates a distinct sound characterised by its sharp attack and high volume. These playing techniques provide the drummer with a wide range of sonic possibilities and are often used to add texture and dynamics to a musical performance (Southall, 2019). In hip-hop and jungle music, the snare drum is often used to emphasise beats two and four, while ghost notes are used between accented beats to add colour and texture (Greenwald, 2002; Hockman, 2014).

Cymbals: Cymbals are a type of idiophone, which are self-sounding instruments that produce sound through the vibration of their own metallic material (Marcuse, 1975). Flat circular plates are used to create cymbals, which produce bending waves that travel both circularly and across the diameter (Fletcher and Rossing, 1998). Cymbals are typically made of an alloy comprised of varying amounts of copper, tin and silver, each with its unique tonal character (Petrella, 2002). The sound of a cymbal also depends on its diameter, curvature and thickness, with more random modes producing inharmonic and non-definable pitch profiles that are similar to full spectrum noise (Fletcher and Rossing, 1998). Other contributing factors of the sound depend upon the position of the cymbal that is struck (e.g., the bell or profile), the velocity it is struck, and the material used to strike it (e.g., a wooden stick, mallet, or soft beater) (Rossing, 2001). The hi-hat comprises two cymbals mounted on a stand and the interaction between them is controlled using a foot pedal. Hi-hats play a fundamental role in drumming and are widely used to maintain rhythm. The hi-hat can be played in an open (i.e., open hi-hat) or closed (i.e., closed hi-hat) position, with a closed position producing a dampened, less energetic sound and an open position resulting in more energetic noise. The opening and closing of the hi-hat cymbals creates distinct sound variations that can be utilised to add depth and texture to a performance. Other cymbals commonly used in drum kits include the ride, crash, and splash cymbals. These cymbals are typically

Figure 2.13: Roland TR-909 drum machine (left), XLN Audio Addictive Drums 2 (right).

used to emphasise specific musical sections and build the energy and momentum of a performance (Southall, 2019).

Toms and other percussion: The standard drum kit also includes rack toms, which are mounted to the kick drum; and floor toms, which are usually positioned under the ride cymbal. Toms are often used in drum fills and solos that typically occur during transitional points in music (e.g., between verse and chorus) (Stewart, 2000). The drum kit can often be accompanied by other percussion instruments such as bongos, congas, hand claps, tambourines, and shakers. (Greenwald, 2002).

Producers may choose to use live recorded drums rather than synthesised drums for a variety of reasons. Firstly, the use of live drums can provide a more authentic and natural sound, which is often sought after in genres such as rock, jazz, and blues. The variations in performance and unpredictability that come with live drumming can make the sound more organic and can complement the overall rhythm (Greenwald, 2002). Secondly, the unique acoustic qualities of live drums, which are influenced by the room, microphone placement, and other factors, can add character and depth to the sound (Bartlett and Bartlett, 2009). Furthermore, different recording techniques such as close-miking or room-miking can be used with live drums to achieve different tonal and spatial characteristic (Toulson, 2021). Additionally, some producers simply prefer the aesthetic of live drum recordings over synthesised sounds for their specific style of production or mixing (Hockman, 2014).

2.3.2 Drum Machines and Virtual Drumming Software

In addition to the vast collections of pre-recorded drum samples available on the internet, various commercial tools and equipment have been designed to create and manipulate drum samples, providing even more opportunities for experimentation and customisation. These tools provide a cost-effective and efficient way to explore different drum sounds and techniques, eliminating the need for expensive recording equipment or a live drummer. The drum machine—an electronic instrument for creating percussion sounds and programming drum patterns—has had a lasting impact on popular music since the release of the Roland TR-808 in the 1980s and its predecessor the TR-909. Drum machines are popular electronic instruments that imitate a drum kit through the creation and arrangement of synthesised percussion sounds. The left side of Figure 2.13 displays the front panel of the Roland TR-909, which includes various parameters for tuning the pitch, timbre, and dynamics of different types of drums. Additionally, it features a row of sixteen keys at the bottom that allow drum events to be programmed and provide a visual representation of a 16-step sequence.

Another example is virtual drumming software (VDS), which simulates the recording and tuning parameters of acoustic drum kits, allowing users to create realistic and expressive drum sounds without the need for recording equipment or a live drummer. Typically, VDS is sample-based and includes carefully curated and processed recordings of live drummers to capture the nuances of different drumming styles and techniques. Moreover, VDS is controlled through MIDI and often includes libraries of symbolic drum loops that can be sequenced to create complex rhythmic patterns, providing users with a wealth of creative possibilities. Examples of VDS software include XLN Audio's Addictive Drums,[13] FXpansion's BFD3[14] and Toontrack's SuperiorDrummer.[15] The user interface of Addictive Drums 2 is displayed on the right side of Figure 2.13. It features a diverse range of instruments from the drum kit, which can be easily replaced or edited through an array of processing functionalities such as compression, saturation, and reverb. At the bottom of the interface, there are several volume faders that enable the user to balance the mix of individual drum sounds, as well as adjust the levels of the microphones used during recording, such as overheads and room mics. Despite the high quality of virtual drumming software libraries, the user experience of these applications can be negatively impacted by their large size, suboptimal user interfaces, and an excessive amount of complex parameters. Furthermore, the level of control over a sound can be limited by the fact that transformations are tied to multi-sample recordings.

2.3.3 Drum Synthesis

As an alternative to sampling, music producers can create drum sounds using a wide variety of synthesis techniques. Synthesis provides a considerable amount of control and flexibility, allowing the characteristics of a drum sound, such as the pitch and timbre, to be carefully sculpted by the user. Music producers will often use a combination of synthesis and sampling techniques in order to achieve a balance of both precision and character in their productions. There are numerous commercially available synthesisers and audio plug-ins, such as the Arturia DrumBrute[16], the Sonic Academy Kick 2,[17] and Blck Noir by Endorphin.es,[18] that offer different interfaces and functionalities to assist music producers in crafting their desired drum sounds.

Perhaps the simplest method for synthesising audio is additive synthesis, in which complex sounds can be created by combining multiple discrete sinusoidal waveforms (Smith, 2010). Figure 2.14 demonstrates how a simple kick drum can be created by modulating the amplitude and pitch of a single sinusoidal waveform over time. This simulates a beater striking the stretched surface (i.e., *batter* head) of an acoustic kick drum, which creates an initial high-frequency sound that drops in pitch and amplitude over a short period of time. A wide range of sounds with more complex timbres can be created by adding more sinusoidal waveforms of different frequencies, amplitudes and phases. Although the additive synthesis parameters are somewhat close to human perception, this method can become computationally expensive as many components are required to model the rich texture of natural sounds. Another option is subtractive synthesis, in which a sound source containing many harmonics, such as a sawtooth, square waveform, or a noise signal, is manipulated by subtracting frequencies using a filter (or bank of filters) (Miranda, 2012). Subtractive synthesis offers less precision than additive synthesis in terms of individual

[13] https://www.xlnaudio.com/products/addictive_drums_2
[14] https://www.fxpansion.com/products/bfd3
[15] https://www.toontrack.com/product/superior-drummer-3
[16] https://www.arturia.com/products/drumbrute
[17] https://www.sonicacademy.com/products/kick-2
[18] https://www.endorphin.es/modules/p/blck-noir

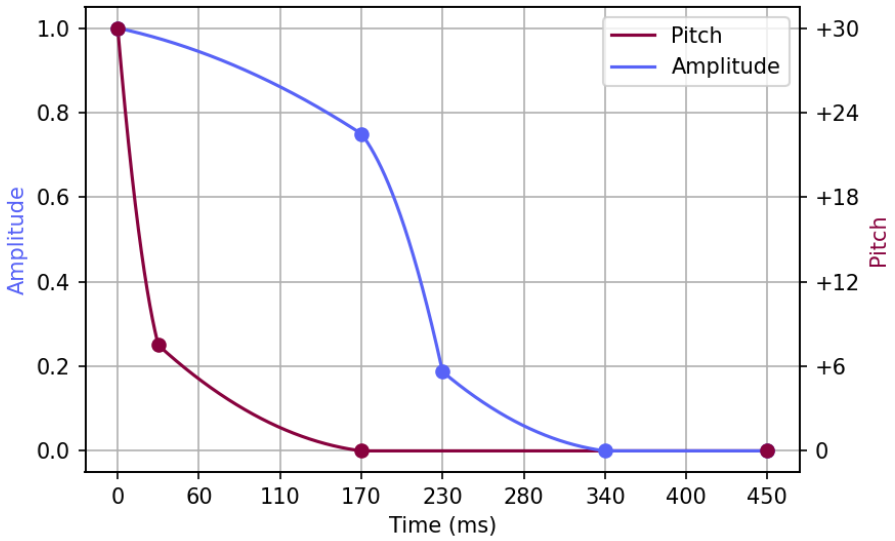

Figure 2.14: A demonstration of the pitch (red) and amplitude (blue) envelope parameters used to synthesise a kick drum.

harmonic control; however, it is computationally efficient, intuitive, and easy to control, making it a popular choice in electronic music production (Miranda, 2012). Furthermore, subtractive synthesis is often correlated with the production of warm and sonorous sounds and is implemented in popular analogue drum machines (e.g., Roland TR-808 and TR-909). Frequency modulation (FM) synthesis (Chowning, 1973) is another synthesis technique that can be used to create percussion sounds. FM synthesis can create complex waveforms by modulating the frequency of a carrier signal with various other signals. For example, the frequency of a basic sine wave can be modulated using a series of sine waves at different frequencies to emulate the inharmonic partials heard in a cymbal. The Karplus-Strong algorithm (Karplus and Strong, 1983) can also be used to synthesise percussive sounds. Karplus–Strong synthesis uses a delay line to create a synthetic sound that mimics the behaviour of a vibrating medium, such as a plucked string or a struck drum. The energy loss of a string or a drum is simulated using a buffer of samples and a feedback loop to excite the buffer, which in turn creates the sound. By shortening the delay time and increasing the feedback gain, the Karplus–Strong algorithm can be used to generate percussive sounds that have a sharp attack and a short decay, similar to those of a drum or a cymbal.

Physical modelling is a method of sound synthesis that mimics the behaviour of real-world physical systems to create sounds. In this approach, the physical properties of percussion instruments are simulated using mathematical models. By applying the principles of acoustics and elasticity, these techniques provide a physical representation of the main vibrating components of musical instruments through the use of partial differential equations (Rabenstein and Trautmann, 2001). This allows the accurate synthesis of a wide range of sounds that cannot be easily reproduced using traditional waveform synthesis methods. Physical models have been developed for snare drums (Torin et al., 2014), bass drums (Bilbao et al., 2019), and a variety of other percussion instruments (Fontana and Rocchesso, 1998; Bilbao and Webb, 2013; Bilbao et al., 2019). Physical modelling of sound is highly regarded for its desirable characteristics in synthesis, control, and expressiveness. However, it has some limitations,

such as the control parameters not being directly related to the produced sound and a high number of parameters being required to model realistic data (Serra, 2007).

2.4 Related Research

The previous sections have provided an overview of music sampling and the standard practices employed in SBEM production, emphasising the fundamental concepts of sample sourcing, selection, manipulation, and arrangement. Modern SBEM production often entails exploring digital audio collections, such as sample libraries and recorded music, while carefully selecting samples based on aesthetic judgment. Producers then adapt, manipulate, and arrange these samples to create unique compositions. This section offers an in-depth examination of research related to SBEM, concentrating on techniques for the automated analysis, retrieval, and generation of electronic music samples. The objective of this review is to showcase both the advancements and current constraints in the existing body of research, thereby contextualising the systems developed in this thesis within the wider scope of SBEM research.

2.4.1 Music Tagging

In the field of MIR, there has been a substantial effort in researching and developing techniques for automatic music tagging (AMT) as a means of addressing the labour-intensive process of manual annotation. AMT is a subtask of music classification, a broad field that encompasses any process of categorising music based on its features or characteristics, such as genre, style, and instrumentation. Music classification can be based on a variety of factors and may involve objective or subjective criteria. For example, music could be classified based on its tempo, genre, or lyrics, or it could be classified according to cultural or historical contexts. Automatic music classification is typically achieved using machine learning algorithms to automatically assign musical audio into predefined categories or classes based on its audio content. Features, such as tempo, pitch, and key, are extracted from the audio and provided as input to a machine learning model. The model is trained through the use of a labelled dataset of musical examples, and as a result of this process, it develops the capability to accurately predict the relevant categories for newly introduced musical content. This can be useful for a variety of applications, such as music recommendation systems, music databases, and music education. There are many music classification tasks including, genre classification (Tzanetakis and Cook, 2002; Collins, 2012), artist identification (Mandel and Ellis, 2005), instrument classification (Marques and Moreno, 1999; Heittola et al., 2009), and music similarity (Slaney et al., 2008). To perform automatic music classification, algorithms typically extract features from the audio content of the music, such as tempo, pitch, and key, and use these features as input to a machine learning model. Some prevalent algorithms for automatic music classification include clustering, k-nearest neighbours, support vector machines and neural networks. Overall, automatic music classification has the potential to make the process of organising and labelling music more efficient and accurate, eliminating the laborious task of human annotation.

AMT is a multi-label classification problem, in which a single instance of audio can be assigned to multiple attributes. AMT can be of use in music streaming services, music libraries, and other applications where it is necessary to quickly and accurately identify the meta-data for a large number of music files using tags. Estimating music tags from audio requires the extraction of appropriate acoustic features.

Conventionally, AMT was achieved using classification models that learn a mapping between a predefined vocabulary of semantic tags and human-engineered audio features that describe rhythmic, timbral and tonal content (Tzanetakis and Cook, 2002; Mandel et al., 2011; Sordo et al., 2012). The main limitation of this type of approach is that considerable expertise about the problem is required to design features that uncover relevant information in the input. Additionally, the models rely on the capacity of the hand-crafted features to capture relevant information from the audio, which may not be particularly optimised for the task.

Current AMT methods utilise data-driven deep learning architectures to jointly optimise feature extraction and classification. Deep learning systems utilise neural networks to learn an internal representation for both audio feature detection and classification, reducing the requirement for manual feature engineering. Neural networks are trained using annotated data and can estimate multiple classes associated with a song or excerpt. Convolutional neural networks (CNNs) (described in detail in Chapter 3) are a type of deep learning architecture that emulates the human visual system by utilising trainable filters to extract patterns from images. Motivated by the success in the computer vision domain, CNNs have been adopted by MIR researchers to tackle various audio classification problems. One of the key preprocessing steps in applying CNNs to audio-related tasks is the transformation of raw audio data into a time-frequency representation, such as a spectrogram. This transformation is often necessary as it allows for the representation of audio signals in a format that better captures the temporal and spectral characteristics inherent in audio data.

Choi et al. (2016) presented a fully convolutional neural network for AMT that uses multiple stacks of convolutional layers to observe local harmonic structures at different time-frequency resolutions. As an alternative, Pons et al. (2018) investigate architectures that incorporate domain knowledge by designing musically-motivated filters. Custom vertical and horizontal filter sizes are used to efficiently model the timbral and temporal characteristics present in spectrograms. Lee et al. (2018) propose an end-to-end learning strategy for AMT that operates directly on raw audio samples using sample-level filters that analyse short segments of the audio signal. Sample-level filters are able to extract local features, providing a fine-grained representation of the audio data. The benefit of learning directly from raw audio is that the networks can autonomously discover frequency decompositions and learn features that are invariant to phase and translation without assumptions imposed by engineered audio input representations (Dieleman and Schrauwen, 2014). Choi et al. (2017) present a convolutional recurrent neural network for AMT, in which a CNN is utilised for local feature extraction and a recurrent neural network for temporal summarisation of the extracted features. This hybrid approach enables the model to account for the sequential nature of music. An alternative approach, to achieve temporal summarisation of the extracted features, is to use a transformer architecture (Devlin et al., 2019) with the self-attention mechanism, as discussed in (Won et al., 2019, 2021a). The self-attention mechanism allows the model to weigh the importance of different elements in the sequence as it processes them, rather than using a fixed ordering as in recurrent neural networks. Transformers have also been proposed to jointly model both spectral and temporal sequences of an input time-frequency representation (Lu et al., 2021). Furthermore, Won et al. (2020a) consider the role of harmonic structure in human auditory perception and present a harmonically-stacked trainable representation that takes advantage of the harmonic information to yield more efficient audio representations.

It is often difficult to directly compare the performance of the various deep learning methods for a specific task due to differences in software versions and training data used in the evaluations. To help

mitigate this issue for AMT, Won et al. (2020b) evaluated various CNN architectures under a consistent experimental set-up. The aforementioned CNN-based ATM models are trained using three annotated music tagging datasets: MagnaTagATune (Law et al., 2009), Million Song Dataset (Bertin-Mahieux et al., 2011), and MTG-Jamendo (Bogdanov et al., 2019). These datasets contain a variety of different tags including, genre, instrument and mood. Model performances are evaluated using the area under receiver operating characteristic curve (ROC-AUC) and the area under precision-recall curve (PR-AUC) evaluation metrics. Additionally, the authors investigate the robustness of each model by assessing their ability to generalise to audio deformations introduced by applying time stretching, pitch shifting, dynamic range compression and the addition of white noise. In their experiments, a CNN trained on log-scaled Mel spectrograms of short audio excerpts performed best when training on the original dataset; however, when trained on the augmented data, the harmonic CNN showed better generalisation abilities against each of the different audio effects. In a recent approach, (Won et al., 2021b) proposed multimodal metric learning for tag-based music retrieval, which establishes a distance metric to measure data similarity, enabling direct nearest neighbour searches in the embedding space.

2.4.2 Instrumentation Role Classification

Established sample libraries commonly categorise samples by their role in a composition, such as bass, melody, and sound effects (Section 2.1.4). Automatic instrumentation role classification (AIRC) is a recently proposed MIR task that involves automatically labelling audio loops based on these roles. By automatically labelling audio loops according to their specific role, producers can more efficiently organise, manage, and navigate their personal collections of samples. With sufficient training data, automatic music tagging systems can be employed to predict the instrumentation roles of samples (Ching et al., 2020). Research in AIRC has been facilitated by the development of the Freesound loop dataset (FSLD) (Ramires et al., 2020b), a large public collection of loops and corresponding instrumentation role annotations from Freesound. Ching et al. (2020) benchmarked AIRC performance of deep learning and other machine learning models using a subset of loops from FSLD, with the best performing model being a harmonic CNN (Won et al., 2020a). Despite this, there is potential for further exploration and improvement, such as investigating alternative models and examining additional applications of AIRC.

2.4.3 Structure Analysis

The analysis and visualisation of music structure can play a crucial role in promoting and enriching the understanding and appreciation of musical works among diverse audiences, including listeners, performers, composers, and musicologists. Music structure analysis involves dissecting a musical composition into its fundamental components and elucidating the connections among them. This process involves identifying elements such as musical form (e.g., AABA, sonata-allegro), sections (e.g., verse, chorus, bridge), transitions, key changes, and repetitions. These insights are invaluable for efficiently navigating extensive music collections to locate specific segments of tracks with desired musical characteristics or structural elements. This utility is especially evident in sample discovery and sample-based music production. For example, Levy et al. (2006) proposed a method for characterising the high-level structure of music based on the consistent distribution of timbre-types, which can be loosely associated with specific combinations of instruments. By analysing timbral features and segmenting the music based on these timbre-types, this approach can identify structural segments within a piece of music. Moreover, this approach has the

potential to assist in sample discovery by enabling listeners to search for sections of songs with a similar overall timbre.

Within the field of MIR, several researchers have developed methods to assist with the analysis of electronic music (EM) structure and arrangement. Rocha et al. (2013) developed an algorithm to identify key structural sections in EM by segmenting compositions and subsequently evaluating the timbral similarity between the segments. Scarfe et al. (2014) proposed an algorithm capable of segmenting a full EM DJ mix into individual tracks using self-similarity. Yadati et al. (2014) and Aljanaki et al. (2014) present strategies for locating the most anticipated section of electronic dance music (referred to as *the drop*), using user-generated tags from online social platforms. Alternatively, Glazyrin (2014) investigate the problem of separating a full DJ mix into single tracks using an iterative algorithm based on spectral features. Vande Veire and De Bie (2018) proposed an automated DJ system for drum and bass music that utilises several genre-specific MIR techniques for beat tracking, downbeat tracking, and structural segmentation, to obtain an understanding of the musical structure. More recently, Kim et al. (2020) present a method for aligning a mix to its original music tracks, from which cue points are obtained and used to segment a continuous DJ mix into individual tracks.

Loop activation transcription (LAT) is a task that derives key structural information from loop-based music (Smith and Goto, 2018). LAT involves identifying the types of loops (e.g., instrumentation roles), and their active periods within a composition. This information could serve as a valuable visual aid for producers and DJs, enabling them to anticipate upcoming sounds (e.g., drums, bass). Furthermore, would be particularly beneficial for tasks such as automatic DJing (Vande Veire and De Bie, 2018), mashups (Davies et al., 2014), and the identification of key structural events in SBEM (Yadati et al., 2014). To this end, López-Serrano et al. (2016) proposed a method for transcribing loop activations using non-negative matrix factorisation (NMF) deconvolution Smaragdis (2004) to estimate spectral templates and rhythmic activations from magnitude spectrograms. Additionally, Seetharaman and Pardo (2016) proposed an NMF method for the simultaneous segmentation and source separation of loop-based music. Building on this research, Smith and Goto (2018) introduced an alternative method for source separation and the LAT using non-negative tensor factorisation (FitzGerald et al., 2006). This approach was subsequently integrated into an interface designed to facilitate the extraction and remixing of loops (Smith et al., 2019). While the aforementioned approaches allow for the separation of mixed audio into the constituent loops, they rely on non-varying repetitions of loops and do not optimise independence between roles. This presents a challenge when transcribing loop activations in more intricate SBEM compositions, where multiple instruments may fulfil the same role and often exhibit variations through automation and resequencing (Section 2.2.4). SBEM compositions frequently feature numerous instances of melodic, harmonic, and percussive content, with the instruments and timbre changing while maintaining their original role within the piece. This complexity makes it difficult to enforce separation between the roles with NMF-based methods, resulting in multiple templates for each role.

2.4.4 Sample Identification

Knowing the source of a sample can provide valuable information about the history and origins of the sample, as well as help with legal and ethical issues related to crediting the original music (Van Balen et al., 2012). This knowledge can also be beneficial for organising music collections based on sample

usage. For example, a DJ could search their collection for tracks that have incorporated the well-known *Amen break*. Due to these benefits, researchers have addressed the problem of detecting the use of samples in pre-recorded music. Van Balen et al. (2012) introduced the problem of automatic sample identification, situating it in the field of content-based music retrieval. The proposed system utilises an audio fingerprinting technique (Wang et al., 2003) that extracts content-based signatures from audio to detect whether a query song contains a sample inside a given music collection. Dittmar et al. (2012) present a toolbox based on NMF to facilitate the inspection of suspected music plagiarism (i.e. the use of another work while presenting it as original music). Their proposed system can detect sampled material by deriving spectral templates from a source sample through NMF, then detecting the presence of the template in the target song. Whitney (2013) also present a method based on NMF; however, their automatic sample recognition system is suited to hip-hop and is robust to audio manipulations prevalent in the genre, such as frequency equalisation, time-stretching, and pitch modifications. Gururani and Lerch (2017) propose an alternative system that employs dynamic time warping (Müller, 2007) to compute an alignment path between NMF activations of a song and a query sample. Features are extracted from the alignment path and used to train a random forest classifier (Breiman, 2001) that learns to detect the presence of a sample in a given song.

In addition to sample identification, researchers have delved into the related task of determining whether a song can be considered a derivative work of another. Notably, Casey and Slaney (2006b) introduced an algorithm tailored for discovering similarities between songs, primarily by analysing intersections between segments of audio data. This algorithm utilises locality-sensitive hashing, a technique for approximating similarity within a dataset of audio shingles—short, overlapping segments of audio features. Once these shingles are hashed, they can be utilised to create an index or data structure that allows for efficient storage and retrieval of similar shingles. This approach has proven effective for remix detection (Casey and Slaney, 2007), where identifying derivative works and their original sources is crucial. Moreover, its application could extend to the retrieval of similar audio samples that closely resemble a reference.

2.4.5 Sample Retrieval

The process of retrieving samples continues to be a crucial challenge in the creative process of producing electronic music (Andersen and Knees, 2016). Automatic sample retrieval concerns the task of developing methods that can facilitate the navigation of collections of music samples. Several MIR studies have focused on the development of browsing interfaces that can facilitate the navigation of collections of drum samples. Pampalk et al. (2004) introduced an approach for automatically structuring drum loops, in which similar loops are clustered based on a similarity measure and then hierarchically organised using self-organising maps (SOM) (Kohonen, 2001). Drum sample retrieval has also been approached through a query-by-vocal, in which a vocalised input imitating a drum sound, is provided to a system that searches for corresponding sounds in a database of drum samples (Kapur et al., 2004; Gillet and Richard, 2005; Mehrabi et al., 2018; Delgado et al., 2022). Fried et al. (2014) proposed a system that arranges audio samples in a two-dimensional layout based on user preferences for efficient navigation within collections of snare drum and synth sounds. Shier et al. (2017) presented a browsing interface for kick and snare samples based on feature similarity, in which features are extracted from the audio using the Essentia Library (Bogdanov et al., 2013) and principal component analysis (PCA) is used to reduce the dimensionality of the feature space. Drum samples could then be mapped to a grid interface using the first two dimensions obtained from PCA. Dimensionality reduction for drum sample retrieval has

also been explored using a student-t stochastic neighbour embedding (t-SNE) (Turquois et al., 2016), and Shier et al. (2021) evaluated various other manifold learning techniques for dimensionality reduction tasks. López-Serrano et al. (2017) presented a system for the retrieval breakbeat samples in funk, soul, and jazz recordings. The system leverages an approach for automatic singing voice detection (Lehner et al., 2014), and incorporates features obtained through harmonic-residual-percussive source separation to identify the location of breakbeats in the original recordings. Bruford et al. (2019) presented a visual interface for navigating libraries of symbolic drum loops, in which SOMs are used to map symbolic drum loops onto a two-dimensional space according to rhythmic similarity. Kim and Nam (2020) took inspiration from music producers who use reference songs to find instrument samples and propose a query-by-example system that takes mixed audio as a query and retrieves single audio samples. Sample retrieval has also been approached through visual sketches, that express the mental model associated with a sound, to query a database of samples (Knees and Andersen, 2016; Engeln et al., 2021). More recently, Lattner (2022) proposed a system for automatic drum sample retrieval based on aesthetic principles learned from the data, assisting producers by automatically ranking the samples in their library by how well they fit the current musical context at different stages of the production process.

Multiple methods in literature have followed the adaption of corpus-based concatenative synthesis to jointly extract and organise samples from a database of pre-recorded music Coleman (2007); Schwarz et al. (2006). These methods rely on a two-stage process: 1) sample segmentation through onset detection to segment pre-existing music into samples at each of the estimated onsets (Dupont et al., 2009; Coleman, 2007) or by identifying repeated patterns though self-similarity (Ong and Streich, 2008; Streich and Ong, 2008), and 2) organisation of extracted samples based on the similarity between hand-crafted features that describe musical properties such as timbre, harmony and rhythm (Streich and Ong, 2008; Coleman, 2007; Dupont et al., 2009; Roma and Serra, 2015). The samples are then visualised as graphical objects in a graphical user interface (GUI), where a user can search for sounds of interest using feature descriptors as navigation controls. The main limitation of these systems is that they are designed with pre-assumptions that a given collection samples will have an interesting variation along a given set of feature descriptors. Roma et al. (2019) addressed this issue, presenting an alternative framework for learning features directly from a corpus of samples and comparing various dimensionality reduction techniques to map high-dimensional summaries of sounds into a lower-dimensional space.

In addition to these methods, audio similarity-based retrieval techniques could contribute to the discovery of audio samples within extensive music collections. For example, Casey and Slaney (2006a) introduced a method that leverages temporal sequences of harmonic features to identify musical passages within a collection of compositions that are similar to a requested song. This method would enable producers to search their music catalogue for potential samples that relate to a given reference. Moreover, methods for automatically separating an audio recording into isolated sources could enhance the sample retrieval process. Topel and Casey (2011) demonstrated the use of probabilistic latent component analysis-based decomposition to retrieve specific sonic components from music and then utilised these samples to compose new music. These approaches provide alternative means for discovering and extracting audio samples from existing collections of music, facilitating more efficient sample retrieval and new ways to compose music.

Other related MIR studies have been proposed to assist in the retrieval of compatible audio segments based on various criteria. Kitahara et al. (2015) presented a loop sequencer that automatically selects music loops in techno music based on temporal evolution of features related to excitement. Inspired by

music *mashups*—music compositions that are created through the combination of two or more songs to create a hybrid recording (Shiga, 2007)—multiple systems exist for assessing the compatibility of musical segments. Davies et al. (2014) presented a measure of *mashability* that considers rhythmic, harmonic and spectral compatibility of phrase segments. Systems for creating automatic music mashups rely on three main sub tasks: 1) rhythm analysis to identify beat and downbeat locations, 2) structural analysis for identifying musical phrases and 3) a measure of *mashability*. The *mashability* measure has shown to be useful in applications including 'The CrossSong Puzzle' (Smith et al., 2017) and for singing voice identification (Lee and Nam, 2019). Shi and Mysore (2018) presented an interactive system for creating and selecting loops from pre-recorded music. Loops are identified by dividing a song into segments and measuring the similarity of each segment using features that describe harmony, timbre and energy. The main limitation of the aforementioned approaches is that the hand-crafted representations cannot fully describe all features in a musical segment and it can be possible for audio segments to be compatible despite different harmonic and rhythmic content.

More recent approaches that assess *mashability* utilise deep learning algorithms that can capture complex compatibility relationships between two segments of musical audio. Chen et al. (2020) presented a method for automatically combining loops using a neural network that estimates the compatibility between two one-bar loops. As a labelled dataset to train a model for this particular task does not exist, the authors propose a data generation pipeline for creating positive data using a loop extraction algorithm (Smith and Goto, 2018) and compare various strategies for choosing negative examples, such as random combinations of loops, reversing and rearranging beat segments. Huang et al. (2021) constructed a data generation pipeline by taking advantage of music source separation algorithms (Andreas et al., 2017; Prétet et al., 2019) to separate pre-recorded music into isolated stems (e.g., vocals, drums, and bass). Positive examples for training their model are achieved using stem tracks from the same song as well as random combinations of stems with matching key and tempo. Random combinations of stems with different keys and tempo are used negative examples. Alternatively, Bernardo and Bernardes (2021) compute a population of compatible mashups through from loop recombinations an artificial immune system algorithm (De Castro and Timmis, 2002). The creation of optimal mashability is achieved by finding local minima in a feature space that objectively represents the harmonic and rhythmic compatibility of the audio loops.

Existing research on sample retrieval has primarily focused on devising methods to assist in navigating sample collections. Apart from the breakbeat retrieval system proposed by López-Serrano et al. (2017), automated systems that assess existing music recordings and pinpoint samples meeting specific criteria have received limited attention. Moreover, while numerous systems offer methods and interfaces for identifying samples, organising them, and evaluating their compatibility based on similar characteristics, the capacity to continuously explore and uncover intermediate samples within collections would represent a valuable advancement.

2.4.6 Neural Audio Synthesis

Producers are frequently constrained by the limited timbre diversity offered by sample packs or synthesis techniques. As a result, they often need to use complex production techniques such as layering, compression, and equalisation to achieve their desired sound. Moreover, the process of navigating through sample packs or making adjustments to synthesiser controls can prove time-consuming and

demanding, significantly impeding the creative flow. A new and promising approach to synthesis is through the use of deep learning models. Neural audio synthesis employs deep learning models to manage the synthesis process, allowing for continuous exploration of a diverse range of sounds (Esling et al., 2018a; Engel et al., 2019, 2020). These models can learn expressive and intuitive latent variables, enabling more effective sound exploration and synthesis. Neural audio synthesis has demonstrated significant success in experimental settings and has led to the development of production-ready music creation tools, such as Mawf,[19] Neurorack,[20] and DrumGAN.[21]

Neural audio synthesis has emerged through advancements in deep generative models (DGM), including neural autoregressive (AR) models (Bengio et al., 2001; Uria et al., 2014a), variational autoencoders (VAE) (Kingma and Welling, 2014) and generative adversarial networks (GAN) (Goodfellow et al., 2014). A comprehensive exploration of these models is provided in Section 3.5. With appropriate training data, DGMs can learn the underlying patterns and structures of audio signals, offering innovative ways to efficiently generate audio. These models enable the creation of new audio samples without being restricted to hand-designed components, such as oscillators and wavetables, or a specific synthesis process, such as those discussed in Section 2.3.3.

Sarroff and Casey (2014) demonstrated the use of autoencoders (AE) as synthesisers, introducing an interactive musical audio synthesis system. AEs consist of two neural networks: an encoder, which learns to extract a compact encoding from an input, and a decoder, which learns to invert the encoding back to its original form, allowing for data compression and reconstruction. The system learns a high-level representation from audio data by training an AE to reconstruct spectrograms from a diverse dataset of music spanning various genres. Interaction with the system is enabled through a real-time interface, where users gain access to the innermost hidden layer of the AE for interaction. This allows for audio streaming through the model, and users can modify the hidden units to manipulate specific aspects of the input audio. Alternatively, the encoder can be substituted with a subset of arbitrary hidden units, which can then be processed by the decoder to synthesize audio without the need for an initial input. Another influential contribution in this field was the NSynth (Engel et al., 2017), a deep learning system for generating music notes from a wide range of acoustics and electronic instruments. The system builds upon WaveNet (van den Oord et al., 2016) with an AE architecture to learn a manifold of embeddings that allows for morphing and interpolation between generated instrument sounds. WaveNet is an AR model that functions by sequentially predicting each waveform sample (see Section 3.5.1). While capable of generating high-quality audio, sequential waveform generation comes at the expense of resource-intensive and potentially slow inference.

VAEs and GANs offer the advantage of parallelised training and generation, which can significantly accelerate the training process and allow for more efficient generation of new audio samples. In contrast to conventional AEs, which establish a fixed and deterministic encoding for each input, VAEs operate by learning a probabilistic latent space, where every point in the latent space corresponds to a probability distribution. This probabilistic nature enables smooth interpolation between data points, making VAEs particularly suitable for data generation tasks. Esling et al. (2018a) demonstrated the capabilities of VAEs in generating and continuously exploring instrument sounds through the navigation of a latent space regularised to match perceptual distances collected from timbre studies. On the other hand, GANs map complex data distributions to low-dimensional latent spaces through an adversarial training

[19] `https://mawf.io`
[20] `http://acids.ircam.fr/neurorack`
[21] `https://www.steinberg.net/vst-instruments/backbone`

strategy (Goodfellow et al., 2014). Donahue et al. (2019) introduced GANs for musical audio using a modified deep convolutional GAN (Radford et al., 2016) that operates on raw audio data. Alternatively, Engel et al. (2019) proposed GANSynth, which leveraged improvements in the training stability of GANs (Karras et al., 2017; Arjovsky et al., 2017; Gulrajani et al., 2017) to generate magnitude and instantaneous frequency spectrograms that are used to approximate the time-domain signal.

An emerging area within neural audio synthesis research concentrates on synthesising drum sounds. Aouameur et al. (2019) introduced a system for real-time generation of drum sounds using a DGM. The DGM consists of a conditional Wasserstein autoencoder (Tolstikhin et al., 2018) that learns a mapping between a latent space and a dataset of spectrograms extracted from a collection of labelled drum sounds. The model is coupled with a spectrogram inversion model consisting of a multi-head CNN that is trained separately to reconstruct the waveform associated with each spectrogram. Ramires et al. (2020a) presented a system for synthesising drum sounds with control over high-level timbral characteristics of the sounds. Their approach utilises a feedforward CNN architecture based on Wave U-Net (Stoller et al., 2018) that is conditioned on a set of timbral features. The features are derived from the Audio Commons collection of perceptual models (Pearce et al., 2017), which describe high-level timbral characteristics such as boominess, warmth, and brightness. These characteristics are based on a predefined set of audio features, informed by frequently used search terms on the Freesound website. The main limitation of this approach is that it learns a deterministic mapping between the input features and the corresponding waveform, thereby constraining the capacity to capture the variability within the data. Nistal et al. (2020) elaborated on this idea using a progressive growing Wasserstein GAN conditioned on the same set of timbral features. An auxiliary regression loss term is utilised in the discriminator during training, which allows for continuous control over the conditional features when generating audio. Lavault et al. (2022) proposed an alternative style-based generator for drum synthesis, which is conditioned using a differentiable implementation of the perceptual features (Pearce et al., 2017).

While employing GANs conditioned on timbral attributes allows for semantic control over the output, a complete representation of these features might not be achievable, especially when dealing with limited datasets. An alternative approach to explore is to derive synthesis controls from the latent space of the generator network through an unsupervised process. Ramires et al. (2022) evaluated various methods for obtaining synthesis parameters from the latent space of an adapted version of style-based GAN (Karras et al., 2020b) for drum synthesis. The user evaluation revealed that most participants found the synthesis parameters obtained from an unsupervised closed-form factorisation approach to be preferable, despite these parameters lacking clear semantic meaning in comparison to those derived from conditioning on timbral characteristics. Although some experienced users might desire more semantically meaningful synthesis parameters, the participants appreciated the opportunity to discover new timbres without requiring in-depth music production knowledge. Additionally, it was observed by the participants that some semantic parameters were redundant and not orthogonal to one another. Another approach to neural drum synthesis was proposed by Rouard and Hadjeres (2021), which builds upon developments in diffusion process modelling with stochastic differential equations (Song et al., 2021).

2.5 Chapter Summary

This chapter has provided a comprehensive exploration of the history, core concepts, and technologies associated with SBEM production, laying the groundwork for the research presented in the thesis. The origins and evolution of sampling were traced, emphasising the influence of technological advancements on the practice. Through an examination of the practices of sample sourcing, selection, manipulation, and arrangement, the common challenges and the diverse approaches adopted by producers when working with samples in contemporary SBEM production scenarios were identified. The chapter also discussed the unique characteristics of drums and the variety of techniques utilised by producers to integrate them into SBEM compositions. Furthermore, it presented a comprehensive review of music information retrieval and deep learning research, highlighting their potential to address the identified challenges associated with SBEM production. The growing adoption of deep learning methodologies in related MIR fields, combined with their demonstrated high performance, served as motivation to integrate deep learning-based systems in the approaches presented in subsequent chapters, with the aim of assisting in SBEM production.

Contemporary SBEM production entails exploring digital collections of audio, comprising both libraries of samples and recorded music. Selecting samples from these sources constitutes a meticulous retrieval task, during which music producers leverage their aesthetic judgment to listen and discern samples suitable for use in their compositions. Online sample libraries generally include tags for easy navigation; however, personal collections of samples may become unmanageable if individuals frequently sample from various recordings without appropriately labelling or organising them. A significant body of research has focused on automating music tagging, yet the specific domain of music sample tagging has not been extensively studied. Furthermore, while existing research on sample identification and retrieval has primarily focused on examining samples previously used by producers and the development of methods to facilitate the navigation of sample collections, an area that has received limited attention is the use of automated systems to assess existing music recordings and locate samples that meet particular criteria. Moreover, neural audio synthesis offers the potential to manipulate and create seamless transitions between existing elements within sample collections, thereby addressing the challenge of limited timbre diversity often encountered in such collections. This approach can also help reduce the creative barriers associated with complex synthesis techniques, which typically involve steep learning curves.

To this end, Chapter 4 presents a deep learning system for the automatic classification of samples based on their instrumentation roles (e.g., percussion, chords, melody). Through a series of evaluations, the system is shown to effectively label samples according to their functional roles, generate a high-level summary of SBEM arrangements, and identify *sample-able* material in existing recordings. Additionally, Chapter 5 introduces a neural audio synthesis system that facilitates continuous exploration and manipulation of sounds generated from a collection of drum samples. Functional control over the generation and transformation of drum samples is achieved using dimensionality reduction techniques to identify synthesis parameters from the underlying structure of the generator network. Furthermore, an encoder network is proposed to project input drum samples into this structure, which enables the re-synthesis of similar samples that can be modified using the learned synthesis controls. The forthcoming Chapter 3 will outline the fundamental deep learning techniques required to accomplish these solutions.

Chapter 3

Deep Learning Preliminaries

The previous chapter presented an overview of SBEM production practices and related technologies, highlighting the advantages and disadvantages of existing systems designed to assist in the creation process. This chapter will introduce the fundamental concepts of deep learning and provide an overview of the techniques used throughout the thesis. These preliminaries are essential to understand the concepts referred to throughout this thesis and the capabilities of the proposed systems.

The term deep learning refers to a family of machine learning methods within the field of artificial intelligence (AI), composed of multiple non-linear processing modules that can learn complex relationships from data without the need for explicit programming. A representation is learned from data using hierarchically structured layers that progressively extract more abstract characteristics, offering an advantage over traditional machine learning methods that require human-engineered features. The principal component of a deep learning model is an artificial neural network, which is a machine learning algorithm inspired by the biological neural network that constitutes the human brain. Artificial neural networks seek to exploit latent structures in a given input data distribution by learning feature hierarchies, where higher-level features are formed by the composition of lower-level features. Deep learning systems can be broadly classified into three data-driven optimisation approaches: 1) supervised methods, which use input data and corresponding labels during training, 2) unsupervised methods, which infer a function to describe latent structure from unlabeled data, and 3) semi-supervised learning, which involves training with only a subset of labelled input data. Additionally, deep learning algorithms can be distinguished as discriminative, where the goal is to identify decision boundaries between various classes in a dataset, or generative, which learn the probability distribution of a dataset and can generate new data instances.

Driven by widespread access to data and the necessary computing power, deep learning has enabled solutions to a diverse range of problems in fields such as computer vision (Voulodimos et al., 2018), autonomous driving (Grigorescu et al., 2020), medical diagnosis (Bakator and Radosav, 2018), and natural language processing (Otter et al., 2020). While deep learning has been extensively explored for data analysis and processing tasks, recent advancements in deep generative learning techniques have unlocked enormous potential for creative applications. For instance, neural style transfer enables the production of new images that combine the content of an arbitrary photograph with the appearance of well-known artworks (Gatys et al., 2016). StyleGAN (Karras et al., 2019) has been employed to generate realistic human faces that are indistinguishable from real photographs, as well as to edit and morph facial characteristics. DALL-E (Ramesh et al., 2021) uses text-to-image co-creativity to generate

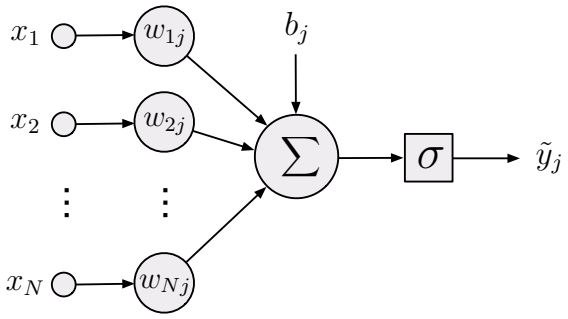

Figure 3.1: An illustration of a single neuron for inputs x, corresponding weights w, bias b, and activation function σ applied to the weighted some of the inputs.

aesthetically pleasing images from descriptions in natural language, providing a novel way of creating art with AI. Recent research has also begun to investigate ways in which deep learning systems can support humans in the creative process (Main et al., 2022; Louie et al., 2022).

Within the realm of music-related applications, deep learning techniques have been successfully applied to music information retrieval (MIR) tasks such as genre classification (Ndou et al., 2021), recommendation systems (Martín-Gutiérrez et al., 2020), automatic music transcription (Benetos et al., 2018; Wu et al., 2018), and music source separation (Hennequin et al., 2020). Deep learning can also be employed to generate new music content by analysing and extracting features from existing recordings, subsequently forming learned representations of musical elements. Applications of this technique include the development of powerful neural audio synthesisers (Engel et al., 2017; Esling et al., 2018a; Engel et al., 2019) and models capable of generating complete musical compositions (Carr and Zukowski, 2018; Dhariwal et al., 2020; Shen et al., 2022).

3.1 Mutli-layer Perceptrons

Multi-layer perceptrons (MLP) are the fundamental component of many deep learning systems. They are a supervised machine learning approach that can learn to approximate any continuous function $f(\cdot) : R^N \rightarrow R^o$, where N is the number of input features and o is the number outputs. An MLP is a feedforward neural network comprised of multiple densely connected layers of artificial neurons that define a mapping $y = f^*(x)$. Feedforward implies that information flows through the network in a forward direction, from input x to output y.

Figure 3.1 depicts a single neuron, which comprises a set of inputs, weights, bias and an activation function. A single neuron is a mathematical function that receives N inputs x_i, where i is the index for a given input. Each input x_i is multiplied by weighted value w_i, which affects the amount of influence an input will have upon the output. The output of a neuron is calculated by summing the weighted values of each input and applying an activation function σ as follows:

$$\tilde{y} = \sigma(b + \sum_{i=0}^{N} x_i \cdot w_i), \tag{3.1}$$

where $\tilde{y}$ represents the output activation, N is the number of inputs, and w_i is the weighted value associated the input x_i. The bias value b is a learnable parameter that is added to the weighted

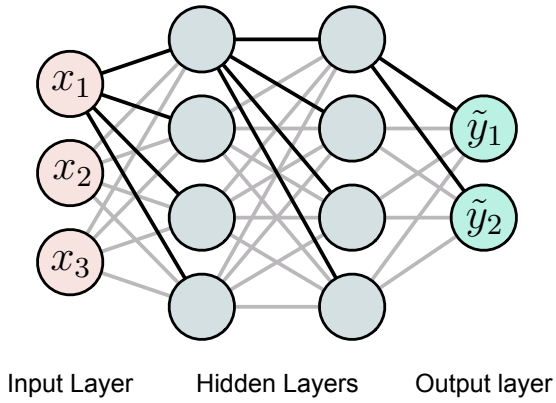

Figure 3.2: An illustration of a multi-layer perceptron combined of densely connected neurons within input, hidden and output layers.

sum of inputs before being passed through the activation function. The bias allows the neuron to adjust its output independently of the input, which enables the neuron to produce an output even if the input values are small or absent. An MLP consists of multiple neurons which are organised into layers, an input layer, one or more hidden layers, and an output layer as illustrated in Figure 3.2. The input layer is designed to receive training data and pass it into the network. The output layer is the final layer of a neural network which produces predictions given the information learned by the rest of the network. Hidden layers are the intermediate layers between the input and output which perform non-linear transformations to the input data.

The activation function σ is a differentiable, non-linear transformation that is applied to the weighted sum of the neuron inputs. Activation functions introduce non-linearity, which allow neural networks to learn and perform more complex problems. They are an important hyperparameter when designing neural network architectures and multiple non-linear functions have been proposed for various applications. One of the key considerations is to select a function that is differentiable, as training a neural network (see Section 3.3) involves calculating the gradient of the activation function. Additionally, the choice of activation function influences the capacity and performance of a neural network, and different activation functions are often used at different stages of a model to perform specific tasks. The hyperbolic tangent function (tanh) is frequently employed to normalise the output of a neural network into a range of $[-1, 1]$. This produces a zero-centered output, which is particularly advantageous for generating audio signals, as it allows the network to effectively capture and preserve the symmetry of audio signals. The tanh activation is calculated as:

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

The sigmoid function maps input values to the range of $[0, 1]$ using an exponential scale. Sigmoid activation functions are useful for binary classification tasks where a real number needs to be converted to a probability. The sigmoid function equation is:

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

The rectified linear unit function (ReLU) thresholds input values at 0, which returns positive values and turns negative values to 0. ReLU is much simpler and faster to compute compared to the sigmoid function, which requires exponentiation. The calculation for the ReLU activation is:

$$relu(v) = \begin{cases} 0, & \text{for } v < 0, \\ v, & \text{for } v \geqslant 0. \end{cases} \tag{3.4}$$

The gradient of the ReLU function is either 0 when $v < 0$ or 1 when $v \geqslant 0$, which allows the model to learn and update its parameters even when the input is large, whereas the sigmoid function has a saturating gradient when the input is large. By having a constant gradient of 1 for positive input values, ReLU also helps to avoid the vanishing gradient problem. Vanishing gradient refers to a phenomenon in which the gradients of a models weights become very small as they are propagated through multiple layers. This issue can impede the learning process as, during training, gradients are multiplied by the weights at each layer while being propagated through the network. If the weights are small, the gradients may diminish to an extent that they are insufficient for effectively updating the parameters of the model.

The leaky rectified linear unit (LReLU) is a modified version of the ReLU activation, where input values less than zero are multiplied by a fixed scalar. In the ReLU function, neurons that receive an input value less than 0 will output a constant value of 0 for all subsequent forward passes. LReLU adds a slope in the negative range to allow small negative values when the input is less than zero. By allowing a small non-zero gradient for negative input values, the model can continue to learn even when the input is negative. The equation for the LReLU activation is:

$$lrelu(v) = \begin{cases} \alpha v, & \text{for } v < 0, \\ v, & \text{for } v \geqslant 0, \end{cases} \tag{3.5}$$

in which, α is a hyper-parameter that determines the degree that negative values leak through the activation.

Another variation of the ReLU is the exponential linear unit (ELU) activation which introduces an exponential non-linearity on negative input values and is calculated as:

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

ReLU and its variants are commonly used as activation functions in the hidden layers of neural networks because they are computationally efficient and have a simple, non-linear form that is easy to optimise using gradient-based methods (see Section 3.3.2). Overall, the choice of activation function depends on the specific requirements of the model and the characteristics of the data. In practice, it is often necessary to experiment with different activation functions to find the one that works best for a particular use case.

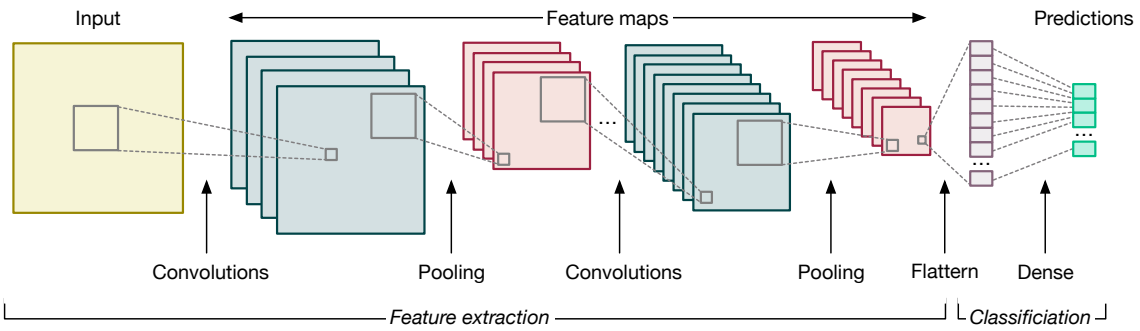

Figure 3.3: An illustration of a basic CNN architecture consisting of a stack of convolutional and pooling layers that systematically extract relevant features from the input. This is followed by a classification stage comprised of a densely connected layer and the predicted output neurons.

3.2 Convolutional Neural Networks

Convolutional neural networks (CNN) were inspired by the hierarchical receptive field model of the animal visual cortex (Fukushima, 1980) and have become one of the most widely used deep learning architectures, particularly in the field of computer vision. While the MLPs described in Section 3.1 are not designed to consider the ordering of inputs, CNNs take into account the spatial and temporal relationships present in structured data, such as the adjacency of pixels in images and the time-frequency relationship in audio spectrograms. Unlike MLPs, which have a connection between every neuron in each layer (i.e., fully connected), the neurons in CNNs use sparse connections, which reduces the number of parameters required for training. This makes them more efficient for processing higher-dimensional data such as images and spectrograms, which often require a large number of parameters. CNNs were originally developed to analyse two-dimensional data, but they can be adapted to process one-dimensional data, such as raw audio waveforms (Dieleman and Schrauwen, 2014). This modification enables the network to learn an end-to-end representation of one-dimensional sequences and has made CNNs a flexible tool for various audio applications, including speech recognition (Tzirakis et al., 2018), environmental sound classification (Abdoli et al., 2019), and automatic music tagging (Pons et al., 2018).

Figure 3.3 visualises a basic CNN architecture, which is formed by stacking convolution layers, pooling layers, and a densely connected layer. The network is typically divided into two stages: a feature extraction stage, where convolutional and pooling layers learn hierarchical representations of the input data, and a classification stage, where the output of the last pooling layer is flattened and fed to a densely connected layer that maps the features to the final output predictions. By stacking multiple convolution and pooling layers, the network can learn increasingly complex and abstract features. The earlier convolution layers use learned filters to extract low-level features (e.g., edges, corners) from the input data, while later convolution layers extract coarse features (e.g., shapes, objects).

3.2.1 Convolutional Layers

The convolutional layer is a fundamental building block of a CNN, providing the network with the ability to automatically learn spatial and temporal features from raw input data. Convolutional layers operate by systematically applying trainable filters to an input through convolution operations that summarise the presence of detected features. The input to a convolutional layer is usually a four-dimensional tensor

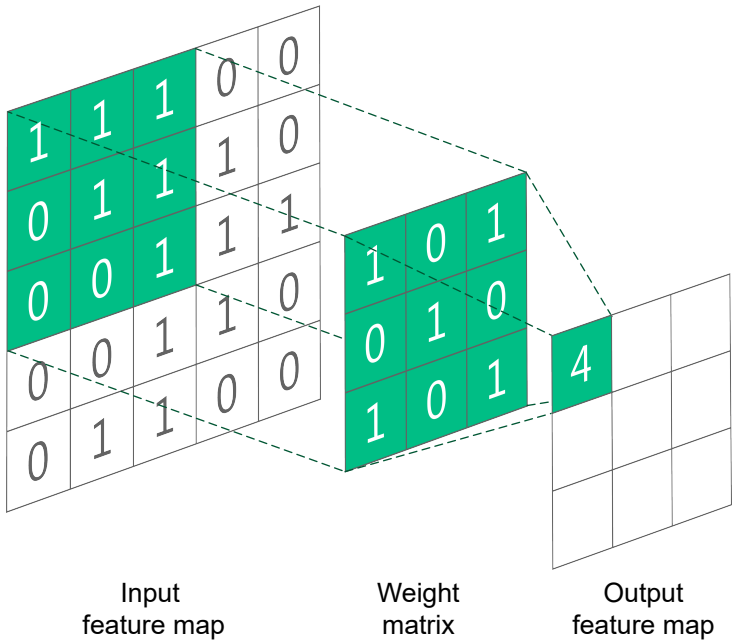

Figure 3.4: Illustration of the convolution operation in a convolutional layer. The input feature map (left) is convolved with the weight matrix (centre) to produce the output feature map (right). Green areas highlight the regions where matrix multiplication has been applied, resulting in the corresponding output value in the output feature map.

$t \in \mathbb{R}^{b \times w \times h \times c}$, with batch size b, width w, height h, and depth c. The batch size b is the number of input samples to be processed during a model update. Dimensions w and h correspond to the width and height of an input respectively. Channels c correspond to the depth, for example, separate channels for red, green and blue (RGB) colours in an image.

The filters—also referred to as kernels—are a small matrix (e.g., 3×3 or 5×5) of weights that are learned during training. Each filter is shifted along the spatial dimensionality of an input matrix and for the receptive field in every position, the sum of the element-wise multiplication (i.e., dot product) between the filter and input is calculated. Figure 3.4 illustrates the dot product between a filter and the receptive field of the an input. The output of the convolutional operation is referred to as a feature map H_j, which is the result of convolving an input matrix X_i with weight matrix W_{ij} and applying an activation function σ, where i and j denote the input and output dimensions respectively. A feature map can be calculated as follows:

$$H_j = \sigma(B_j + \sum_{i=0}^{l} X_i * W_{ij}), \tag{3.7}$$

where the convolution operation is denoted by $*$, the number of input channels are indicated by l and B_j is the bias. Each component of the feature map H_j is analogous to the output of a neuron. Therefore, neurons are only connected to a small local region in the input image, and the receptive field is equal to

the size of the filter. As the neurons are sparsely connected in this manner, fewer parameters need to be calculated, which can improve computational efficiency and training time. Another advantageous characteristic of convolutional layers is that they support a degree of translation invariance, in which the learned features can appear anywhere in an input, regardless of their spatial position.

The final output of the convolutional layer is the result of stacking the resulting feature maps of every filter along the channel dimension. The size of the output is determined by three hyperparameters: depth, which is the number of learnable filters; stride, the step size in which a filter is shifted across the spatial dimensions of an input; and padding, a process used to maintain the input shape by appending additional values to its border. *Same* padding is a technique that ensures that the size of the feature map remains unchanged during the convolution. Unless stated otherwise, *same* padding is used in all systems implemented in subsequent chapters.

3.2.2 Pooling Layers

Pooling layers are usually incorporated between two successive convolutional layers to progressively reduce the spatial resolution of the feature maps while maintaining the most important information. They have a comparable operation mode to convolutional layers but instead of learnable filters, operations such as max-, min-, and average-pooling are applied to summarise nearby neurons. For example, the max-pooling operation calculates the maximum value for a patch of neighbouring neurons in a feature map. The size of the pooling operation is usually a 2×2 matrix with a stride of 2, which reduces the size of each feature map by a factor of 2. By aggregating information from neighbouring neurons and limiting the dimensionality of the input data, pooling layers reduce the number of parameters in the network. This makes the network more efficient and helps to prevent overfitting (see Section 3.3.4). Pooling layers can also achieve partial invariance to local translation—for example, the precise location of an object in an image or the ability to recognise musical patterns in a way that is partially invariant to small timing differences.

3.3 Training Procedures

The weights and biases in the hidden layers of a neural network represent tunable parameters Θ which are updated to calculate a mapping between inputs and outputs through a process called *training*. An output $\hat{y}$ is calculated by parsing input data x through the network, from input layer to the output layer using forward path transfer function $\hat{y} = f_{nt}(\Theta, x)$. The training procedure involves finding an optimal set of network parameters that minimise the difference between the desired output y and the predicted output $\hat{y}$ using a loss function L that represents the error between the two as a single number.

3.3.1 Loss Functions

A loss function is a function for evaluating the performance of neural network whilst training on a dataset. Loss functions calculate the difference between target output y and approximated output $\tilde{y}$ as a single number that represents the overall loss L. There exist multiple different loss functions, each better suited for particular tasks and models. Mean square error (MSE) measures the average squared difference between observed and predicted values and is often used in regression tasks where the goal

is to predict a continuous output variable based on one or more input variables. MSE is calculated as follows:

$$L_{\text{MSE}} = \frac{1}{N}\sum_{i=n}^{N}(y_n - \tilde{y_n})^2, \tag{3.8}$$

where N represents the dimensionality of the output layer and y_n is the n^{th} element of output vector y.

For classification problems with multiple classes, the categorical cross-entropy (CE) loss can be used. CE is calculated as:

$$L_{\text{CE}} = -\sum_{n=1}^{N} y_n \log(\tilde{y}). \tag{3.9}$$

For binary classification tasks a sigmoid activation can be used in the final layer with the binary cross-entropy (BCE) loss. BCE can be calculated as follows:

$$L_{\text{BCE}} = -\frac{1}{N}\sum_{n=1}^{N} y_n \cdot \log(\, p(\tilde{y}_n)) + (1 - y_n) \cdot \log\,(1 - p(\tilde{y}_n)). \tag{3.10}$$

3.3.2 Parameter Optimisation

Finding optimal network parameters for a given training dataset can be achieved using an optimisation algorithm to iteratively search through a space of possible parameter values. Parameter optimisation is the process of incrementally updating the network parameters Θ ($\Theta = [W, B]$) to minimise (or maximise) the loss L and reach optimal prediction accuracy. The layered structure of a neural network enables weight updates through backpropagation, a method for computing the gradient of a loss function with respect to all the trainable parameters (Rumelhart et al., 1986). Backpropagation calculates a gradient of a loss function L with respect to all network weights (elements in Θ):

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

The gradient ∇_{Θ} is a vector that represents the rate of change of the loss function L with respect to each element of Θ. By recursively applying the chain rule in the backward direction of the network, starting from the output layer and moving towards the input layer, ∇_{Θ} is computed by taking the partial derivatives of L with respect to each parameter and assembling them into a single vector. These partial derivatives indicate the direction in which to update the model parameters Θ to minimise the loss function. If the derivative for every activation function in the network is known, automatic differentiation can be used to calculate the gradients $\mathcal{G}$ for each of the trainable parameters. The gradients are then used to incrementally update the model parameters using an optimisation algorithm such as gradient descent, which reduces the error by iteratively adjusting the weights in the direction of steepest descent. Each incremental update, termed an iteration, is calculated by subtracting the gradient of the loss function with respect to the parameters multiplied by a learning rate:

$$\Theta \leftarrow \Theta - \alpha \cdot \mathcal{G}, \tag{3.12}$$

where α is the learning rate, a hyperparameter that determines the step size for each update. The dataset for training a neural network consists of multiple pairs of corresponding input and output data. One complete cycle through all observations in the training dataset is referred to as an epoch.

There are three different strategies for updating the model parameters using the training examples: batch gradient descent, which uses gradients computed over the entire training set; stochastic gradient descent, where each update is calculated for individual training examples; and mini-batch gradient descent, which uses a small subset (i.e., a mini-batch) of the training data per parameter update. Stochastic gradient descent offers faster convergence and requires less memory compared to batch gradient descent, but can be sensitive to noise in the training data and may converge to a suboptimal solution. Mini-batch gradient descent offers a trade-off between the advantages and disadvantages of batch and stochastic gradient descent. It enables efficient use of computational resources, while also being less sensitive to noise, thus providing a smoother convergence trajectory due to the averaging of gradients over each batch. Graphics processing units (GPU) are particularly well-suited for batch processing as they can efficiently process large amounts of data in parallel. By dividing the data into batches, the GPU can process each batch in parallel, reducing the overall processing time. The memory size of the GPU is an important consideration when training deep learning models, as it can limit the batch size, the number of processes that can be done simultaneously, and the size of the model that can be loaded into memory.

The chief goal of an optimisation algorithm is to locate the set of parameters that produces the lowest possible error on the training data with regard to the loss function L (i.e., global minima). However, the standard gradient descent is prone to converging at a suboptimal solution that produces poor performance (i.e., local minima). The learning rate α controls the degree of a model update and subsequently how quickly the model converges to an optimal solution. If the value for α is set too small, then gradient updates cannot escape the local minima, whereas a value too large can result in drastic updates and divergent behaviour. There are several techniques to address the problem of local minima in optimisation. For example, stochastic gradient descent (SGD) with momentum (Sutton, 1986) uses short-term memory of previous gradients to increase the learning rate and accelerate convergence by adding momentum at each iteration, thus helping to overcome local minima. Other examples of gradient descent variations include: adaptive gradient algorithm (Duchi et al., 2011), root mean square propagation (Tieleman et al., 2012), and adaptive moment estimation (Adam) (Kingma and Ba, 2015). Adam is a popular optimisation algorithm that combines the benefits of several optimisation algorithms and is used for training the systems implemented throughout this thesis. In Adam, the model parameters are updated using a combination of the gradient and an exponentially decaying average of past gradients m_t (i.e., the first moment) and square gradients v_t (i.e., the second moment):

$$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t, \tag{3.13}$$

$$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2, \tag{3.14}$$

where g_t is the current gradient, and β_1 and β_2 control the decay rates for the moving averages m_t and v_t respectively. The algorithm also incorporates a bias correction term to adjust for the fact that the moving averages are initialised at zero, leading to moment estimates that are biased towards zero during the early stages of the optimisation. The bias-corrected estimates $\hat{m}_t$ and $\hat{v}_t$ are calculated as follows:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \tag{3.15}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \tag{3.16}$$

The parameter update is then calculated using:

$$\Theta \leftarrow \Theta - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \tag{3.17}$$

where ϵ is a small constant added to prevent division by zero (e.g., $\epsilon = 10^{-8}$), ensuring numerical stability when computing the adaptive learning rates for each parameter in the model. One of the main advantages of Adam is that it can adapt the learning rates of each parameter individually, rather than using a fixed learning rate for all parameters. This makes the optimisation process more efficient as larger updates can be performed on sparse parameters and smaller updates can be performed on abundant parameters. The adaptive learning rate can help to overcome some of the limitations of fixed learning rates in gradient descent, such as slow convergence or diverging from the optimal solution.

3.3.3 Initialisation

Initialisation refers to the process of defining the initial values for the model parameters Θ prior to training. Selecting an appropriate method for parameter initialisation is a crucial step that ultimately affects the likelihood of the global minima being reached. In general, the parameters in a neural network are randomly initialised using some form of probability distribution to ensure that each neuron learns different features that have varying influences on the loss. For instance, the uniform initialisation method entails random initialisation of network parameters from a uniform distribution between two values, often within the range of -1 and 1. On the other hand, the Xavier initialisation (Glorot and Bengio, 2010) method randomly initialises the parameters using a normal distribution with a mean of 0 and a standard deviation that is proportional to the square root of the number of neurons. Similarly, the He initialisation (He et al., 2015) method randomly initialises the parameters using a normal distribution with a mean of 0 and a standard deviation that is proportional to the square root of the number of neurons divided by 2. Differences in convergence rate and final performance of a neural network can be attributed to varying initialisation methods. As such, the choice of initialisation techniques should be tailored to the specific neural network architecture and problem context.

3.3.4 Regularisation

Regularisation refers to a set of techniques which are used to improve the capacity of a model to generalise to new, unseen data. Generalisation allows the information learnt by a deep learning algorithm

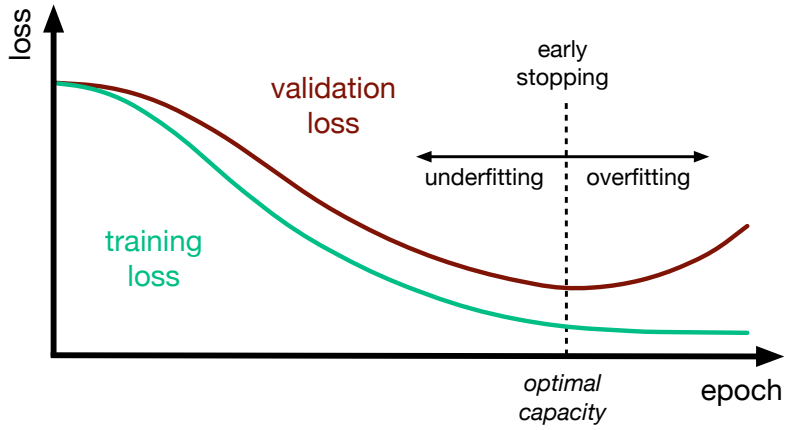

Figure 3.5: Training (light green) and validation (dark red) loss curves, where the vertical dotted line represents when early stopping occurs (i.e., the optimal model capacity).

to be applied in other contexts. Overfitting occurs when a model accurately represents the training data but fails to generalise to new data. The data used for training a model is usually a small sample of the real data associated with a particular problem. Consequently, model parameters may be adapted to fit the training dataset too closely. This can result in poor performance on test data and inaccurate predictions for new, unseen data. Conversely, underfitting occurs when a system does not accurately represent the training data or test data. Training a network that generalises well for a given objective is typically achieved by combining different regularisation methods. Presented below are a variety of regularisation techniques that can be employed to achieve optimal model capacity, where the model is neither too simple to capture relevant patterns in the data nor too complex, and thus prone to overfitting or poor performance on new data.

Early Stopping

Early stopping is a regularisation technique that seeks to stop training before the model starts to overfit to the training data. Training a model for too many epochs will result in overfitting, whereas training for an insufficient number of epochs will result in underfitting. This strategy aims to identify the epoch in which optimal model capacity can be achieved by splitting training data into three separate sub-sets: training data, used to update the gradients and fit the parameters of the model; validation data, used to provide an unbiased evaluation of model performance during training; and test data, used to assess the performance of the final model. During training, a validation loss is calculated to monitor the effect of training on the isolated validation data. Figure 3.5 illustrates the training and validation loss for a model trained without early stopping. The point in which the validation loss begins to increase is good indication that the model has begun to overfit and training can be stopped at this epoch (illustrated by the vertical dotted line in Figure 3.5).

$\mathcal{L}_1$ and $\mathcal{L}_2$ norm regularisation

Adding a penalty term to the loss function L limits the modelling capabilities of a neural network and can help to counteract overfitting. $\mathcal{L}_1$ norm regularisation involves adding a penalty term that is proportional to the sum of the absolute values of the weights:

$$\mathcal{L}_1 = L + \lambda \cdot \sum |w|, \tag{3.18}$$

where λ is a tunable hyperparameter that controls the strength of the regularisation. This encourages the model to have sparse weight values, as many of the weights may become 0. On the other hand, $\mathcal{L}_2$ norm regularisation penalises the squared magnitude of the weights:

$$\mathcal{L}_2 = L + \lambda \cdot \sum w^2. \tag{3.19}$$

This encourages the model to have small but non-zero weight values. Both $\mathcal{L}_1$ and $\mathcal{L}_2$ norm regularisation can be used alone or in combination to control the complexity of a model and counteract overfitting. The choice between regularisation methods is often dependent on the problem context and the characteristics of the data used in training.

Batch Normalisation

Batch normalisation (Ioffe and Szegedy, 2015) is a method that accelerates training and improves the performance of a neural network by normalising the activations between each layer to have a mean of 0 and a variance of 1 across a mini-batch of input data. As described in Section 3.3.2, training a neural network requires calculating the gradient of the loss function with respect to the model parameters. During training, the distribution of the neuron activations in a layer can change, which can make it difficult for the gradients to flow through the network and update the parameters effectively. Batch normalisation ensures that the neuron activations are roughly in the same range, making it easier for the gradients to flow through the network and update the parameters effectively. Batch normalisation is implemented by first calculating the mean $\mu_{\mathcal{F}}$ and variance $\sigma^2_{\mathcal{F}}$ of features $\mathcal{F} = \{x_1 \dots x_m\}$ across a mini-batch with m examples as follows:

$$\mu_{\mathcal{F}} = \frac{1}{m} \sum_{i=1}^{m} x_i, \tag{3.20}$$

$$\sigma^2_{\mathcal{F}} = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_{\mathcal{F}})^2. \tag{3.21}$$

For each feature x_i, the mean is subtracted and then divided by the variance before scaling and shifting with two learnable parameters $\lambda_{\mathcal{F}}$ and $\beta_{\mathcal{F}}$ as follows:

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{F}}}{\sqrt{\sigma^2_{\mathcal{F}} - \epsilon}}, \tag{3.22}$$

$$y^i = \lambda_F \hat{x}_i + \beta_{\mathcal{F}}, \tag{3.23}$$

where ϵ is a small constant added to avoid dividing by 0 and y_i is the batch normalised output. Batch normalisation ensures that the distribution of the activations stays relatively stable whilst training the network and can dramatically reduce the required number of training epochs.

Dropout

Dropout (Srivastava et al., 2014) is a simple regularisation technique that helps avoid overfitting by randomly disabling a fraction of the connections between the neurons in two adjacent layers of a neural

network during training. This means that the input to each neuron can come from a randomly selected subset of the neurons in the previous layer for each training example, which prevents the network from relying too heavily on any one neuron or feature. For each training iteration, neurons are randomly disabled with a frequency of rate r, which is the likelihood that a given neuron will be temporarily removed from the network. Neurons that are retained during training are scaled up by $\frac{1}{1-r}$ such that the sum over all inputs is unchanged. As a result, a thinner network is created, with the architecture slightly varying between iteration. Probabilistically disabling connections can be used to simulate having a large number of different network architectures without the expense of additional computational resources required for training and multiple models. Disabled neurons are re-enabled during inference, which can be interpreted as using an ensemble of the multiple thinner networks to make a prediction. This process prevents neurons from co-adapting during training and encourages the network to learn a more efficient representation (i.e., fewer redundant or irrelevant features) of the input data.

Data Augmentation

Data augmentation is a method often used to prevent overfitting when there is insufficient training data available. The approach involves applying various transformations to an existing dataset using domain-specific knowledge to artificially create additional examples. This method can be particularly useful for addressing imbalanced datasets. The type of transformations applied will depend on the specific application; for instance, when working with images, transformations such as rotation, re-scaling and cropping can be used, while in the context of audio signals, techniques such as manipulating the pitch, amplitude envelope, or duration can be used. It is crucial to design a careful augmentation strategy that considers the given task to ensure that the approach does not negatively impact performance or stray too far from the original objective. Ultimately, the target application and the properties of the trained model will guide the choice of augmentation techniques to use.

3.4 Audio Representations

For a neural network to process an audio signal, it requires a digital representation of that signal. Neural networks can benefit from specific representations of audio data depending on the task and network architecture. Extracting musical information from raw audio signals can be challenging due to their complexity and variability. Time-frequency representations offer a more discernible perspective of the audio signal, simplifying analysis and processing. This section presents typical audio representations, as depicted in Figure 3.6, which are often used as input for training deep learning models in various analysis and generation tasks.

3.4.1 Raw Audio Waveform

The lowest level audio representation used in deep learning is the sampled raw audio waveform, for example, a mono WAV file sampled at 44.1 kHz with 16-bit resolution. This uncompressed audio representation, illustrated in Figure 3.6, consists of a series of waveform samples $x = [x_1, ..., x_n]$, where each sample x_n depicts the amplitude value of the signal at that particular time step. Training with raw audio data allows a neural network to build an internal representation without any prior assumptions. However, time-domain representations can be computationally expensive and may not efficiently capture frequency information.

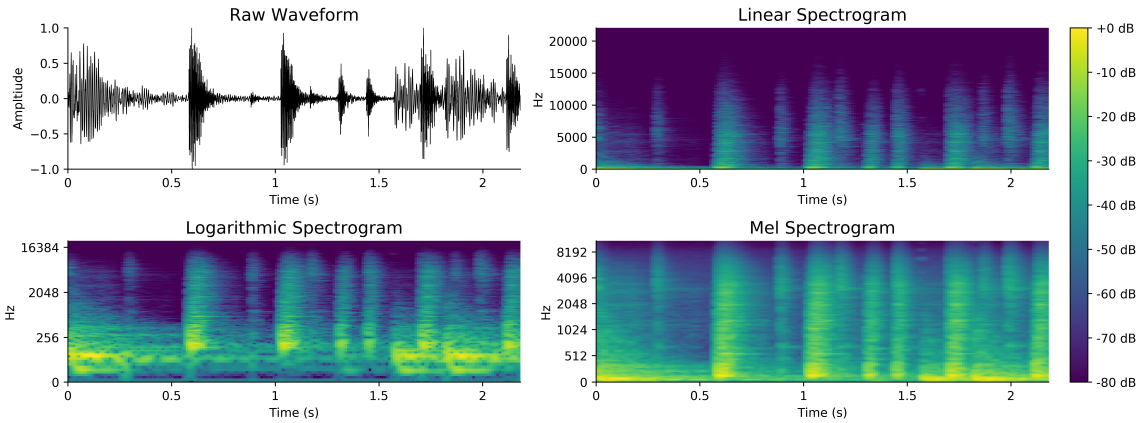

Figure 3.6: Visualising a drum loop using various audio representations: raw audio waveform (top left), linear spectrogram (top right), logarithmic spectrogram (bottom left), and Mel spectrogram (bottom right). Each representation highlights different aspects of the audio signal, offering unique levels of detail and emphasis.

3.4.2 Short-Time Fourier Transform

The short-time Fourier transform (STFT) is a widely used technique to transform audio data into the time-frequency domain that captures the frequency content of the signal over time. To compute the STFT, an audio file of N samples is divided into T frames, each containing p samples. Typically, a Hann window ω is applied to these frames to smooth out boundary discontinuities, using a hop size δ of $\frac{p}{4}$ samples. This process results in framed audio features A with a dimensionality of $p \times T$, where the size of each frame influences a balance between temporal and spectral precision. A frequency representation X of each frame A is then calculated using the discrete Fourier transform. This leads to a $\frac{p}{2} \times T$ representation, with $\frac{p}{2}$ representing the number of frequency bins. The STFT can be represented by the following equation:

$$X(k,t) = \sum_{n=0}^{N-1} x(n)w(n - t\delta)e^{\frac{-2\pi jkn}{N}}, \tag{3.24}$$

where k denotes the frequency bin index, t denotes the frame index, and j represents the imaginary number unit.

3.4.3 Linear Spectrogram

The linear spectrogram can be obtained by squaring the magnitudes of the STFT X, as shown in the following equation:

$$S = |X|^2. \tag{3.25}$$

In Figure 3.6, a linear spectrogram is displayed with the vertical axis representing frequency and the horiontal axis representing time. Magnitude values in the spectrogram are mapped to a decibel scale, where the gradient from yellow to blue indicates the signal amplitude in decibels at the corresponding time and frequency.

3.4.4 Logarthimic Spectrogram

The large number of frequency bins ($\frac{p}{2}$) in linear spectrograms can result in a computationally expensive training process. To reduce complexity and better align with human perception of loudness, the linear frequency bins in Hertz (Hz) can be converted to a logarithmic scale using b bands per octave. The logarithmic spectrogram is calculated by applying a filter bank FB through matrix multiplication in the frequency domain. The filters of the logarithmic filter bank FB can be derived using:

$$FB_{k,b} = \begin{cases} 0, & k < g(b-1) \\ \frac{k-g(b-1)}{g(b)-g(b-1)}, & g(b-1) \leqslant k \leqslant g(b) \\ \frac{g(b+1)-k}{g(b+1)-g(b)}, & g(b) \leqslant k \leqslant g(b+1) \\ 0, & k > g(b+1), \end{cases} \tag{3.26}$$

in which $g()$ represents the boundaries of logarithmically-spaced frequency bins, with $b \in B$ as an index that identifies the specific frequency bin to which a given frequency belongs.

3.4.5 Mel Spectrogram

Mel spectrograms are an audio representation that takes into account the human auditory system (Moore, 2012). They are created by converting frequency values from cycles per second (Hz) to the Mel scale, which is a perceptually uniform scale of pitches. The Mel scale reduces the resolution of higher frequencies, while retaining the most relevant information, which helps to reduce the overall size of the representation. A linear spectrogram S is transformed into a Mel spectrogram M using a Mel filter bank, consisting of η Mel bands (typically $\eta = 128$). The frequency f in Hz is converted to the Mel scale m using:

$$m = 2595 \log_{10}(1 + \frac{f}{700}). \tag{3.27}$$

The Mel filter banks F are created using Equation 3.26; however, $g()$ is replaced by the Mel-spaced frequencies derived from Equation 3.27 and $B = \eta$. The resulting Mel filter bank is then used to transform a linear spectrogram S into a Mel spectrogram M through matrix multiplication. Figure 3.6 provides a visual representation of the resulting Mel spectrogram.

3.5 Deep Generative Models

Deep generative models (DGM) are a family of deep neural networks that learn complex functional mappings to approximate high-dimensional probability distributions from training data. The aim is to model the true but unknown data distribution of a training dataset for the purpose of generating new data samples with similar statistical properties. DGMs are provided with a finite set of N training examples $\mathcal{X} = \{x_1, ..., x_N\}$ drawn independently from the true data distribution function $p_d(\mathcal{X})$ and learn a model $p_\theta(\mathcal{X})$ that estimates this distribution. During training, the most plausible network parameters θ^* are

learnt to minimise some notion of distance between $p_d(\mathcal{X})$ and $p_\theta(\mathcal{X})$. This optimisation task can be formulated as follows:

$$\theta^* = \min_\theta \mathcal{D}(p_d, p_\theta), \tag{3.28}$$

where θ are the network parameters and $\mathcal{D}$ is a measure of divergence between the data distribution p_d and the model distribution p_θ.

Some of the differences between the different types of DGMs can be eliminated by focusing on the versions of models that learn via the principle of maximum likelihood estimation (MLE). The principle of MLE is to learn the network parameters θ which maximise the probability of obtaining the training samples $x \in \mathcal{X}$ given the model $p_\theta(x)$. For computational simplicity and numerical stability, it is more convenient to do this in log space (i.e., a sum rather than a product over examples). This process of MLE for parameters θ can be formulated as follows:

$$\theta_{MLE} = \max_\theta \sum_{i=1}^{N} \log p_\theta(x_i), \tag{3.29}$$

where N is the number of data observations. In practice, only a subset of data observations are available from the true data distribution, which is defined as the empirical distribution $\hat{p}_d$. The MLE can also be interpreted as minimising the degree of dissimilarity between the empirical distribution $\hat{p}_d(x)$ and the model distribution $p_\theta(x)$ measured by the Kullback–Leibler (KL) divergence (Goodfellow et al., 2016). The KL divergence can be calculated using:

$$\mathcal{D}_{KL}(\hat{p}_d \parallel p_\theta) = \mathbb{E}_{x\sim\hat{p}_d}[\log \hat{p}_d(x) - \log p_\theta(x)], \tag{3.30}$$

where the $\parallel$ operator indicates divergence between the two distributions and $\mathbb{E}_{x\sim\hat{p}_d}$ is an expectation with respect to the empirical distribution $\hat{p}_d$. The parameters of a generative model can be trained to minimise the KL divergence using:

$$\theta^* = \min_\theta \mathbb{E}_{x\sim\hat{p}_d}[-\log p_\theta(x)]. \tag{3.31}$$

Once trained, generative models can generate new data samples $\hat{x} \sim p_\theta(x)$ that follow the same probabilistic distribution of the given training dataset. To gain control over the samples drawn from the generative model, the maximum likelihood estimation can be generalised to estimate a conditional probability $p_\theta(x|c)$ in order to predict x given a conditioning variable c. If X represents all of the inputs and C all the observed conditions, then the conditional MLE can be formulated as:

$$\theta_{MLE} = \max_\theta \mathcal{D}(p_d, p_\theta(X|C)). \tag{3.32}$$

The conditional information can be provided as inputs to a generative model that enables targetted control over data generation. For the sake of clarity, the theoretical descriptions that follow this section will assume unconditional modelling.

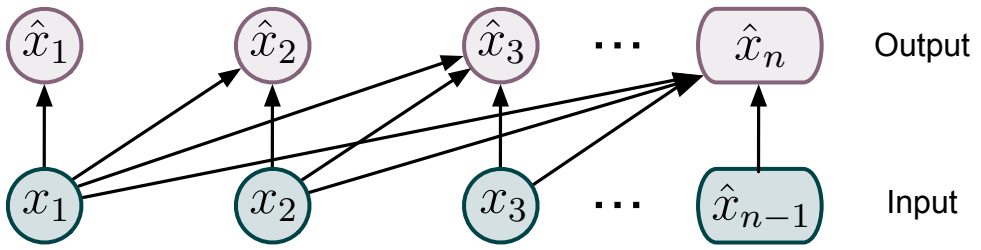

Figure 3.7: A basic autoregressive neural network architecture, illustrating the input neurons feeding into the network and the output neurons responsible for generating predictions based on the learned temporal dependencies.

DGMs address the core task of density estimation, which is to learn a model $p_\theta(x)$ that approximates the underlying structure of the training data $p_d(x)$ (i.e., the probability density). Different approaches to deep generative modelling can be categorised by comparing the strategies employed to compute or approximate the likelihood of the training data given the model parameters and their corresponding gradients (Goodfellow, 2016). The probability density can be estimated explicitly by defining a density function $p_\theta(x)$ and learning the parameters that maximise the likelihood of this function with respect to the training data. For example, autoregressive (AR) networks (Section 3.5.1) learn through a carefully designed parametric function that is able to capture the complexity of the data distribution and remain computationally tractable. In contrast, variational autoencoders (VAE) (Section 3.5.2) involve a density function with an intractable distribution of latent variables responsible for generating the observed data. This complexity makes direct optimisation challenging, so approximations to the likelihood, such as maximising a lower-bound, are used for effective model training. Generative adversarial networks (GAN) (Section 3.5.3), on the other hand, learn a model capable of sampling from $p_\theta(x)$ without explicitly defining it, enabling the generation of new data resembling the original dataset by sampling from a simple probability distribution (e.g., Gaussian).

3.5.1 Autoregressive Models

Autoregressive (AR) models explicitly capture the relationships between various points in a sequence of data, allowing them to model the underlying structure and dependencies. This is achieved by defining an explicit density function that is computationally tractable. AR models are based on the concept of autoregression, which is a method used to model the current output of a system as a function of its previous outputs. This is particularly useful for time series data, as future time steps can be predicted by evaluating the values of all preceding time steps (Bengio et al., 2001; Uria et al., 2014b). AR networks are specifically designed to model sequential data (See Figure. 3.7), where the input is an n-dimensional sequence $x = x_1, \ldots, x_n$. The joint probability of an input sequence x is decomposed into a product of conditional probabilities using the chain rule as follows:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, \ldots, x_{i-1}). \tag{3.33}$$

The conditional distributions over each x_i can be modelled using a deep neural network, which predicts the i^{th} variable x_i by multiplying the probabilities of all of the preceding $i-1$ variables. After defining

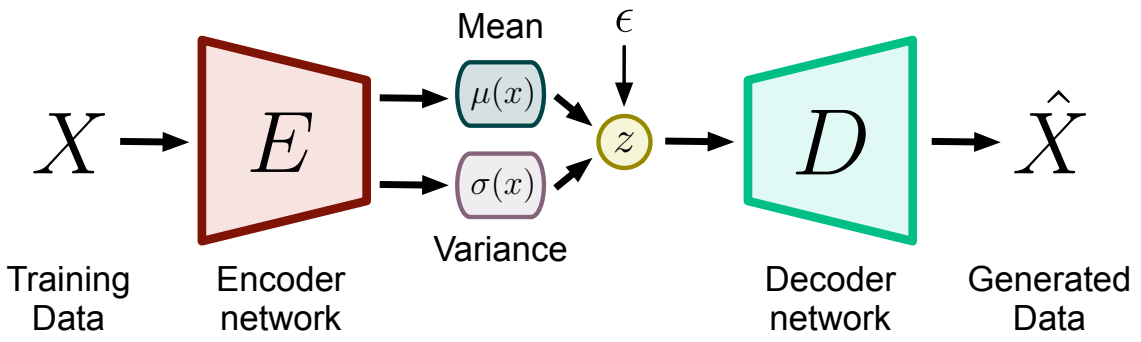

Figure 3.8: Illustration of a variational autoencoder (VAE). Input data samples from the training data X are fed into the encoder, which maps each sample x to the parameters of a normal distribution, defined by mean $\mu(x)$ and variance $\sigma(x)$ in the latent space. The decoder receives random samples from the latent space z as input and generates an approximation of the original data, denoted as $\hat{X}$.

this explicit density function, the parameters of the neural network θ are optimised to maximise the likelihood of training data. The conditional nature of AR models allows them to be trained using a supervised approach, which facilitates the evaluation of their predictive accuracy.

This approach is well-suited for raw audio signals, where each element x_i in the sequence corresponds to the amplitude value of the waveform at a discrete time step (i.e., sample). However, the high temporal resolution of audio data (i.e., many samples per second) can require a complex model with a large number of parameters to accurately capture the temporal patterns in the data. For example, a 1-second audio signal at CD quality has a sampling rate of 44100 Hz and a bit depth of 16 bits, which allows for a total of 65536^{44100} possible sequences. Modelling such high-dimensional data can result in slow training and inference for audio generation. As audio can exhibit patterns at multiple time scales, several techniques have been proposed to improve the receptive field and memory capacity of AR models. This has been achieved through hierarchical structured recurrent neural networks that can model long-term dependencies (Mehri et al., 2017) or through the use of CNNs with causal dilated convolutions (van den Oord et al., 2016), which increases the receptive field of convolutional filters without increasing the number of parameters.

3.5.2 Variational Autoencoders

A variational autoencoder (VAE) (Kingma and Welling, 2014) is a generative model that uses a probabilistic framework to learn a compact representation of the input data in a continuous, lower-dimensional latent (i.e., hidden) space. The latent space can be sampled to generate novel data with statistical properties similar to the training data. VAEs use approximations of the true density function through variational inference to maximise the likelihood of the training data. Variational inference approximates the posterior distribution of the latent variables given the observed data using simpler, more tractable distributions. To achieve this approximation, the parameters of the approximating distribution are iteratively adjusted to enhance its resemblance to the true distribution. By using this technique, the model is more efficient and scalable when handling complex data (e.g., images and audio), which can help mitigate some of the challenges associated with the design requirements of models with tractable density functions.

It can be assumed that the underlying structure in a complex data distribution x can be parameterised by a set of latent (i.e., hidden) variables z. VAEs are designed to learn a latent representation of x using two connected neural networks: an encoder and a decoder network (illustrated in Figure 3.8). The encoder network $q_\phi(z|x)$ learns parameters ϕ that approximate the unknown posterior distribution $p(z|x)$ of the latent variables z given the observed data x. In a standard autoencoder the encoder output is a single latent vector; whereas in a VAE, the encoder maps x to the parameters of a normal distribution, which is defined by a mean $\mu(x)$ and variance $\sigma(x)$ in latet space z. Learning the mean and variance of a normal distribution allows the VAE to capture the uncertainty in the data and generate new samples that are similar to the original data. This is often modelled as an isotropic Gaussian distribution, in which case the encoder can be defined as:

$$q_\phi(z|x) = \mathcal{N}(z|\mu_\phi(x), \sigma_\phi(x)). \tag{3.34}$$

The decoder $p_\theta(x|z)$, with parameters θ, receives random samples $\tilde{z} \sim q_\phi(z|x)$ as input and generates an approximation of the original data $\tilde{x}$. To facilitate this stochastic sampling process during the optimisation of parameters ϕ and θ, Kingma and Welling (2014) propose to reparameterise the latent variable $\tilde{z}$ using a differentiable transformation $\tilde{z} = g_\phi(\epsilon, x)$, in which ϵ is a random variable sampled from a distribution $p(\epsilon)$ that is independent of x or ϕ. After reparameterisation, $\hat{z}$ can be expressed as:

$$\hat{z} = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \tag{3.35}$$

where $\odot$ is the element-wise product and ϵ is sampled from $\mathcal{N}(0, I)$. The encoder and decoder networks can then be trained simultaneously to maximise the evidence lower-bound (ELBO) (Jordan et al., 1999) of the data log-likelihood $p_\theta(x)$ as follows:

$$\begin{aligned} \log p_\theta(x) &\geqslant -\mathcal{D}_{KL}(q_\phi(z|x) \parallel p_\theta(z)) + \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] \\ &\equiv \mathcal{L}(\phi, \theta, x), \end{aligned} \tag{3.36}$$

where the ELBO is denoted as $\mathcal{L}(\theta, \phi, x)$. The first term in Equation (3.36) is the KL divergence between the approximate posterior and the prior distribution, $\mathrm{KL}(q_\phi(z|x)|p_\theta(z))$. The prior over the latent variables z is typically a normal distribution $p(z) = \mathcal{N}(0, I)$. Minimising this term encourages the encoder to produce distributions that are close to prior, which helps to regularise the organisation of the latent space. The second term in (3.36) maximises the expected log-likelihood of the data x under the approximate posterior distribution $q_\phi(z|x)$. This is a measure of error between the input x and the reconstructed $\tilde{x}$ and is typically calculated using the MSE or BCE (Section 3.3.1).

VAEs are an efficient representation learning framework for generative modelling, with fast training, generation and encoding time; however, due to the variational approximation (i.e., the discrepancy between the ELBO and the exact likelihood function), they often produce lower-quality, blurred samples compared to other generative modelling approaches (Huang et al., 2018). Adversarial autoencoders (Makhzani et al., 2015) have been proposed to address the limitations of VAEs by introducing an adversarial loss term that encourages the generated samples to be indistinguishable from the true data distribution. This loss term is combined with the reconstruction loss from the autoencoder to create a hybrid loss function that enables the network to learn a more accurate and representative latent space.

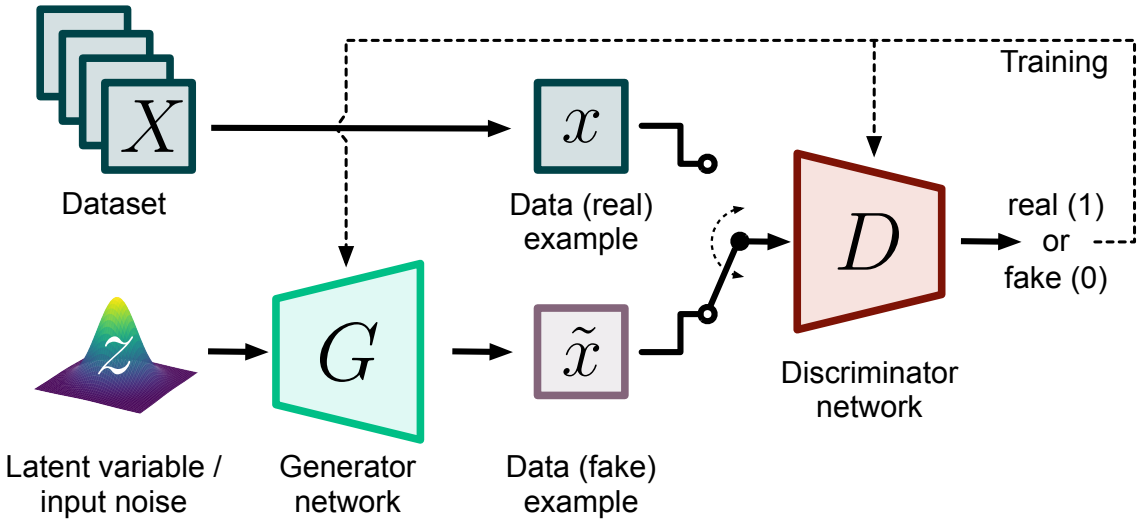

Figure 3.9: Diagram of a Generative Adversarial Network (GAN). The generator network G receives random samples from a latent space Z and generates synthetic data, while the discriminator network D attempts to distinguish between real samples x from the training data and fake samples $\hat{x}$ from the generator.

3.5.3 Generative Adversarial Networks

A generative adversarial network (GAN) (Goodfellow et al., 2014) is a deep learning framework that can learn to produce realistic samples of a given dataset from low-dimensional latent vectors. This is achieved by training two competing (i.e., adversarial) neural networks: a generative model (i.e., generator) and a discriminative model (i.e., discriminator). This type of learning framework does not rely on the specification of output distributions, avoiding the issue of approximating many intractable probabilistic computations. Alternatively, the generator implicitly defines a probability density function by interacting indirectly with the true data distribution $p_d(x)$. The generator represents a complex transformation from a simple distribution $p_d(z)$ to $p_d(x)$, where the architecture provides the family of possible distributions to sample from and the parameters θ select a distribution from within that family (Goodfellow et al., 2016).

The original GAN framework as proposed by Goodfellow et al. (2014) defines an adversarial game-theoretic scenario between the generator network and the discriminator network as illustrated in Figure 3.9. The generator $G : \mathbb{R}^m \rightarrow \mathbb{R}^n$ is used to learn a mapping between the training data space X and the latent space Z, where $Z = \mathbb{R}^{d_z}$. The dimensionality of the latent space d_z depends on the application but it typically has a much lower dimensionality than the generated data. Latent variables $z \in Z$ are sampled from a known prior $p(z)$, which is modelled with a simple distribution (e.g., Gaussian, Uniform). A new example $\hat{x}$ can be generated by providing G with a latent variable $z \sim p(z)$ such that $\hat{x} = G(z; \theta_g)$, where θ_g denotes parameters of the generator. The discriminator $D : \mathbb{R}^n \rightarrow [0, 1]$ learns to estimate the probability c that a given sample comes from the training data distribution $x \sim p_d(x)$ such that $c = D(x; \theta_d)$, where θ_d denotes parameters of the discriminator. The GAN learning objective aims at finding a min-max optimisation of value function V between the pair of G and D (i.e., Nash equilibrium) as follows:

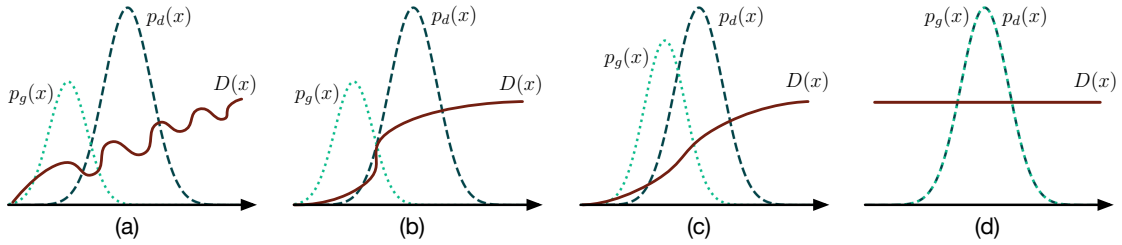

Figure 3.10: Probability distributions for generated data $p_g(x)$, training data $p_d(x)$, and discriminator $D(x)$.

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_d(x)}[\log D(x)] + \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z)))]. \tag{3.37}$$

Both networks are designed to facilitate training through backpropagation. To generate output data $\hat{x}$, the generator takes a latent variable z, and processes it through a stack of upsampling (see Section 3.5.4) and convolutional layers. The discriminator uses the generated output to estimate a divergence measure between the training data distribution $p_d(x)$ and generated data distribution $p_g(x)$. Training is performed through an iterative process that involves providing the discriminator with alternating samples from the generator (i.e., fake samples) and the training dataset (i.e., real samples). Figure 3.10 illustrates the progression of the probability distributions of the generator $p_g(x)$ and the training data $p_d(x)$ over the course of training. In the initial stage of training, the distributions of $p_g(x)$ and $p_d(x)$ exhibit significant differences. After several iterations of discriminator training, the optimal discriminator D is obtained, enabling accurate discrimination between real and generated samples. As generator training continues, $p_g(x)$ becomes more similar to $p_d(x)$, and the distributions begin to overlap. Finally, when $p_g(x) = p_d(x)$, the generator generates samples that are indistinguishable from the training data, as demonstrated in Figure 3.10 (d).

GANs are designed to accurately recover the underlying data distribution $p_d(x)$ at the Nash equilibrium, provided that the model is sufficiently large and the data is infinite. However, the simultaneous training of two models utilising the original GAN formulation introduces several challenges. One prominent issue is the vanishing gradient problem, which arises when the discriminator becomes overly confident in distinguishing between real and generated examples, causing the outputs of D to be too close to either 0 or 1 (Arjovsky and Bottou, 2017). Subsequently, the gradients provided to the generator start diminishing, leading to slow learning and unstable training. Mode collapse is another frequently observed issue, which occurs when the generator produces a limited set of similar samples, resulting in reduced variability in the generated data distribution. To address these issues, Arjovsky et al. (2017) proposed minimising the Wasserstein distance, also known as the Earth-Mover distance, between the generated and training data distributions. This distance, denoted by $W(p_g, p_d)$, can be informally described as the least expense involved in transferring mass to convert distribution p_g into distribution p_d, wherein the cost is determined by the product of mass and the distance over which it is transported. The Wasserstein distance is computationally intractable; however, it can be simplified using the Kantorovich-Rubinstein duality (Villani, 2009). The resulting objective function is:

$$\min_G \max_{D:||D||_L \leqslant 1} = \mathbb{E}_{x \sim p_d(x)}[D(x)] + \mathbb{E}_{z \sim p(z)}[D(G(z))], \tag{3.38}$$

where the discriminator D is employed to approximate the supremum (least upper bound) over a set of all 1-Lipschitz functions. This approach deviates from the standard GAN framework, where D is typically trained as a binary classifier to distinguish between real and generated samples. Instead, the discriminator is now tasked with assigning high and low Wasserstein distances to the samples. Under an optimal discriminator, minimising the objective function with respect to the generator parameters minimises $W(p_g, p_d)$. The Lipschitz constraint is imposed on the discriminator to restrict the range of gradients with respect to the input data, by ensuring that its function is 1-Lipschitz continuous. This constraint is crucial as it provides better gradients to the generator during training, ultimately leading to improved convergence and stability. To enforce the Lipschitz constraint, Arjovsky et al. (2017) proposed the application of a simple clipping function to restrict the maximum weight value in D. To prevent optimisation issues arising from a clipping window that is either too small or too large, Gulrajani et al. (2017) proposed a gradient penalty (GP) that enforces the constraint by adding an additional term to the loss function of the discriminator. GP introduces a penalty based on the norm of the gradients of the discriminator function for random samples $\hat{x} \sim p_g$. Minimising the GP term limits the range of gradients, which helps improve the stability and convergence of the GAN training, especially in cases where the discriminator function may violate the Lipschitz constraint. GP is weighted by a coefficient λ, which is used to balance the strength of the gradient penalty relative to the discriminator loss.

Progressive Growing GAN

Progressive growing of GANs (PGGAN) (Karras et al., 2017) is an extension of the original GAN architecture that uses an alternative training method that allows the model to progressively improve the resolution of the generated images during training. In a PGGAN, the generator and discriminator networks are trained to generate and discriminate high-resolution images, respectively. However, instead of training the networks on high-resolution images from the beginning, the networks are first trained on low-resolution images, and the resolution of the generated images is gradually increased during training. This allows the networks to progressively improve the quality of the generated data, starting from low-resolution data and gradually increasing the resolution as training progresses. This training method can allow GANs to generate high-resolution images with fine details and realistic textures, which can be difficult to achieve using traditional GAN training procedures.

3.5.4 Upsampling Layers

Upsampling layers are a fundamental component of DGMs, particularly in symmetrical network architectures such as variational autoencoders (Section 3.5.2) and generative adversarial networks (Section 3.5.3). They enable the spatial dimensions of the input to be increased, allowing the network to generate progressively higher-resolution outputs. There are several ways to implement upsampling, including transposed convolutions, interpolation upsamplers, and subpixel convolutions. Figure 3.11 provides a visual representation of how a transposed convolution operates when upsampling 2-dimensional and 1-dimensional input data. Transposed convolutions (Zeiler and Fergus, 2014) perform upsampling by first expanding the input feature map with zeros, effectively increasing the spatial dimensions of the input. The resulting feature map is then convolved with a set of trainable filters, which are controlled by the parameters for filter size and stride, to produce an output feature map that is larger than the input. On the other hand, interpolation methods, such as linear or nearest neighbor, upsample the input feature map by computing new values based on the existing values. Linear interpolation calculates the new

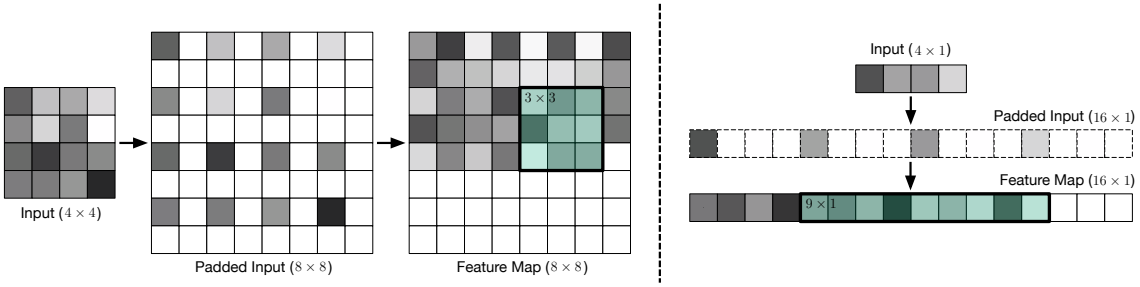

Figure 3.11: Upsampling feature maps using transposed convolution for 2-dimensional feature map (left) and 1-dimensional feature map (right). The process begins with the expansion of the feature maps by padding the input with zeros to achieve the desired output size. These expanded feature maps are then convolved with trainable filters to obtain the upsampled feature map.

values by taking a weighted average of neighbouring points. In contrast, nearest neighbour interpolation assigns the new values based on the closest existing value. Both methods fill the feature maps with these newly computed values instead of zeros during the upsampling process. These interpolated values are then used as inputs to a standard convolution layer. Alternatively, subpixel convolution is a method that uses a convolutional layer to upsample the signal along the channel axis. The output of this layer is then reshaped using a periodic shuffle operation (Shi et al., 2016), which reorders the convolution output to match the desired output shape.

Upsampling layers are typically controlled by a parameter that determines the factor in which an input feature map is upsampled. The choice of upsampling method can impact the quality of the output data, with each method leading to different types of visual or audio artifacts. In the case of audio data, transposed and subpixel convolutions are prone to introduce tonal artifacts, whereas interpolation upsamplers can introduce filtering artifacts, which can be perceptually preferable (Pons et al., 2021).

3.5.5 Conditioning

DGMs can be designed to perform a specific task or make a prediction based on additional input information (Equation 3.32). Conditioning can be useful in a variety of situations where the DGM output depends on some external factors or context. This is particularly useful for enabling the training process to learn the conditional probability distribution of the data x given the labels y. For example, a DGM can be trained to generate specific instrument sounds based on a set of input conditions, such as the type of instrument (e.g., guitar) or its pitch (i.e., frequency). In this scenario, the input conditions would guide the generation of the output audio, providing users with more control over the generated sounds. Methods for categorical conditioning commonly involve encoding conditional labels y as one-hot vectors and concatenating them with the input data (Mirza and Osindero, 2014); however, this can lead to undesirable behaviour such as cross-over between classes. Alternatively, an embedding layer can be implemented as a weight matrix that maps discrete conditioning variables to continuous vectors (Akata et al., 2015). A class-specific encoding is achieved by passing y through the embedding layer with a dimensionality of e. Consequently, each input class is associated with a distinct e-element vector representation. During the training process, the weights of the embedding layer are learned, allowing the resulting vectors to capture the relationships between the conditional variables and their corresponding outputs.

Figure 3.12: Morphing between female and male faces using interpolation in the latent space of a generative adversarial network.

3.5.6 Latent Space Manipulation

Various strategies can be employed to facilitate interactions with DGMs, including conditioning the input based on specific attributes or desired features, regularising the latent space, or altering the model architecture. These approaches allow for enhanced control and adaptability within the generative process. Nonetheless, the viability and methods for facilitating control may differ according to the properties of the training data and the intended application. The output of a DGM predominantly depends on the quality and volume of the data utilised during training, which highlights the importance of meticulously curating training data as one of the most effective means to govern the output of deep generative models.

The latent space is a representation of compressed data that captures the salient features of the training data in a lower dimensional space. In DGMs, the latent space represents the variables that are responsible for generating new data. It is an abstract space where different dimensions of the space represent different aspects or features of the data. New data is generated from a trained generative model by providing it with a sample from the latent space (i.e., a latent vector). By manipulating the dimensions of the latent vector, it is possible to control the output of the DGM and generate new samples with specific features or attributes. Latent space interpolation is a typical manipulation technique in generative models, which involves generating intermediate data points by computing the linear or spherical path between two vectors in the latent space. Figure 3.12 demonstrates morphing between female and male faces by interpolating the latent space of a GAN. This technique enables the creation of novel data points that combine the attributes of two original data points. Enhanced control over the content generation process can be attained by learning a disentangled latent representation, where each dimension encapsulates the salient characteristics of the training data, all the while exhibiting relative invariance to changes in other factors (Bengio et al., 2013). For a dataset of human faces, an efficient disentangled representation could assign separate sets of dimensions to specific attributes, including age, hairstyle, head rotation, and the presence or absence of eyeglasses. In contrast, for a dataset of musical instrument samples, the disentangled representation could assign separate sets of dimensions to pitch, instrument, and playing technique (Bitton et al., 2019). However, in cases where learning is unsupervised, the latent dimensions may not be related to perceptual properties, and thus, may not offer sufficient control over the generative process.

During the training process, the representation of latent dimensions can be explicitly guided by utilising regularisation and conditioning techniques, which contribute to a more controlled and refined model structure. Chen et al. (2016) proposed a modification to the GAN objective that encourages it to learn semantic and meaningful representations from image data by maximising the mutual information between a small subset of the noise variables and the training data. Higgins et al. (2017) introduced a variation

of the VAE framework, aiming to learn disentangled representations by incorporating a hyperparameter in the objective function to balance the trade-off between reconstruction quality and latent variable disentanglement. Within the audio domain, VAE modifications have been proposed to regularise latent spaces, aligning them with perceptual distances obtained from timbre studies (Esling et al., 2018b,a). Adversarial autoencoder variations have also been proposed, which regularise latent spaces based on timbre and playing techniques (Bitton et al., 2019), as well as for transforming rhythmic and timbral attributes of drum recordings (Tomczak et al., 2020).

Inspired by style transfer literature (Huang and Belongie, 2017), researchers have proposed several alternative generator architectures for GANs that facilitate unsupervised separation of semantic attributes, including hairstyle, face shape, and the presence of glasses, when trained on human faces (Karras et al., 2019, 2020b, 2021). In style-based generators, a mapping network is utilised to convert the input latent vector into an intermediate latent space, which yields a disentangled representation of high-level attributes and low-level details embedded in the space. Contrasting with traditional GANs, where latent vectors are directly supplied to the input layer of the generator, the intermediate latent vectors are used to modulate the layers of the style-based generator through learned affine transformations. The intermediate latent vectors can be incorporated by modulating the activations of convolution layers through adaptive instance normalisation (AdaIN) (Karras et al., 2019) or by modulating the weights of the convolutional layers through weight demodulation (Karras et al., 2020b). The mapping network and affine transformations serve as a mechanism to draw samples for each *style* from a learned distribution, while the generator network operates as a means to synthesise new data based on a collection of *styles*.

The aforementioned methods can aid in disentangling semantic attributes; however, the high-dimensional structure of the latent space poses challenges in understanding and manipulating the relationships between individual dimensions and their corresponding effects on the generated outputs. To enhance user interaction with DGMs, various techniques have been proposed to identify meaningful directions in the latent space that correspond with human-understandable concepts. By finding and understanding these latent directions, operations such as interpolation, extrapolation, or attribute manipulation can be performed in the latent space. Latent directions can be discerned through supervised approaches, a process which typically involves the following steps: sampling a multitude of latent vectors from the latent space; generating corresponding outputs (e.g., images, audio); annotating the generated outputs with pre-defined labels based on their attributes, such as object class, colour, size, or style; and training a classifier to establish a mapping between the latent codes and the assigned labels. Supervised approaches have been proposed to represent both discrete (Shen et al., 2020; Jahanian et al., 2020) and continuous factors of variation (Plumerault et al., 2020; Nistal et al., 2020). Alternatively, to circumvent the need for pre-defined classifiers, dimensionality reduction techniques have been employed to uncover latent directions that govern the majority of the variance in the generated outputs in an unsupervised manner (Härkönen et al., 2020; Shen and Zhou, 2021).

DGMs are typically trained to fit a target data distribution, which inherently limits their creative abilities. In the field of computational; creativity, the term active divergence (Berns and Colton, 2020) refers to methods that enable generative models to produce novel and creative outcomes that diverge from the training data. In contrast to typical generative models, where the goal is to minimise the divergence between a given data distribution and approximate distribution, active divergence methods aim to seek the creation of a new distribution that does not resemble the data distribution or any other known data distributions (Broad et al., 2021a). Such techniques for active divergence include training generative

models without data (Broad and Grierson, 2021), fine-tuning pre-trained models to diverge from the original training data (Broad et al., 2020), and integrating deterministic transformation layers into the architecture of pre-trained models (Broad et al., 2021b). Furthermore, Collins et al. (2020) investigated autoencoders as audio effects units, demonstrating that manipulating the internal layers of a trained model (weights, biases and activations) provides as a creative tool for morphing audio in real-time. These approaches offer a promising avenue for advancing the interactivity of DGMs by intentionally challenging models to produce creative results.

3.6 Chapter Summary

This chapter has provided a comprehensive introduction to the various architectures and approaches commonly employed in deep learning, highlighting their respective strengths and weaknesses. Core principles related to neural networks, training procedures, deep generative models, and strategies for manipulating the latent space have been examined. These concepts and techniques are essential for understanding the performance and capabilities of the deep learning approaches employed in the thesis, facilitating a thorough evaluation of the research presented in the following chapters. Deep learning models are effective at analysis and generation due to their ability to learn complex, non-linear relationships between input and output data, which makes them particularly suited for tasks involving high-dimensional and diverse data, such as music. Given sufficient labelled data, deep learning models can be trained to extract features from data with minimal human intervention. This capability proves highly valuable in automating music analysis tasks such as music classification, recommendation, or transcription, as it streamlines the analytical process, enhances overall efficiency, and enables fast and scalable labelling that would otherwise require labour-intensive human annotation. Conversely, deep generative models learn the underlying structure of training data and capture the distribution within a compact latent space. The latent space can be sampled to create new data that exhibits similar statistical properties, and its manipulation allows for expressive interactions with the generative process, unlocking unprecedented creative potential.

Despite considerable advancements in the field of deep learning, there remains untapped potential for its application in SBEM production, a creative endeavour encompassing various labour-intensive tasks that are well-suited for automation. To address this, Chapters 4 and 5 introduce novel deep learning systems aimed at assisting in SBEM production through focused efforts in analysis and generation. The following chapter introduces a new system for labelling samples based on their specific functions within SBEM through the task of automatic instrumentation role classification (AIRC). These labels can assist music producers in identifying compatible samples within unstructured audio databases, generating high-level summaries of SBEM arrangements, and automatically retrieving material suitable for sampling from existing recordings.

Chapter 4

Analysing SBEM with Automatic Instrumentation Role Classification

Chapter 2 delved into sample-based electronic music (SBEM) production, emphasising how technological advancements have shaped the process. Through an examination of the practices of sample sourcing (Section 2.2.1), selection (Section 2.2.2), manipulation (Section 2.2.3), and arrangement (Section 2.2.4), the chapter offered insights into SBEM production. Additionally, it addressed the current challenges and laborious tasks that producers face and discussed related research that could assist in overcoming these obstacles (Section 2.4). In Chapter 3, the preliminaries of deep learning were discussed, introducing various techniques and architectures commonly employed in the field, and highlighting their respective strengths and weaknesses for analysis and generation tasks. In this chapter, a new deep learning system is introduced for automatically labelling samples based on their specific functions within SBEM through the task of automatic instrumentation role classification (AIRC). AIRC is a music tagging task that estimates the presence of active instrumentation role groups within audio recordings. The proposed system is designed to automatically label the instrumentation roles of audio samples, which may represent a single role, including chords, melody, bass, drums, sound effects, or a combination of these roles. Automatically assigning these labels to samples can simplify and streamline the production process by facilitating the identification of samples with specific functions within unorganised audio databases. Moreover, this system can be used to generate high-level summaries of SBEM arrangements by identifying the various instrumentation roles used throughout the composition, and for automatically retrieving samples with desirable characteristics from existing recordings.

The main contents of this chapter can be summarised as follows. Section 4.1 presents an overview of instrumentation roles and their relevance in SBEM production. Section 4.2 provides the method for AIRC with the implementation and training specifications of the various architectures considered. In Section 4.3 the architectures are evaluated in order to determine the optimal configuration for AIRC. In Section 4.4, the structural characteristics of loop-based music are analysed through AIRC, and the performance is compared to that of previous approaches for loop activation transcription, a task that involves estimating the locations in which loops occur throughout a piece of music. To exhibit the capabilities of the AIRC system in identifying sections of music appropriate for sampling, Section 4.5 assesses its ability in detecting the locations of breakbeats (Section 2.1.2) within funk, soul, or jazz recordings. The chapter summary is provided in Section 4.7.

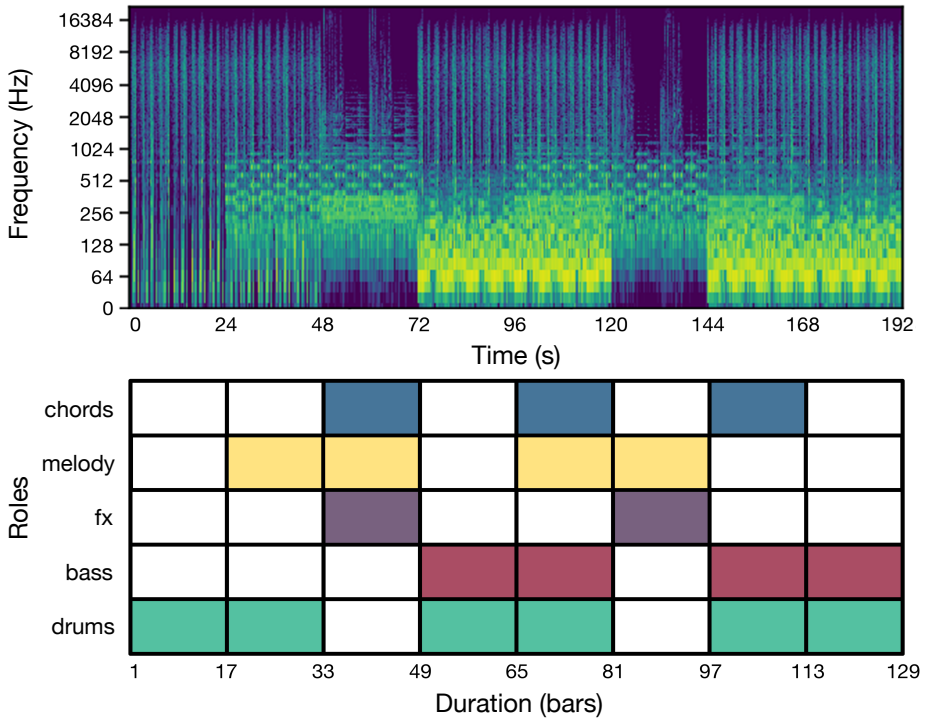

Figure 4.1: A depiction of a simplified SBEM composition structure, created using five loop layers. At the top, a log-scaled STFT power spectrogram is presented, while at the bottom, corresponding instrumentation role activations are displayed at four-bar intervals. These activations include: *chords*, *melody*, sound effects (*fx*), *bass*, and *drums*.

4.1 Instrumentation Roles

In contemporary music production, the practice of categorising musical loops by their instrumentation roles, such as bass, melody, or effects, rather than specific instruments, has gained popularity (Section 2.1.4). This approach to categorisation is evident in established sample libraries, including Splice[1] and Loopcloud[2]. Annotating with instrumentation roles allows producers to explore a wide range of traditional and synthesised timbres without being limited by instrument specificity, which may be difficult to determine when dealing with heavily processed or experimental samples.

Loops often provide as the foundational material that producers use to create and arrange compositions through a variety of editing and combinatory processes, such as layering, splicing, and the application of audio effects (Section 2.1.4). A commonly employed compositional technique involves the repetitive activation and layering of loops, which serve to establish an underlying musical structure (Section 2.2.4). Figure 4.1 demonstrates the structure of a simplified composition constructed by layering and repeating loops with distinct functional roles. The composition commences with the activation of a *drum* loop. Following one cycle, a *melody* loop is incorporated alongside the *drums*. In the subsequent cycle, the *drums* are removed, and two additional layers are introduced: a *chord* loop and a sound effects (*fx*) loop. During the next cycle, *drums* are reintroduced, accompanied by a *bass* loop, while all other loops are

[1] `https://splice.com`
[2] `https://www.loopcloud.com`

removed. Throughout this simplified composition, these loops are selectively activated and deactivated to establish the overall structure.

At present, few resources exist to assist producers in structuring or arranging samples into a finished piece of music (Section 2.4.3). The analysis and visualisation of music structure can play a crucial role in promoting and enriching the understanding and appreciation of musical works among diverse audiences, including listeners, performers, composers, and musicologists. Loop activation transcription (LAT) is a task that leverages the loop-based structure inherent to SBEM. It aims to estimate the roles of loop occurrences within a composition to provide a high-level summary of its arrangement and underlying structure (López-Serrano et al., 2016). Contrary to traditional music transcription that prioritises specific notes or instruments, and structure analysis tasks that investigate phrases such as chorus and verse, LAT concentrates on the transcription of SBEM recordings by highlighting key structural components, encompassing chords, melody, sound effects, bass, and drums. This transcription process offers a valuable solution to analyse existing SBEM recordings by deconstructing them into their constituent elements. This can be visualised in the form of an instrumentation role activation map (IRAM), in which instrumentation roles and their corresponding activations are displayed in a manner akin to the simplified representation shown in Figure 4.1. The IRAM can provide new insights into the underlying structures of SBEM and has many potential use cases in music production and performance such as automatically generating tracks using a set of loops and a reference song, providing visual cues for DJs to anticipate upcoming song events, or to facilitate MIR systems that rely on structural information such as automatic DJ systems (Vande Veire and De Bie, 2018) and music mashups (Davies et al., 2014).

The two recent methods for LAT (López-Serrano et al., 2016; Smith and Goto, 2018) are based on non-negative matrix factorisation (NMF), and while these apporaches allow for the separation of mixed audio into the constituent loops, they rely on non-varying repetitions of loops and do not optimise independence between roles. This presents a challenge when transcribing loop activations in more intricate electronic music compositions, where multiple instruments may fulfil the same role and often exhibit variations through automation and resequencing. This complexity makes it difficult to enforce separation between the roles, resulting in multiple templates for each role. SBEM compositions frequently feature numerous instances of melodic, harmonic, and percussive content, with the instruments and timbre changing while maintaining their original role within the piece. For instance, a sparse drum loop might be used in the introduction, whereas a more dense and complex drum loop could be used during the core section (Section 2.2.4). Similarly, a composition may transition between different synthesiser sounds or alternate between various breakbeats, as described by Hockman (2014). These changes deliver variety without disrupting the underlying structure.

Building on the understanding of activations and instrumentation roles of loops, this knowledge can also be applied to locate *sample-able* material within existing recordings. By leveraging automated systems that recognise and analyse these roles, it becomes possible to identify and extract samples that meet specific criteria. While existing research on sample identification (Section 2.4.4) and retrieval Section 2.4.5) has primarily focused on examining samples previously used by producers and the development of methods to facilitate the navigation of sample collections, the utilisation of automated systems to assess existing music recordings for potential sample material has received limited attention. Automatic retrieval of *sample-able* material could save considerable time and effort, as producers would no longer need to listen through extensive amounts of music and rely on aural perception to find samples fitting specific roles, such as drums or melody. This approach offers an automated method for crate digging

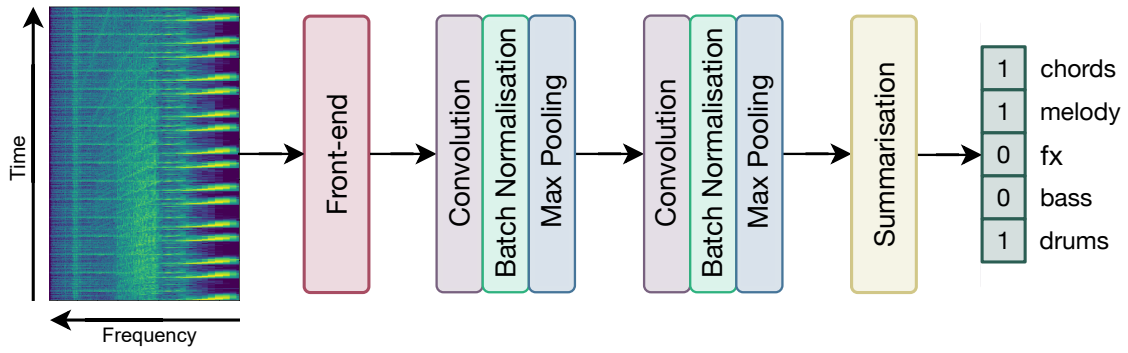

Figure 4.2: AIRC system overview block diagram. Audio loops are input into the network as a spectrogram representation. The front-end extracts features from the incoming spectrogram and subsequent convolutional layers learn a latent representation. Predictions are made using a pooling layer to summarise the information learnt by the network.

(Section 2.2.1), serving as an invaluable tool for artists to quickly identify potential samples within extensive digital recording collections. Additionally, this approach could be used as a preprocessing stage in breakbeat classification (Hockman and Davies, 2015) by identifying the location of the breakbeat prior to classification. This would be beneficial to websites like WhoSampled[3], which currently depend on user input for cataloguing the usage of samples in music.

The system proposed in this chapter expands on the AIRC problem formalisation established by Ching et al. (2020), to full SBEM compositions where multiple instrumentation roles are active concurrently (Drysdale et al., 2022). Research in AIRC has been facilitated by the development of the Freesound Loop Dataset (FSLD) (Ramires et al., 2020b), a large public collection of loops and corresponding instrumentation role annotations from Freesound.[4] The annotations in FSLD encompass chords, melody, bass, sound effects, drums, and vocals. In their experiments, Ching et al. (2020) observed that overusing single labels resulted in reduced accuracy and increased bias due to limited coverage of multi-label annotations in the FSLD. To address this class imbalance, this chapter introduces a novel data augmentation technique and explores its application in combination with various CNN architectures (Section 3.2) and pooling operations to improve performance. The resulting system yields a state-of-the-art AIRC system that has potential applications in various areas of music production, including sample recommendation, transcription, and structure analysis.

4.2 Method

This study evaluates several CNN architectures (Section 3.2) to identify the optimal system for AIRC. Prior studies have shown that automatic music tagging systems exhibit enhanced performance when using spectrograms instead of raw audio signals (Won et al., 2020b), especially when limited training data is available (Pons et al., 2018), as is the case for the FSLD (Ramires et al., 2020b). Figure 4.2 offers a general system overview, demonstrating that audio loops are input into the network in the form of spectrogram representations (Section 3.4), and the network subsequently generates multi-label predictions. The front-end encompasses the primary filters that engage with an incoming spectrogram to

[3] https://www.whosampled.com
[4] https://freesound.org/

extract features, while the subsequent convolutional layers serve to learn a latent representation, which is then summarised to make the instrumentation role predictions. Each architecture utilises different front-end configurations and pooling operations (Section 3.2.2), which derive the final predictions by summarising the information learned by the network.

The diversity of musical audio data used in AIRC, ranging from tonal melodies to heavily processed experimental samples, prompts experimentation with architectures intended for a variety of sound classification tasks. Three different front-ends are considered: general domain square filters; vertical filters (Pons et al., 2017), tailored towards capturing the timbre of melodic instruments; and previous state-of-the-art for AIRC—harmonic band-pass filters (Won et al., 2020a), which capture harmonic relationships while preserving spectral and temporal information. In order to refine the final AIRC predictions, two methods for summarising the information extracted from the final convolutional layers of a CNN are examined. The standard approach is to use global max-pooling (GMP); however, this infers strict assumptions about the label characteristics of the data. In the closely related field of sound event detection, auto-pooling (McFee et al., 2018) has been introduced as a method for automatically selecting the most suitable pooling operation by interpolating between max-, mean-, and min-pooling during the training process.

4.2.1 Architectures

Each network receives audio input in the form of log-scaled Mel spectrogram representations (Section 3.4.5), with features being extracted through stacks of convolutional layers. The output predictions provide values between $[0., 1.]$, indicating the presence of active instrumentation roles. For each network, the input layer is a four-dimensional tensor $t \in \mathbb{R}^{b \times w \times h \times c}$, with batch size b, number of frames w, number of frequency bins h, and channels c.

Square Filter Network

The square filter network (`SF-CNN`) is designed around the utilisation of small, square filters in its front-end layer. This design is motivated by the fact that square filters make minimal assumptions about the local characteristics of the input spectrogram, potentially allowing any structure to be learned through the hierarchical combination of small-context representations. Additionally, previous research has shown that with sufficient data, assumption-free models employing general domain square filters tend to achieve enhanced performance in the domain of automatic music tagging (Pons et al., 2018; Won et al., 2020b). The `SF-CNN` contains four 2-D convolutional layers with 128 small-square filters of size 3×3 and *same* padding. After each convolutional layer, batch normalisation is applied with an ELU activation function. Each convolutional layer, except for the final one, is followed by strided $(2, 2)$ max-pooling. The final convolutional layer uses a summarisation pooling layer before making predictions.

Vertical Filter Network

The vertical filter network (`VF-CNN`) is motivated by the strategy proposed by Pons et al. (2017), which incorporates domain knowledge into the design of the filters used in the front-end of the model. This approach uses rectangular filters to more efficiently capture the timbral characteristics and temporal patterns of spectrograms. It has been shown to perform well in automatic music tagging and musical instrument recognition when limited training data is available (Pons et al., 2017). Figure 4.3 provides an overview of the `VF-CNN` architecture configuration. The input spectrogram is set to be of size 500×128

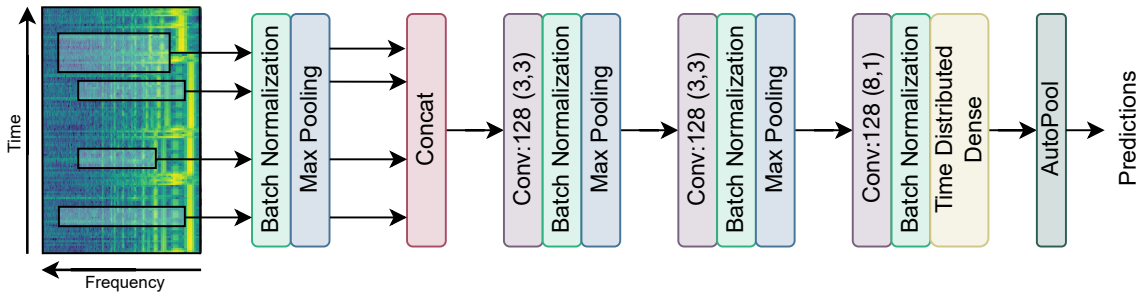

Figure 4.3: Block diagram illustrating the configuration of the vertical filter network with auto-pooling. The black rectangles denote various vertical convolution filter sizes used in the front-end to extract information from an incoming spectrogram. The intermediate layers of the network are summarised into predictions using a time-distributed dense layer and auto-pool, a trainable operator that can adapt to data characteristics by interpolating between min-, max-, or average-pooling.

to accommodate for longer observations of audio loops (see Section 4.2.2). The front-end utilises several vertical convolution filter sizes (black rectangles in Figure 4.3) to efficiently model timbral characteristics present in the spectrogram. Custom filter sizes are used to capture both wide (e.g., chords, bass) and shallow spectral shapes (e.g, drums). The numbers and sizes of filters applied in the front-end are as follows: 128 filters of sizes 5×1 and 80×1; 64 filters of sizes 5×3 and 80×3; and 32 filters of sizes 5×5 and 80×5.

All convolutions in the front-end use *same* padding, and max-pooling is applied to obtain a 16×16 summary of each feature map. This is followed by two 2-D convolutional layers, each featuring batch normalisation (Ioffe and Szegedy, 2015) and exponential linear unit (ELU) (Clevert et al., 2016) activation functions. The first 2-D convolutional layer is followed by strided $(2, 2)$ max-pooling. After the final 2-D convolutional layer, a summarisation pooling layer is used to condense the information learned by previous layers and make predictions.

Harmonic Filter Network

Ching et al. (2020) approached AIRC using a CNN with a data-driven harmonic filter-based front-end (`H-CNN`) (Won et al., 2020a). The benefit of a harmonically stacked trainable representation is that it can capture harmonic relationships while preserving spectral and temporal information. This method is designed to focus on the harmonic content of the input signal, emphasising the frequency relationships that are prominent in pitched instruments. However, AIRC encompasses percussion and noise-like sound effects, and the distinct characteristics of these sounds often lie in their transient nature and distinct attack-decay patterns, rather than in well-defined harmonic structures. The `H-CNN` architecture (Ching et al., 2020) was re-implemented to use as a baseline to test our proposed models. The input t is passed through a set of triangular band-pass filters to obtain a tensor representing it as six harmonics. Harmonic structure is captured by treating the harmonics as channels and processed by a 2-D CNN. The CNN consists of seven convolution layers and a fully connected layer. All but the final convolutional layer is followed by 2×2 max-pooling, batch normalization and a ReLU activation function. Global max-pooling is applied to the final convolutional layer. The output layer is a 5-way fully-connected layer with a sigmoid activation function and a 50% dropout.

Summarisation Pooling

Two pooling operations were considered for summarising the information learned in the final convolutional layers: standard global max-pooling (`GMP`) and auto-pooling (`AUTO`). For configurations employing `GMP`, the final convolutional layer of the network is summarised through max-pooling and then provided to a dense layer consisting of r output neurons, sigmoid activation functions and a 50% dropout.

Following McFee et al. (2018), for the configurations that use `AUTO`, the final convolutional layer uses a filter size $(8, 1)$. This is followed by batch normalisation and a time-distributed dense layer with a sigmoid activation function and r output nodes, where r is equal to the number of classes. In a time-distributed dense layer, the dense layer is applied to each time step in the $(8, 1)$ input sequence separately, preserving the temporal structure of the data. The output of the time-distributed dense layer is provided to the auto-pooling layer, a trainable pooling operator capable of adapting to data characteristics by interpolating between min-, max-, or average-pooling (McFee et al., 2018). This approach enables the model to learn distinct weights for each time step, potentially capturing the temporal dynamics within the input spectrogram, thereby increasing its robustness against time-frequency shifts and perturbations.

Loss Function

AIRC constitutes a multi-label binary classification task; therefore, binary cross-entropy (BCE) (Equation 3.10) is used as the loss function for updating the parameters of each model. In the multi-label setting, each label is considered a separate binary classification problem. A sigmoid activation (Equation 3.3) is applied to the output of each model, mapping the logits to probabilities in the range of 0 to 1 and representing the likelihood of each label being present. The BCE loss is calculated independently for each label, and the errors over all labels are summed.

4.2.2 Network Training

Input audio is pre-processed through resampling and conversion to a spectrogram representation. Audio loops are resampled to 16kHz and the short-time Fourier transform (STFT) of each loop is calculated using a window size of 512 samples and a hop size of 256 samples. For the `H-CNN`, magnitudes of STFT are provided as input to the model. For the `SF-CNN` and `VF-CNN`, the inputs are log-scaled Mel spectrograms (Section 3.4.5) with 128 Mel-frequency bands. All models are trained using the Adam optimiser (Section 3.3.2), with a learning rate $1e{-}4$, and each iteration processes a mini-batch of 8 examples. In order to promote equalised learning, all weights are initialised using He's constant (He et al., 2015). Following Pons et al. (2017), each model uses L2-norm regularisation (Equation 3.19) of filter weights to encourage loudness invariance, except for the harmonic CNN-based models, which use a weight decay of $1e{-}4$ (Won et al., 2020a). Early stopping (Section 3.3.4) is used to complete the training once the model performance ceases to improve over 15 epochs. The epoch that achieves the best accuracy on the validation set is used for testing.

4.2.3 Loop Activation Transcription

Loop activation transcription involves predicting the loop activations of instrumentation roles as they occur over time. Taking advantage of the grid-based structure and consequently fixed tempo of loop-based SBEM (Section 2.2.4), AIRC can analyse the loop structure of a given composition. Instrumental role predictions for complete SBEM compositions are achieved by processing audio files through the

AIRC system in four-bar segments and evaluating the resulting activations. By dividing a full-length SBEM composition into these segments, AIRC can extract instrumentation role activations for each segment, resulting in a form of structural transcription. The AIRC system inherently ensures distinct separation between instrumentation roles, providing a robust solution capable of adapting to variations such as instrument changes and resequencing. This sets it apart from NMF-based approaches, which require loops to be precise repetitions of themselves.

4.3 Evaluation 1: Automatic Instrumentation Role Classification

In the following section, the optimal configuration for the AIRC is identified by evaluating the proposed architecture configurations and data augmentation technique. Specifically, the ability of the system to accurately label audio loops within the FSLD according to their instrumentation roles, which include chords, melody, bass, sound effects, and drums, is assessed. The resulting system could facilitate labelling large sample libraries (Section 2.2.1), particularly those created by music producers who frequently record and splice samples from diverse sources, including vinyl records and digital recordings. Such collections may contain a vast number of audio files that would necessitate considerable human effort to label manually.

4.3.1 Evaluation Methodology

In order to ascertain the optimal configuration for AIRC, the architectures (i.e., `VF-CNN`, `SF-CNN`, and `H-CNN`) and pooling strategies (i.e., `GMP` and `AUTO`) presented in Section 4.2 are evaluated. Following Pons et al. (2018); Ching et al. (2020); Won et al. (2020b), two sets of performance measurements are used for the evaluation: the area under the receiver operating characteristic curve (ROC-AUC) and the area under the precision-recall curve (PR-AUC). ROC-AUC indicates how the number of correctly classified positive examples varies with the number of incorrectly classified negative examples. However, when dealing with an imbalanced class distribution, such as in the FSLD, this measurement can present an overly optimistic view of model performance (Davis and Goadrich, 2006). In such cases, PR-AUC provides a more informative indication of model performance. PR-AUC focuses on the relationship between precision and recall (See Section 4.5.1), which are particularly relevant when dealing with imbalanced datasets, as they better capture the ability to correctly identify the minority class instances (Davis and Goadrich, 2006). Both PR-AUC and ROC-AUC are calculated on a test set for each of the models under evaluation, providing a more comprehensive understanding of model performance. While Ching et al. (2020) also calculates the F-measure score, it was decided to omit this evaluation metric because it depends on a decision threshold applied to the per-class output scores. In contrast, ROC-AUC and PR-AUC measure model performance globally, integrating all possible thresholds.

4.3.2 Evaluation Data

The Freesound Loop Dataset (FSLD) (Ramires et al., 2020b) is used to train and evaluate the architectures described in Section 4.2.1. The FSLD contains a large public collection of audio loops uploaded to Freesound (Font et al., 2013) under Creative Commons licensing. The dataset includes various annotations such as tempo, key, and instrumentation roles. Among these annotations, the

multi-label annotation of loop instrumentation roles is the most relevant for this study. The possible roles encompass chords, melody, sound effects (fx), bass, drums, and vocals. These annotations are crucial for evaluating the performance of the different models on classifying audio loops based on their instrumentation roles. The FSLD contains 2936 loops, with 1531 loops annotated as having a single instrumentation role, while 1405 loops are annotated as having multiple roles. As evident from the class distribution in Table 4.1, the classes in this dataset exhibit significant imbalance.

Drums	54.95	Fx	24.80
Bass	19.10	Melody	21.31
Chords	11.90	Vocal	2.29

Table 4.1: Distribution ($\%$) of instrumentation roles within the FSLD.

In order to adapt the dataset for the task at hand, various modifications are applied to the data. First, all vocal loops are removed as they do not provide sufficient training and testing material. To ensure all remaining loops have a consistent length, adjustments are made to their tempo and duration. Loops are time-stretched to 120 beats per minute (BPM), the most prevalent tempo in the FSLD, which minimises the extent of time-stretching and accommodates the majority of examples in the dataset (Ramires et al., 2020b). Loops exceeding four bars in length are trimmed to exactly four bars (i.e., 8 seconds), while loops shorter than four bars are trimmed to either one or two bars and subsequently repeated until they span four bars. Loops with multiple instrumentation roles are separated from those with only one role, and 70% of each group are randomly selected for training while 30% were reserved for validation and testing. From the latter split, 60% is used for testing (Table 4.2) and 40% for validation.

Drums	27.59	Fx	23.17
Bass	20.33	Melody	18.15
Chords	10.77		

Table 4.2: Distribution ($\%$) of instrumentation roles in the test set.

Besides using the previously described training set of the FSLD, a data augmentation procedure was applied to handle the main imbalance issues on the dataset. These are 1) the lesser presence of loops with more than one instrumentation role (i.e., multi-label) compared to the ones with just one role (i.e., single-label) and; 2) the number of loops for each instrumentation role class, shown in Table 4.3. The data augmentation procedure utilises common production techniques that are used in commercial music recordings including key matching, tempo matching and the use of audio effects such as distortion, reverb and chorus.

Drums	929	Fx	222
Bass	92	Melody	174
Chords	102		

Table 4.3: Distribution ($\%$) of loops annotated as having a single instrumentation role within the FSLD.

Ramires and Serra (2019) proposed an augmentation strategy that applies audio effects to one-shot instrument sounds, aiming to enhance the robustness of models to audio effects. Given that the data utilised in AIRC comprises loops already processed with audio effects, this data augmentation approach

Model	Pooling	Dataset	PR-AUC	ROC-AUC	Bass	Fx	Drums	Chords	Melody
SF-CNN	GMP	FSLD-OG	68.74	83.83	58.72	63.11	95.97	64.74	61.14
SF-CNN	GMP	FSLD-AUG	68.40	81.59	**62.21**	59.80	95.93	62.39	61.68
SF-CNN	AUTO	FSLD-OG	**71.28**	85.12	57.76	59.18	95.98	**73.20**	70.30
SF-CNN	AUTO	FSLD-AUG	68.15	82.19	55.12	68.30	**98.03**	58.81	60.49
VF-CNN	GMP	FSLD-OG	70.62	**85.72**	53.83	**71.73**	97.84	64.90	64.78
VF-CNN	GMP	FSLD-AUG	65.60	80.99	47.11	64.92	97.62	63.11	55.22
VF-CNN	AUTO	FSLD-OG	66.98	82.52	57.59	66.43	95.75	50.37	64.77
VF-CNN	AUTO	FSLD-AUG	67.47	81.40	46.18	67.10	97.05	56.13	**70.89**
H-CNN	GMP	FSLD-OG	61.83	80.39	53.65	42.21	94.10	60.30	58.89
H-CNN	GMP	FSLD-AUG	59.18	77.34	40.30	57.05	94.60	47.92	56.01

Table 4.4: AIRC performance comparison for various model, pooling method, and training data configurations. The PR-AUC, ROC-AUC values, and individual role accuracies (%) are provided for each configuration, where bold indicates the highest scores.

is extended and adapted to balance the distribution of instrumentation roles across the loops in the dataset. This is achieved by first processing the loops through one of the following audio effects: reverb, delay, flanging, chorus, tube saturation, and bitcrushing, resulting in 1000 loops for each of the r classes under observation ($r = 5$), totalling 5000 loops. Following (Ramires and Serra, 2019), the augmentation effects were applied to each loop using the MrsWatson[5] command-line audio plugin host and TAL Software[6] audio plug-ins with the default factory preset parameters. Additional data with multiple instrumentation roles present is then created by overlapping loops from each augmented single-label class. To ensure that all single and multi-role classes contain an equal number of loops, the number of possible combinations $\binom{r}{k}$ is calculated, where k is the number of instrumentation roles in the combination ($2 \leqslant k \leqslant 5$). The loops featuring multiple instrumentation roles are created by harmonically combining the single instrumentation role loops. Single-role loops with compatible modes (e.g., major and minor) are selected for combination, and before merging, the loops are pitch-shifted to their average key. This augmentation process emulates techniques commonly used by SBEM producers when combining samples from different sources (Section 2.2.3).

Upon removal of the original multi-role loops of the training set, this process results in a total of 25000 loops that can be used for training. To evaluate the effectiveness of the augmentation procedure (FSLD-AUG), the accuracy of models trained with FSLD-AUG is compared to those trained with the original dataset (FSLD-OG) using the same test and validation data. The models are evaluated using use the macro-average (MA) of the PR-AUC and of the ROC-AUC as a global metric. For individual instrumentation roles, only the PR-AUC is displayed. Due to the imbalance of the FSLD-OG, which also affects the test set (Table 4.2), MA is used to provide an average accuracy over each class.

4.3.3 Results and Discussion

Table 4.4 presents the results of the AIRC evaluation of the models described in Section 4.2. The ROC-AUC performance measure is consistently higher than PR-AUC. As previously mentioned in Section

[5] https://github.com/teragonaudio/MrsWatson
[6] https://tal-software.com/products

4.3, ROC-AUC metric can lead to over-optimistic scores when the dataset is unbalanced, which is the case for the FSLD-OG. However, the augmentation procedure used to create the FSLD-AUG dataset has balanced the distribution of instrumentation roles, potentially making ROC-AUC a more meaningful metric for comparing model performance trained with this dataset. The best-performing models *w.r.t* PR-AUC are the SF-CNN with AUTO (71.28%) followed by the VF-CNN with GMP (70.61%). Both models surpass the previous state-of-the-art, H-CNN trained on FSLD (61.82%) by a substantial margin. The SF-CNN model generally outperforms its VF-CNN counterpart on both the FSLD-OG and FSLD-AUG datasets. While vertical filters have been demonstrated to produce comparatively better results with tonal musical audio Pons (2019), the data employed in AIRC comprise more experimental and heavily processed audio. The results suggest that the use of general-domain square filters in SF-CNN could potentially provide better performance for handling non-standard types of audio that are commonly encountered in SBEM production. This finding is in accordance with the automatic music tagging evaluation conducted by Won et al. (2020b), where a simple square-filter CNN trained on short excerpts of full music recordings outperformed models that incorporated domain knowledge in their design on a variety of different datasets.

The overall best-performing model in terms of PR-AUC is the SF-CNN with AUTO pooling trained on the FSLD-OG. However, by closely inspecting the results achieved for individual instrumentation roles, it can be seen that it surpasses the PR-AUC achieved by other models in the *Chords* class by almost 10%, while not achieving such a high result in *Bass*, *Fx* and *Drums*. The most accurately predicted instrumentation role for all models is *Drums*, which is expected as this role had the largest number of examples in the FSLD. The systems generally perform worst on *Bass* and *Melody* roles, which are among the fewest occurrences in the original FSLD. The accuracy for predicting the *Bass* role is improved considerably when using a combination of the SF-CNN model with GMP pooling and the FSLD-AUG dataset. On the other hand, the *Fx* role is predicted with the highest accuracy when the VF-CNN model is trained with the FSLD-OG dataset and GMP pooling. While the best three performing models in terms of PR-AUC are trained on the FSLD-OG, it can be seen that the *Bass*, *Drums* and *Melody* role predictions tend to benefit from training with the FSLD-AUG. As each configuration performs better at predicting different instrumentation roles, it is possible to use a combination of the models for classifying individual instrumentation roles. This combination would lead to an average PR-AUC of 75.21%, substantially surpassing each model.

4.4 Evaluation 2: Loop Activation Transcription

As described in Section 4.2.3, LAT can be executed by partitioning a full composition into separate four-bar sections and performing AIRC on each of these segments to identify the active roles. This method produces a high-level summary of the arrangement, represented as an instrumentation role activation map (IRAM). LAT differs from the previous task as it concentrates on complete SBEM compositions, where samples are combined and arranged throughout the composition using various combinatory processes such as layering, applying audio effects, and automation. Additionally, it is more likely that the observed sections will encompass multiple instances of different instrumentation roles, as discussed in Section 2.2.4. The subsequent evaluation illustrates the performance of the proposed AIRC system in executing this task.

4.4.1 Evaluation Methodology

The aim of this evaluation is to investigate the capacity of AIRC for predicting loop activations in SBEM compositions through a comparison of the architecture configurations presented in Section 4.2.1. The best-performing configurations are then compared with the results of the previous approaches to loop activation transcription proposed by López-Serrano et al. (2016) and Smith and Goto (2018). Following these previous approaches, the loop activation predictions are evaluated against a ground truth in terms of accuracy. Since accuracy evaluation requires a binary transcription, a repeated k-fold cross-validation method is employed together with a grid search to identify the best threshold for binarising the predictions of each role. In order to investigate the generalisation of the proposed models, two-fold cross-validation is repeated ten times, for which one fold is used as a validation set to identify thresholds and the other is reserved for computing accuracy against the ground truth. Thresholds for each class are identified by performing a grid search over a range between 0.01 and 1 with a step size of 0.01, then selecting the thresholds which yield the highest accuracy on the validation set.

4.4.2 Evaluation Data

Artificial Dataset

The proposed models are applied to the dataset used by López-Serrano et al. (2016); Smith and Goto (2018). This dataset comprises simplified SBEM compositions, constructed by generating templates using four-bar loops. Designated as the *Artificial* dataset, it demonstrates a simplified approach to SBEM composition by employing loops that are iteratively added and removed. The automatic arrangement method provided by Smith and Goto (2018) is used to generate 21 compositions across seven electronic music genres and three templates–*composed*, *factorial* and *shuffled factorial*. In the *composed* template, loops are introduced and removed iteratively in a manner typical of SBEM structure. The *factorial* template contains all possible combinations of the individual loops, arranged iteratively. The *shuffled factorial* template contains the same loop combinations, with shuffled ordering. The *factorial* and *shuffled factorial* templates are designed to evaluate the model performance on all possible loop combinations within the *Artificial* dataset. In contrast, the *composed* template emulates typical SBEM compositions with respect to the iterative introduction and removal of loops throughout the piece (Section 2.2.4). Following the AIRC procedure (Section 4.3.2), the generated compositions are time-stretched from their annotated tempo to 120 BPM and divided into four-bar loops, which are provided as input to the AIRC system.

Empirical Dataset

Within SBEM, loops are generally created from pre-recorded musical segments that are reiterated throughout a piece (Section 2.1.4). In contrast to the compositions in the artificially generated dataset, modern SBEM compositions often exhibit variations in loops, partially due to the advanced capabilities of contemporary production sampling software, DAW, which facilitate the creation of such variations with ease. DAWs provide an extensive collection of tools that facilitate the processes of automation and resequencing, enabling producers to introduce variations in sample characteristics, such as pitch, timbre, or rhythm pattern. Furthermore, compositions may feature multiple instances of melodic, harmonic, and percussive content, with the instruments changing while maintaining their original role within the piece (Section 2.2.4).

Model	Pooling	Dataset	Mean	Bass	Drums	Fx	Melody
SF-CNN	GMP	FSLD-OG	81.8	69.2	**100.0**	63.4	94.6
SF-CNN	GMP	FSLD-AUG	86.2	71.7	**100.0**	79.6	93.2
SF-CNN	AUTO	FSLD-OG	81.0	66.9	97.3	71.6	88.4
SF-CNN	AUTO	FSLD-AUG	**86.9**	68.3	**100.0**	**85.7**	93.4
VF-CNN	GMP	FSLD-OG	80.9	69.0	99.3	65.8	89.4
VF-CNN	GMP	FSLD-AUG	84.7	71.7	**100.0**	75.7	91.4
VF-CNN	AUTO	FSLD-OG	80.2	63.7	98.6	63.1	**95.3**
VF-CNN	AUTO	FSLD-AUG	82.5	**74.2**	99.7	79.7	76.6
H-CNN	GMP	FSLD-OG	75.1	71.8	96.1	55.6	76.7
H-CNN	GMP	FSLD-AUG	79.5	53.1	95.8	81.6	87.6

Table 4.5: Loop activation transcription accuracy results ($\%$) for the AIRC configurations evaluated on the artificial dataset. The mean accuracy over all roles and individual role accuracies are provided, with bold indicating the highest scores.

To assess the performance of the AIRC system in analysing the structure of SBEM compositions featuring such variations, a dataset of ten empirically annotated compositions was compiled. The dataset covers several popular SBEM genres such as hip-hop, drum and bass, techno, house and garage, with a tempo range of 120-175 BPM. The music was sourced from Soundcloud[7] and Bandcamp [8] and are available under a Creative Commons license. Ground truth annotations were obtained by listening to each composition and labeling the active instrumentation roles at four-bar intervals. Following the artificial dataset, the possible roles include bass, drums, fx, and melody. The annotations for each track were saved a matrix of binary values, in which rows correspond to individual roles, and columns indicate the presence (1) or absence (0) of that role during each four-bar interval. These annotations provide a reference for assessing the performance of the AIRC system in accurately identifying the instrumentation within SBEM compositions. The dataset and corresponding annotations can be accessed from the following link.[9]

The compositions from both *Artificial* and *Empirical* datasets are time-stretched from their original tempo to 120 BPM and divided into four-bar loops, which are provided as input to the AIRC systems. Subsequently, the output of the AIRC systems are evaluated against the corresponding ground truth annotations to assess their accuracy.

4.4.3 Results and Discussion

Artificial Dataset

Table 4.5 presents the LAT results using the AIRC configurations (Section 4.2) to transcribe the compositions in the *Artifical* dataset. The table provides the mean classification accuracy over the instrumentation roles and classification accuracy for each individual role (i.e., *bass*, *drums*, *fx* and *melody*). The overall best performing model uses the `SF-CNN` with `AUTO` pooling configuration trained using the `FSLD-AUG` dataset (86.9%) followed by the `SF-CNN` with `GMP` (86.2%). The evaluation results

[7] `https://soundcloud.com`
[8] `https://bandcamp.com/`
[9] `https://jake-drysdale.github.io/research.html`

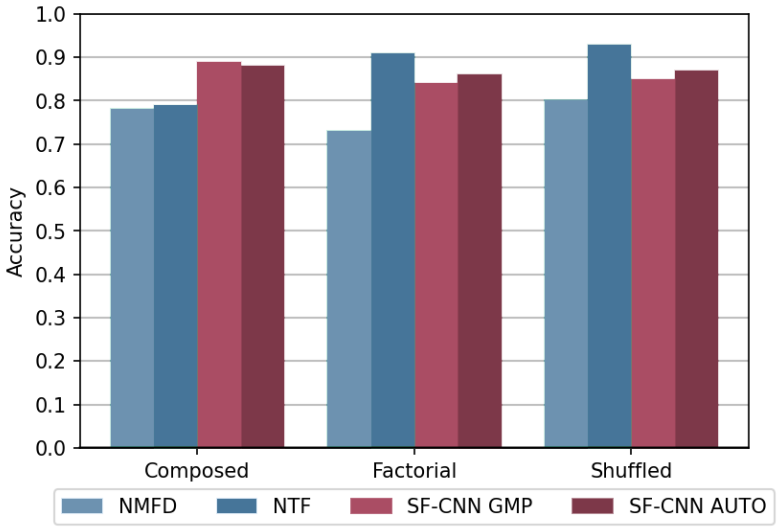

Figure 4.4: Bar graph depicting the mean loop activation transcription accuracy (%) for the three template variations of artificial SBEM compositions. Previous methods (NMFD and NTF) are represented by blue and the two best performing AIRC configurations (`SF-CNN GMP` and `SF-CNN AUTO`) are shown in red.

reveal that models trained with the augmentation technique (`FSLD-AUG`) exhibit enhanced performance in transcribing artificial compositions when compared to those trained on the `FSLD-OG`. The observed difference could be attributed to the balanced distribution of all possible role combinations achieved through the augmentation process, which accommodates the common occurrence of multiple active roles within a given loop structure in the compositions. Drums are classified most accurately for all model configurations with four models achieving 100% accuracy. This was anticipated as the *Drum* role has the largest number of samples in the FSLD dataset, and is usually the most prominent element in SBEM compositions. In some cases, the `VF-CNN` configuration seems to improve the performance of *Melody* and *Bass* roles, which could suggest that the classification of roles containing melodic instruments benefit from using vertical filters at the front-end of the system.

Figure 4.4 presents loop activation transcription results for the three template variations using the two best performing AIRC configurations (i.e., `SF-CNN-GMP` and `SF-CNN-AUTO`) compared with the NMFD (López-Serrano et al., 2016) and NTF (Smith and Goto, 2018) methods previously proposed for LAT. The AIRC architectures outperform the previous methods in regards to accuracy for the *composed* layout, with `SF-CNN-GMP` (light red) achieving the highest score. NTF (dark blue) achieves the best performance for the *factorial* layouts closely followed by the `SF-CNN-AUTO` architecture. As mentioned in Smith and Goto (2018), an additional shortcoming of the NTF and NMFD approaches is that the algorithms depend on loop roles not co-occurring throughout the composition. The proposed AIRC approach enforces independence between the different roles, thus making it more suitable for transcribing loop activations of real-world EM compositions, in which loops often vary through automation and resequencing. Furthermore, the AIRC system has a considerably faster runtime than NTF ($\sim$30 secs per composition) and NMFD ($\sim$10 mins per composition). Predictions for a full EM composition are calculated in under a second using AIRC, which could be beneficial when analysing larger collections of music.

Model	Pooling	Dataset	Mean	Bass	Drums	Fx	Melody
SF-CNN	GMP	FSLD-OG	81.9	73.8	64.4	92.1	97.4
SF-CNN	GMP	FSLD-AUG	81.0	72.8	62.1	89.5	**99.7**
SF-CNN	AUTO	FSLD-OG	88.6	**98.3**	82.3	86.2	87.6
SF-CNN	AUTO	FSLD-AUG	87.5	85.9	73.5	92.7	97.7
VF-CNN	GMP	FSLD-OG	**89.4**	94.8	68.9	**97.0**	96.9
VF-CNN	GMP	FSLD-AUG	80.5	79.5	54.7	88.2	99.6
VF-CNN	AUTO	FSLD-OG	81.6	55.4	**88.0**	83.6	99.3
VF-CNN	AUTO	FSLD-AUG	78.6	89.6	58.3	68.2	98.2
H-CNN	GMP	FSLD-OG	78.6	79.2	61.3	75.9	97.9
H-CNN	GMP	FSLD-AUG	78.4	73.5	57.7	83.9	98.5

Table 4.6: Loop activation transcription accuracy results (%) for the AIRC configurations evaluated on the empirical dataset. The mean accuracy over all roles and individual role accuracies are provided, with bold indicating the highest scores.

Empirical Dataset

NMF-based approaches enable the separation of mixed audio into the individual loop sources; however, they face challenges with variation and automation, making it arduous to enforce separation between the instrumentation roles and leading to multiple templates for each role. This limitation is problematic when a high-level summary of a composition is desired. In contrast, the AIRC system does not rely on loops being exact repetitions of themselves, allowing for more flexibility and variation. For instance, different instruments can be used throughout the composition while still maintaining their intended role, which is achievable due to the capacity of the AIRC system to consider the wider context of each segment, rather than relying solely on the exact repetition of loops. As a result, comparisons between the AIRC system and NMF-based methods on the empirical dataset were not possible. Instead, the various AIRC configurations were compared to identify the optimal method for this task.

Table 4.6 presents the loop activation transcription results obtained using the AIRC configurations (Section 4.2) for transcribing the compositions in the *Empirical* dataset. The highest performing model, in terms of mean accuracy, is the VF-CNN with GMP trained on the FSLD-OG (89.4%), closely followed by the SF-CNN with AUTO pooling (88.6%). However, there is a considerable decline in performance for the drum class in the VF-CNN GMP model, while the SF-CNN AUTO model demonstrates consistently strong performance across all roles. The results indicate that configurations with AUTO pooling perform significantly better at classifying drums in comparison to those that utilise GMP. Moreover, *Melody* is classified most accurately for most model configurations with four models achieving above 99% accuracy. In each configuration, there is a trade-off between the accuracy of role classifications. Considering this, a hybrid approach that combines the strengths of different models might serve as a more effective solution.

The results indicate that the examples in the empirical dataset yield better loop activation transcription performance when training with the FSLD-OG than the FSLD-AUG. The reason for this may lie in the fact that the examples in the augmented dataset were combined using an artificial technique, while loops in real-world compositions are typically combined by human producers with careful attention to balancing amplitude levels, equalisation, and other factors. While the augmentation technique includes features such as key matching and time-stretching to more effectively combine loops, human producers are likely better equipped to make decisions about which loops fit together best. Improving the augmentation

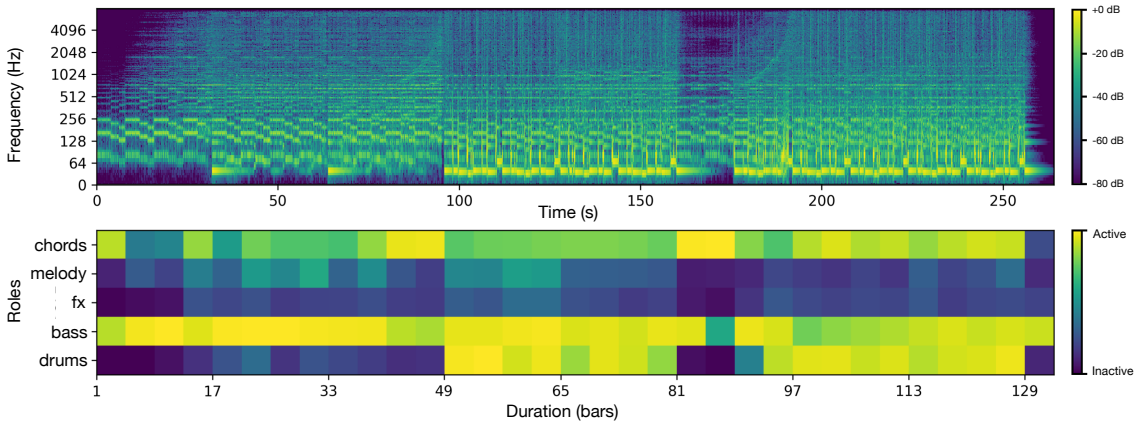

Figure 4.5: Estimated loop activation structure of *Joyspark* (2020) by Om Unit using the proposed AIRC system. Log-scaled STFT power spectrogram of the EM composition (*top*) and estimated templates corresponding to the loop activations showing predictions for each class: chords (`C`), melody (`M`), sound effects (`F`), bass (`B`), and drums (`D`) at four-bar intervals (*bottom*).

technique to more closely resemble the way real producers combine loops, such as by introducing equalisation and balancing amplitude levels of combined loops, may lead to better system performance. In SBEM compositions, samples of different instruments often appear louder or quieter in a mix. As in the artificial datasets (López-Serrano et al., 2016; Smith and Goto, 2018), the augmentation technique currently sums loops together without considering how amplitude levels would typically be balanced by a producer.

Figure 4.5 presents an *instrumentation role activation map* (IRAM) of the composition *Joyspark* (2020) by Om Unit[10] using the proposed AIRC method for loop activation transcription. For visualisation and comparison, a log-scaled STFT power spectrum of the EM composition is shown above the IRAM. The IRAM allows us to visualise activations for each role over the duration of the EM composition, where each square is a measurement of four bars. Furthermore, it can be seen see how each role develops throughout the EM composition. For example, the melody role activations progressively increase between bars 1–41, which corresponds with a synthesiser arpeggio that is gradually introduced by automating the cut-off frequency of a low-pass filter. Additionally, the chord role activations increase between bars 1–49 in correlation with the chords in this section that gradually increase in volume. Activations for the drum role also correlate well with the composition as can be seen between bars 49–81 and 97–129—the only sections that contain drums. Finally, the key structural sections of the composition are easily identifiable. For example, the introduction to the composition (bars 1–49) begins relatively sparse in the composition and IRAM; whereas, bars 49–81 and 97–129 are quite clearly the *core* of the piece—that is, the most energetic sections of the composition typically established by the *drop* (Yadati et al., 2014).

4.5 Evaluation 3: Automatic Retrieval of Samples from Existing Recordings

When searching for samples in existing music recordings (i.e., crate digging), SBEM producers usually listen to a wide range of music, from different genres and time periods, analysing tracks for specific

[10] `https://omunit.bandcamp.com/track/joyspark-bandcamp-exclusive`

sections or elements that could be incorporated into their own productions (Section 2.2.1). This process is a meticulous listening process that requires attentively listening to the entire content in search of an appealing section to sample (López-Serrano et al., 2017). Producers often listen for isolated sections of music containing melodies, drums, basslines, and sound effects (Ableton, 2012; Kim, 2015; Weiss, 2016). Isolated instrumental components hold significant appeal due to their versatility and potential for integration into new compositions. For example, when sampling breakbeats, the presence of other harmonic music or instruments can restrict their flexibility and adaptability (Hockman, 2014; López-Serrano et al., 2017). These samples can be readily modified and serve as the rhythmic foundation in the development of new musical compositions (Morey and McIntyre, 2014). For example, a producer may select a breakbeat from a funk recording and employ it as the foundation for a hip-hop piece, while integrating their own creative elements on top. Alternatively, they might take a solo instrument from a jazz performance (e.g., a saxophone) and divide it into phrases or individual notes, subsequently using these segments to create personalised melodic motifs within their own production 2.2.3. The following section proposes a new approach for the automatic retrieval of samples from existing recordings using the AIRC system. This approach aims to streamline the process of finding suitable samples from large collections of music.

While the previous evaluations (Section 4.3 and Section 4.4) demonstrate the capability of the AIRC system to identify isolated instrumentation roles in loop-based music, there is a need to investigate its performance on genres commonly sampled in SBEM. Currently, there is no existing ground truth dataset that provides annotated locations of sampled sections in existing music recordings, which makes it difficult to evaluate the effectiveness of algorithms designed for this task. However, there is a dataset available for the closely related task of breakbeat detection, which involves identifying passages that only contain isolated drums in funk, soul, or jazz recordings (López-Serrano et al., 2017). This dataset, as described by López-Serrano et al. (2017), is used in this section to evaluate the AIRC system for sampled section detection, as it shares similarities with the task of breakbeat detection in terms of the need to identify specific portions of a musical track. Although the breakbeat detection dataset is designed to identify drum-only sections, the system evaluated on this dataset has the potential to generalise to other types of audio samples in music recordings, such as melody, chords, and bass. As such, while the primary focus of this evaluation is on the ability of the AIRC system to detect drum-only sections, its broader implications suggest potential benefits for music producers seeking to identify suitable sample sections in a track that include other types of audio samples beyond drums (e.g., chords, melody, bass).

Figure 4.6 presents a selected portion of *Amen, Brother* (1969) by The Winstons, which displays the waveform, ground truth instrumentation role activations, and the location of the breakbeat section. As can be observed, the breakbeat section is characterised by its homogeneous instrumentation, which sharply contrasts with neighbouring segments that feature additional active roles. The absence of tonal and harmonic content in this particular segment renders it a highly suitable candidate for sample extraction. This quality affords producers the greatest degree of flexibility when incorporating their own tonal and harmonic content, thereby circumventing the risk of dissonant compositions (Section 2.1.2).

4.5.1 Evaluation Methodology

The identification of breakbeats constitutes a binary classification task, where each observation is classified as either having a breakbeat or not having a breakbeat. To evaluate the performance of the

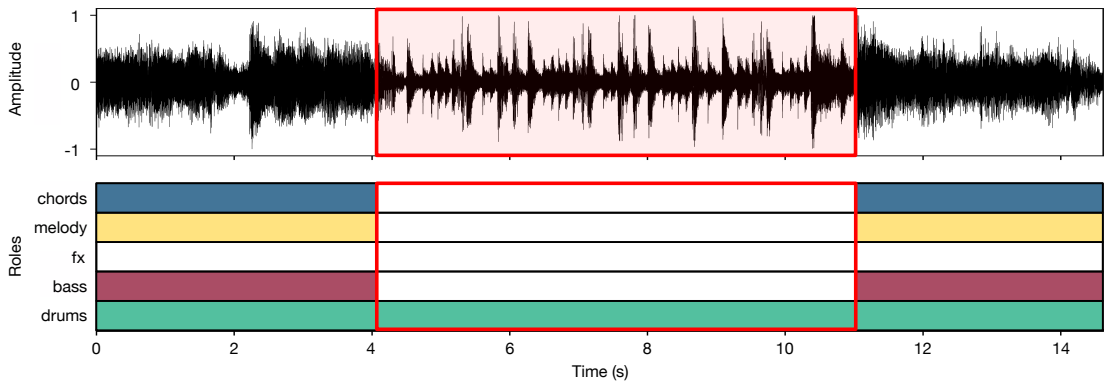

Figure 4.6: An excerpt from *Amen, Brother* (1969) by The Winstons. Above is an audio waveform and below are the instrumentation role activations. The area highlighted in red displays the location of the Amen break sample.

proposed AIRC system in identifying breakbeat sections in pre-recorded music, precision, recall, and F-measure metrics are used. An observation is either a true positive (TP), which is a valid breakbeat prediction; a false positive (FP), which is an invalid breakbeat prediction, or a false negative (FN), which represents instances where a breakbeat section in the ground truth data was not identified by the model. Precision (P) measures the proportion of TP predictions among the observations classified as positive by the model. It indicates how accurate the model is when predicting positive instances, and a high precision score suggests that the model is making fewer FP predictions. The precision is calculated as follows:

$$P = \frac{TP}{TP + FP}. \tag{4.1}$$

Recall (R) measures the proportion of true positives (TP) that are correctly classified by the model out of all actual positive instances in the dataset. It indicates how well the model is able to identify positive instances, and a high recall score suggests that the model is making fewer FN predictions. The recall is calculated as follows:

$$R = \frac{TP}{TP + FN}. \tag{4.2}$$

The F-measure (F) takes both precision and recall into account is calculated as follows:

$$F = 2 \cdot \frac{P \cdot R}{P + R}. \tag{4.3}$$

To determine the optimal thresholds for binarising the AIRC predictions in identifying breakbeat sections in pre-recorded music, the evaluation involves employing a cross-fold validation and grid-search approach on a validation subset of the evaluation data (Subsection 4.5.2). Specifically, the thresholds are identified by performing a grid search over a range between 0.01 and 1 with a step size of 0.01, and then selecting thresholds that provide the highest accuracy on the validation set. To reduce the computational complexity involved in optimising all possible thresholds for all classes and all models, the study proposes

a simplified rule: if all roles apart from drums are below a certain threshold and drums is above that threshold, the composition is classified as a breakbeat; otherwise, it is classified as non-breakbeat. Finally, the performance of the AIRC system is evaluated by identifying the thresholds that achieve the highest F-measure on the validation data, which is then tested on a test split using the thresholds obtained from validation.

4.5.2 Evaluation Data

To evaluate the AIRC system for detecting sections suitable for sampling, the breakbeat dataset compiled by López-Serrano et al. (2017) is utilised. The dataset consists of 280 full recordings spanning various genres, including funk, soul and jazz. All audio files are mono WAV files sampled at 22050 Hz with 16-bit resolution. Each recording is annotated with time intervals enclosing the breakbeat section and predicted tempo. In the dataset, the minimum breakbeat length is 0.84 seconds, the mean length is 10.23 seconds, and the maximum length is 224.84 seconds.

The music present in the evaluation data differs from the SBEM music used in Section 4.4 in that it lacks a grid-based structure and often exhibits tempo fluctuations, making it difficult to partition the recordings into metrical segments. To overcome this challenge, downbeat detection is used to identify the locations of downbeats, which are subsequently used to segment the evaluation data appropriately. The state-of-the-art downbeat detection algorithm proposed by Böck et al. (2016) is employed, which achieves an F-measure of 0.97 on the HJDB dataset Hockman et al. (2012), a dataset that includes genres characterised by their use of breakbeats.

Given that the annotations do not consider any metrical structure, the ground truth time intervals are snapped to the nearest downbeat to facilitate comparison with predicted locations. Furthermore, the AIRC system expects input segments of eight seconds (i.e., four bars at 120 BPM); however, the annotated breakbeats can be as short as 0.84 seconds. To avoid retraining the models on shorter segments, recordings are time-stretched to 120 BPM using the tempo annotations, sliced into bar-length segments at each downbeat and concatenated to form four-bar segments.

4.5.3 Results and Discussion

Table 4.7 shows the mean F-measure, precision, and recall results for breakbeat identification using AIRC configurations. The best-performing model is the `SF-CNN` with GMP trained on the unaugmented FSLD, achieving high recall (85%), indicating that a high percentage of the actual breakbeats in the audio were correctly identified, and a low rate of false negatives. The precision ensures that the predicted breakbeats are accurate and that the system is not producing false positives that could lead to incorrect analysis of the audio. The recall could be considered the most important metric for breakbeat idenfication, as it indicates how well the system is capturing actual breakbeats in the music. Although precision is significantly lower than the recall, indicating that some non-breakbeat sections have been classified as breakbeats, these sections could be filtered out through manual refinement, which is an acceptable trade-off as the system is capturing a good deal of the actual breakbeats.

In general, the models trained without the proposed data augmentation technique (`FSLD-AUG`) outperformed those trained with it for this task. This was somewhat expected since the effects applied through the augmentation technique, such as distortion, pitch-shifting, and bitcrushing, are less likely

Model	Pooling	Dataset	F-measure	Precision	Recall
SF-CNN	GMP	FSLD-OG	**0.67**	0.63	**0.85**
SF-CNN	GMP	FSLD-AUG	0.64	0.64	0.79
SF-CNN	AUTO	FSLD-OG	0.64	0.64	0.78
SF-CNN	AUTO	FSLD-AUG	0.65	**0.67**	0.73
VF-CNN	GMP	FSLD-OG	0.65	0.66	0.74
VF-CNN	GMP	FSLD-AUG	0.61	0.62	0.77
VF-CNN	AUTO	FSLD-OG	0.61	0.58	0.80
VF-CNN	AUTO	FSLD-AUG	0.49	0.48	0.70
H-CNN	GMP	FSLD-OG	0.43	0.42	0.60
H-CNN	GMP	FSLD-AUG	0.41	0.41	0.55

Table 4.7: Breakbeat identification results for various model, pooling method, and training data configurations. The table includes the F-measure, precision, and recall values for each configurations. Bold values indicate the highest scores.

to be apparent in funk, soul, and jazz music. In most instances, the SF-CNN models demonstrated superior performance compared to the VF-CNN models, implying that square filters appear to generalise more effectively for tasks of this nature. A noteworthy finding from this experiment is that despite being trained on electronic music loops from Freesound, the AIRC system can successfully identify breakbeats in a variety of different genres that use contrasting production techniques, instrumentation, and are often recorded from vinyl. Jazz, funk, and soul music have different sound characteristics and often more complex structures, making them less predictable than loop-based electronic music. Despite these differences, the AIRC system was still able to successfully identify breakbeats in these genres, demonstrating its versatility and applicability beyond its original training data. This discovery carries notable significance, especially considering the limited availability of datasets with annotated sample regions, providing a valuable workaround for sample detection in such music where existing datasets are not sufficient to train a deep learning model.

Figure 4.7 illustrates the role predictions, breakbeat predictions (blue line), and ground truth annotations (red line) for the song *Amen, Brother* (1969) by The Winstons. In the breakbeat region, drum predictions are prominently indicated by the yellow hue, while all other role predictions remain subdued (dark blue). For *Amen, Brother*, an iconic example, the predicted locations align accurately with the ground truth data. By leveraging this information, the breakbeat section can be efficiently extracted from the original recording, eliminating the need for a producer to listen to the entire track. This method saves both time and effort, particularly when searching for breakbeats in a large music collection.

4.6 Conclusions

The evaluations conducted in this study have provided insights into the performance and effectiveness of various CNN-based front-ends, pooling strategies, and a data augmentation technique for AIRC, LAT, and the detection of samples in existing records. The results from the AIRC evaluation (Section 4.3) demonstrate that employing the SF-CNN with AUTO pooling significantly enhances performance (71.28%) compared to the previous H-CNN baseline (61.83%) in terms of PR-AUC. The AUTO pooling method enables the model to capture the temporal dynamics within the input spectrogram, thereby increasing

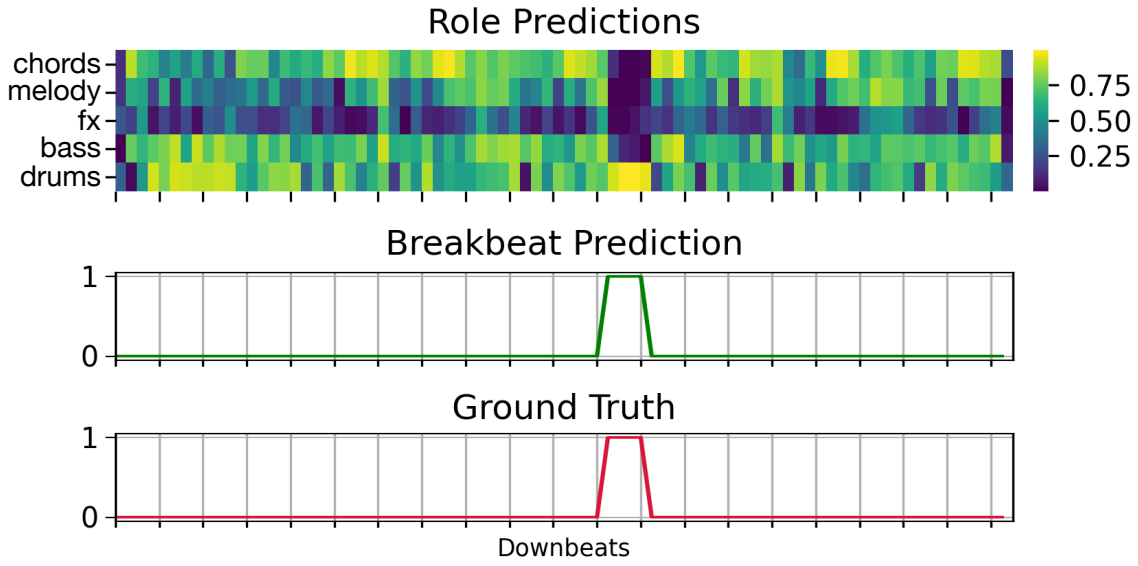

Figure 4.7: Estimated location of the breakbeat in *Amen, Brother* (1969) by The Winstons using the proposed model. Predicted role activations (*top*), predicted breakbeat location (*middle*) and ground truth location (*bottom*).

its robustness against time-frequency shifts and perturbations, leading to improved accuracy. Although the augmented dataset (`FSLD-AUG`) did not enhance the overall performance of AIRC, it contributed to improved accuracy for some of the less represented instrumentation roles, such as *Bass* and *Melody*, within the FSLD. As each configuration performs better at predicting different instrumentation roles, it is possible to use a combination of the models for classifying individual instrumentation roles, which would lead to an average PR-AUC of 75.21%.

The second evaluation (Section 4.4) demonstrated the effectiveness of AIRC for transcribing the instrumentation role activations in SBEM compositions. The results for the artificial dataset of simplified SBEM compositions revealed that models trained with the `FSLD-AUG` dataset exhibited enhanced performance. Compared to the previous NMF-based approaches to LAT, the AIRC system improved performance on the composed layouts by 10%, while achieving similar accuracy on the factorial layouts. A major advantage of the AIRC system is its capability to operate without depending on loops being precise duplicates, while simultaneously ensuring clear separation between instrumentation roles. This distinguishes it from NMF-based methods, which necessitate exact repetition of loops, constraining their adaptability in comparison. The AIRC system offers a robust solution for LAT that can adjust to variations, such as instrument changes and resequencing. As a result, AIRC proved highly effective in transcribing instrumentation role activations in real-world SBEM compositions. In terms of accuracy, the best performing configurations for this task were the `VF-CNN` with `GMP`, achieving 89.5%, and the `SF-CNN` with `AUTO` pooling, attaining 88.6%.

The third evaluation (Section 4.5) demonstrated the capacity of the AIRC system for identifying the location of breakbeats in funk, soul and jazz recordings. Despite being trained on data sourced from a community-based collection of user-uploaded loops and samples (FSLD), the models can be applied to fully produced music recordings from various genres and styles. The best performing model for breakbeat detection in terms of F-measure was the `SF-CNN` with `GMP` trained on the `FSLD-OG`. The high recall of 0.85 achieved in the breakbeat identification task indicates that the model can accurately detect a

considerable proportion of breakbeat locations within the music. Although the lower precision rate points to some false identifications, the high recall rate suggests that a substantial portion of the predicted segments likely contain breakbeats. These results demonstrate the potential of the AIRC system for identifying *sample-able* material from large collections of existing music. Additionally, the prediction thresholds could be tuned and manually refined to allow the system to be more or less lenient when searching for samples of a particular instrumentation role.

In conclusion, the results of each evaluation reveal that the selection of model configuration affects the accuracy of instrumentation role classifications. While the `SF-CNN` with `AUTO` pooling, trained with the `FSLD-OG`, consistently performed well across all tasks, a hybrid approach that combines the strengths of different models might enable more reliable classification of loops within collections of audio.

4.7 Chapter Summary

This chapter introduced a new deep learning system designed for automatic instrumentation role classification, enabling the identification of samples based on their specific functions within SBEM. The system is adept at determining the instrumentation roles of audio samples, which may represent single roles, such as chords, melody, bass, drums, and sound effects, or a combination of these roles. The versatility of the system was demonstrated by conducting three experiments, each representing distinct SBEM production tasks (Section 2.2). The first experiment identified the optimal configuration for AIRC through a comparison of various CNN-based architecture configurations. AIRC can assist in the production process through efficient labelling and organisation of unstructured sample collections, which in turn facilitates the identification of samples with particular structural roles or characteristics. In the second experiment, AIRC was employed to generate high-level summaries of SBEM arrangements by identifying the different instrumentation roles present throughout a composition. The resulting instrumentation role activation map serves as a visual guide, assisting producers in understanding the structural roles of individual sections within a full SBEM composition. Additionally, this visualisation enables listeners to appreciate the relationships between various musical elements and gain a deeper understanding of the composition. The third experiment showcased the automatic retrieval of samples with desired characteristics from existing recordings. This can assist SBEM producers by eliminating the time-consuming process of attentively listening to an entire recording to find samples that fit specific roles in their music. It is important to acknowledge that the sample detection system would depend on the user-provided music recordings, and thus the quality and suitability of the identified samples still rely on individual taste and preference. Consequently, the system does not diminish a producer's sampling abilities or the pursuit of finding unique and personalised samples. The system could be utilised with an SBEM producer's personal collection of tracks across various genres, generating a new sample library from the existing recordings.

Overall, this chapter has provided insights into the performance of deep learning systems within the context of SBEM analysis. The next chapter employs deep generative models to learn the underlying characteristics of a drum sample collection, thereby facilitating the continuous generation and exploration of intermediate samples.

Chapter 5

Drum Sample Synthesis with Generative Adversarial Networks

Drums and percussion play a significant role in shaping the rhythmic foundation of SBEM (Section 2.3). Integrating drum samples into an SBEM composition may involve the time-consuming task of browsing music and sample libraries for suitable drum recordings (Section 2.2.2) or using traditional synthesis techniques (Section 2.3.3), which demand mastery over numerous parameters and provide limited control over sound generation. In light of these challenges, neural audio synthesis (Section 2.4.6) emerges as a solution, offering the capability to generate and manipulate audio samples effectively. Neural audio synthesis enables the generation of new audio samples without being restricted to hand-designed components, such as oscillators and wavetables, or a specific synthesis process such as those discussed in Section 2.3.3. This approach can help overcome the restricted timbre diversity in sample collections and alleviates the challenges associated with complex synthesis techniques, which typically involve steep learning curves.

In the previous chapter, a deep learning system was introduced for analysing SBEM through instrumentation role classification. This chapter shifts focus to the deep generative modelling paradigm (Section 3.5) and presents a system for synthesising drum samples using GANs. While neural audio synthesis can be approached through deep generative models such as autoregressive (AR) models (Section 3.5.1), variational autoencoders (VAE) (Section 3.5.2), and GANs (Section 3.5.3), each method has its limitations. Although AR models can produce high-quality audio, the sequential waveform generation process comes at the expense of resource-intensive and potentially slow inference. In contrast, both VAEs and GANs are capable of generating samples in parallel, providing faster and more efficient synthesis processes. This parallel generation capability is particularly desirable in the context of SBEM production, where producers require timely feedback when creating and refining sounds. However, VAEs employ variational approximations that have been known to produce blurry outcomes in both image (Huang et al., 2018) and audio (Aouameur et al., 2019) generation. GANs, on the other hand, learn to produce precise samples through an adversarial learning strategy, in which a generator and discriminator network are trained simultaneously in a competitive manner until the generator can produce examples that are indistinguishable from the training data. While this can lead to some modes of the data being underrepresented, the combination of fast inference and precision makes GANs a suitable choice for synthesising drum sounds in SBEM production.

The choice of audio representation (Section 3.4) used for training neural audio synthesisers is another important design consideration. GANs for audio synthesis have been successfully trained using both raw audio (Donahue et al., 2019) and perceptually-informed spectrogram representations (Engel et al., 2019). Unlike audio analysis tasks, which often only require the magnitude of the spectrogram, achieving accurate audio generation heavily relies on preserving phase information. While the magnitude of the spectrogram represents the energy distribution across different frequencies over time, phase information describes the relative position of the waveform at each frequency component. Most perceptually-informed spectrograms discard phase information and are thus non-invertible. While the missing phase information can be estimated through lossy estimations (Griffin and Lim, 1984) and learned inversion models (Arık et al., 2018; Shen et al., 2018), these methods require an additional processing stage and often introduce undesirable artifacts.

To address this challenge, Engel et al. (2019) proposed to model phase as instantaneous frequencies, which capture the local frequency content of a signal at any given point in time and can be used for approximate inverse linear transformation. While this representation is suited for representing signals with well-defined, stable frequencies, such as those produced by pitched instruments, drums often have a more complex and less defined harmonic structure, which makes it considerably harder to model the phase relationships between different frequency components. Percussive instruments produce transient and non-periodic waveforms (Karplus and Strong, 1983), meaning their frequency components and phase relationships change rapidly over time. This rapid change makes it challenging for a model to learn and generate accurate phase information for percussive instruments, as the phase relationships between different frequency components are less predictable and more difficult to capture. Generating raw audio directly avoids the need to reconstruct the audio signal from its time-frequency representation, thus preserving phase coherence naturally. This can be particularly beneficial for drum sounds, where phase information plays a crucial role in accurately reproducing the transients and preserving the overall sound quality.

For these reasons, a GAN that operates on raw audio waveforms is adopted for synthesising drum samples in SBEM production. The system is designed to be lightweight, in that it is capable of achieving high-quality audio generation with a relatively small training dataset, which emulates the personal collection of drum sounds typically used by music producers. The system utilises high-level conditioning to categorise drum sounds into specific types, and further employs a compact latent space with low dimensionality, enabling intuitive synthesis control over a diverse range of drum sounds. Furthermore, the interpolation of the latent space constitutes an effective technique for navigating, fine-tuning, and blending the generated drum samples. While existing tools for navigating sample collections have allowed sounds with similar characteristics to be positioned closely together in a 2-dimensional space (Section 2.2.2 and Section 2.4.5), the proposed system enables continuous exploration of intermediate samples, facilitating the discovery of sounds that possess combined characteristics of neighbouring samples. The system is realised through a conditional Wasserstein GAN (Section 3.5.3) trained with a dataset of labelled drums sounds. Conditioning is achieved with the three main percussion instruments from the common drum kit—that is, kick drum, snare drum, and cymbals—and the system can generate a diverse range of sounds from each class.

The main contents of this chapter can be summarised as follows: Section 5.1 presents the proposed methodology to drum sample synthesis and details the different architecture configurations and training procedures. Section 5.2 presents the evaluation methodology, which includes a description of the

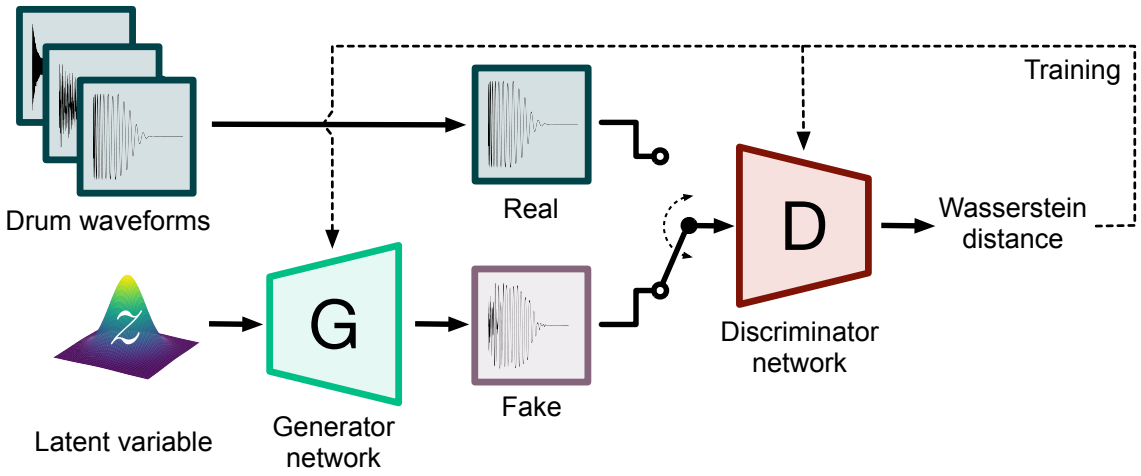

Figure 5.1: General overview of the proposed system for drum sample synthesis: Generator *G* is trained to generate audio given a latent variable *z*. Discriminator *D* is trained to minimise the Wasserstein distance between the generated distribution and the observed distribution.

dataset, the experiment hyperparameters, and the metrics used for evaluating the performance of the different architectures. The evaluation results and discussion are provided in 5.3. Section 5.4 introduces several strategies for controlling synthesis by means of manipulating the latent space. In 5.5, additional experiments are provided, which explore alternative use cases for the proposed system. The chapter summary is provided in 5.6.

5.1 Method

A general overview of the proposed method for drum sample synthesis is presented in Figure 5.1. Generator G is trained to generate drum sample waveforms given a latent vector z, and discriminator D is trained to estimate the Wasserstein distance between the generated and observed distributions $W(p_g, p_d)$ (Section 3.5.3). Both networks are optimised simultaneously until G can produce drum samples that are indistinguishable from the observed training data. The system is trained using a collection of one-shot drum samples and subsequently generates new drum samples by sampling from the latent space of the generator.

The GAN framework defines an adversarial game between generator network G and discriminator network D (Section 3.5.3). G is used to learn mappings from a noise space Z to drum data space X. $Z = \mathbb{R}^{d_z}$, where d_z is a hyperparameter that controls the dimensionality of Z. Latent variables $z \in Z$ are sampled from a known prior $p(z)$, which is modelled with a Gaussian distribution. X is the drum data space that represents the input to D or output of G. As training data, drum samples T are drawn from a real distribution $p_T(x)$. By sampling from $p(z)$, G can be used to generate drums that represent a synthetic distribution $q(x)$. In order to facilitate conditional generation (Section 3.5.5), the objective function from Equation 3.38 (Section 3.5.3) is updated as follows:

$$\min_G \max_{D:||D||_L \leqslant 1} = \mathbb{E}_{x,y\sim p_T(x,y)}[D(x,y)] + \mathbb{E}_{z\sim p(z),y\sim p(y)}[D(G(z,y),y)], \tag{5.1}$$

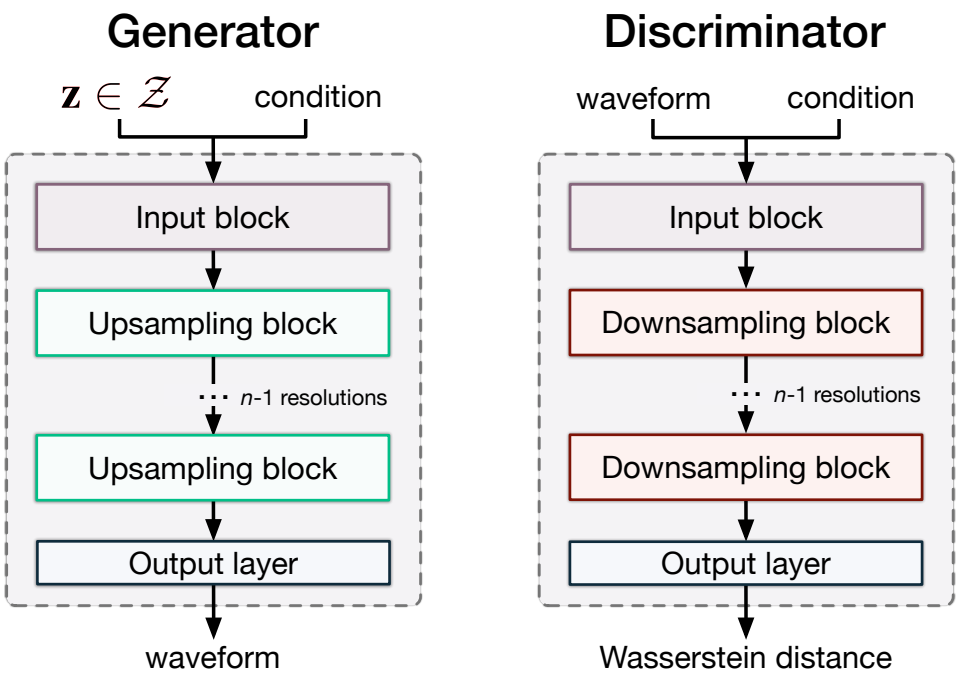

Figure 5.2: Overview of the proposed conditional drum synthesis system, consisting of a generator and discriminator network. The generator network takes a latent vector z and a drum class condition as input, and generates a waveform through a series of upsampling layers. The discriminator network, mirroring the generator, takes a waveform and condition as input. Through downsampling layers, features are extracted from the input, which are then used to compute the Wasserstein distance between the real and generated distributions.

in which $p(y)$ is the prior conditioning distribution. Conditioning the system on labels allows for the targeted generation of drum sounds from a specific category. Under an optimal discriminator, minimising the objective function with respect to the generator parameters minimises $W(p_g, p_d)$.

5.1.1 Implementation

The proposed approach to drum sample synthesis builds upon the architecture of WaveGAN (Donahue et al., 2019) but is designed specifically for the conditional audio generation of kick drum, snare drum and cymbal samples. In order to achieve improved quality and controllability, two new systems are developed: a conditional WaveGAN (Drysdale et al., 2020) and a conditional style-based WaveGAN (Drysdale et al., 2021).

Conditional WaveGAN

The conditional WaveGAN system extends the architecture of WaveGAN (Donahue et al., 2019) for the conditional audio generation of a variety of different drum sounds. Conditioning provides deterministic control over the generated output, enabling targeted synthesis that would allow producers to selectively create specific drum sounds. Figure 5.2 provides an overview of the conditional WaveGAN architecture consisting of a generator G (left) and discriminator D (right). The generator contains an input block which receives a conditioning variable y and a latent vector z. In order to learn a drum-specific encoding, y is passed through an embedding layer (Section 3.5.5) with a dimensionality $e = 3$, such that each

of the three drum classes are mapped to a different e-element vector representation that is learned by G ($e = 3$). Within the input block, the embedding layer and latent vector z are scaled to the initial size of the network using dense layers and then concatenated together. The concatenated result is then passed through a series of n upsampling blocks that systematically increase the resolution of the input. The upsampling process combines one-dimensional nearest neighbour upsampling (Section 3.5.4), in combination with a one-dimensional convolutional layer with a filter size of 16 and rectified linear unit (ReLU) activation function (Equation 3.4). These upsampling blocks are applied iteratively, increasing the number of audio samples by an upsampling factor of s at each step. The final waveform is derived from the output layer which uses a hyperbolic tangent (tanh) activation function (Equation 3.2).

The discriminator network D mirrors the architecture in G. D begins with an input block that receives an audio signal and conditioning variable y as input. In D, y is passed through an identical embedding layer to that in G and is scaled to the size of the input waveform using a dense layer and reshaping. This representation is then concatenated with the input waveform and passed through a series of n downsampling blocks. Each downsampling block consists of a convolutional layer with a stride of s and filter size of 16, and a leaky ReLU activation with $\alpha = 0.2$ (Equation 3.5). Thus, at each stage of the discriminator, the input waveform is decreased by a factor of s. Additionally, the downsampling blocks include a phase shuffle module (Donahue et al., 2019) that randomly perturbs the phase at each layer of *D*. Phase shuffle forces *D* to become invariant to the phase of the input waveform and is controlled by hyperparameter p that perturbs the phase of a layer's activations by -p to p samples. This is to help prevent an optimisation problem that can occur when *D* learns to reject generated audio with artifact frequencies that always occur at a particular phase (Donahue et al., 2019). The final layer of D is a dense layer with a linear activation function that outputs the authenticity of the input audio sample through the Wasserstein distance.

Conditional Style-based WaveGAN

The style-based architecture incorporates several modifications to the conditional WaveGAN generator network, resulting in an enhanced representation of factors of variation within the model. Specifically, the architecture improves the interpolation properties of the latent space, which enables smoother and more predictable transitions between different generative outputs. By introducing these modifications, the style-based architecture improves the ability of the generator network to generate high-quality audio with greater control over specific audio features.

Figure 5.3 provides an overview of the style-based generator, which is composed of a mapping network and an adapted generator network. In the original GAN formulation, output signal generation is controlled by feeding latent vector z with conditioning information to the input layer of the generator. G is decomposed into a series of L intermediate layers $G_1 ... G_L$. Thus, the first layer produces features $f_1 = G_1(z)$ and the subsequent layers produce features as a function of the output of the previous layer, such that $f_l = \hat{G}_l(z) \equiv G_l(f_{l-1})$. In a style-based GAN formalisation (Karras et al., 2019, 2020b), latent space $\mathcal{Z}$ is transformed into an intermediate space $\mathcal{W}$ using mapping network $M : \mathcal{Z} \rightarrow \mathcal{W}$. An intermediate latent vector w is derived by passing z and y to a mapping network M, such that $w = M(z, y)$, where $y \in \mathbb{R}^k$ is a one-hot encoding representing k drum instruments (i.e., kick drum, snare drum and hi-hat). To enhance the efficacy of the mapping network, the generator function G is modified to accept a fixed constant value as input, while intermediate latent vectors $w \in \mathcal{W}$, are provided to each upsampling layer $f_l = G_l(f_{l-1}, w)$.

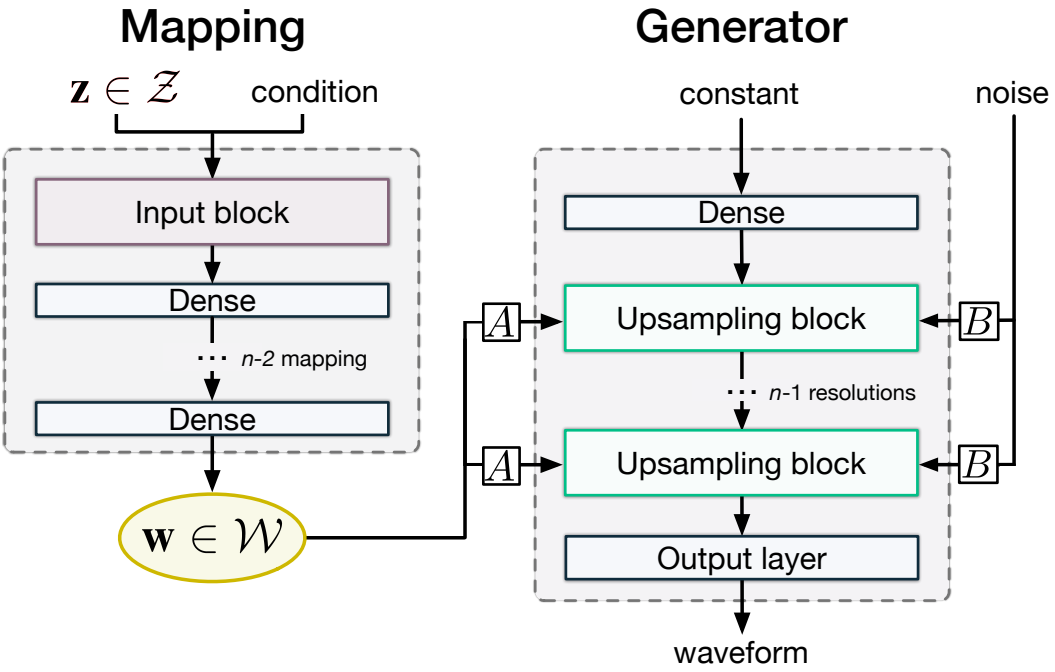

Figure 5.3: An overview of the style-based generator network consisting of a mapping network (left) and a generator (right).

The mapping network M is a conditional multilayer perceptron, that learns to create disentangled features that are integrated at each upsampling block of the generator network G through adaptive instance normalisation (AdaIN) (Karras et al., 2019). For the l^{th} hidden layer output activations AdaIN is defined as:

$$AdaIN(f_l, w_l) = w_l \frac{f_l - \mu(f_l)}{\sigma(f_l)} + w_l, \tag{5.2}$$

where, mean μ and standard deviation σ are computed across features for each channel and sample. The generator is trained to output audio waveforms given a latent vector w for each upsampling block with affine transform A. Additionally, Gaussian noise is added to each upsampling block using per-layer scaling factors B to introduce stochastic variation at each layer.

5.1.2 Network Training

In order to optimise the objective function (Equation 5.1) and reach an equilibrium, alternating updates between networks G and D are performed. At each training iteration, the parameters of network D are updated r times for each G parameter update. The proposed models aim to minimise the Wasserstein distance between the training data distribution and the generated data distribution. To enforce the Lipschitz constraint, a gradient penalty (GP) with a coefficient of λ is applied, as discussed in Section 3.5.3. Models are trained on an NVIDIA 2080ti GPU for $\sim$100000 iterations using the Adam optimiser (Kingma and Ba, 2015) with a learning rate $2\text{e}{-4}$, $\beta_1 = 0.5$, $\beta_2 = 0.99$, where each iteration takes a mini-batch of 64 examples. For the style-based generator, synthesis is performed by integrating an intermediate vector w at each upsampling block of G through AdaIN. In accordance with Karras et al. (2019), mixing regularisation is employed to prevent the network from assuming that adjacent

upsampling blocks are correlated. This is achieved by generating two intermediate vectors w_1 and w_2 by passing latent vectors z_1, z_2 through the mapping network M. During training, each upsampling block is provided with different intermediate vectors by randomly selecting either w_1 or w_2.

5.2 Evaluation Methodology

The following section presents the methodology used to evaluate the quality of generations produced by various configurations of the proposed GAN model for drum sample generation. In order to determine the optimal configuration for drum sample waveform generation, the conditional WaveGAN (`CW-GAN`) and conditional style-based WaveGAN (`CSW-GAN`) architectures, presented in Section 5.1.1, are evaluated. Additionally, an unconditional WaveGAN (`UW-GAN`) is included as a baseline, which is based on the original implementation of WaveGAN by Donahue et al. (2019). This model serves as a benchmark to compare the performance of the proposed architecture modifications. While WaveGAN was originally trained using a large latent dimensionality ($d_z = 100$), the proposed system aims to allow producers to interactively navigate a compact representation of drum samples. A smaller dimensionality offers benefits such as easier visualisation, control, and interpretability. In the context of SBEM production, this facilitates more intuitive browsing and fine-tuning of samples. However, it is essential to find a balance between maintaining high-quality generations and a compact representation when designing generative models for audio synthesis. For comparison, the `CW-GAN` and `CSW-GAN` architectures are trained and evaluated using latent dimensionality d_z values of 3, 16, 64, and 100. This allows for a thorough examination of how varying the latent dimensionality affects the quality and diversity of the generated drum samples.

5.2.1 Evaluation Data

For all experiments, networks are trained using raw audio waveforms. A dataset of oneshot drums samples $\mathcal{D}$ was complied from a wide variety of commercial sample libraries. Drum samples were categorised manually into k domains comprising kick, snare and cymbal samples. Each domain contains 3000 individual samples resulting in a total dataset size of 9000 samples. Prior to preprocessing, all samples in the dataset were mono 16-bit PCM audio files sampled at 44.1kHz, with a mean audio length of 18234 samples (i.e., 0.41s). In order to reduce computational requirements for the purpose of this evaluation, it was determined that the audio files would be downsampled from 44.1kHz to 16kHz. The training data audio length is selected to be the nearest power of two ($T = 8192$) to satisfy the symmetric structure of networks G and D. Each training sample is trimmed or zero-padded to ensure a constant length of T samples. All waveforms are normalised and a short linear fade of samples is applied to the start and end of each waveform to ensure that they consistently begin and end at 0 amplitude. For validation purposes, an additional 1000 drum samples (evenly distributed across the three classes) was collected as test data, resulting in a data split of 90% for training and 10% for testing.

5.2.2 Network Hyperparameters

In Table 5.1, a list of hyperparameter values used for training the architectures discussed in Section 5.1.1 is provided. Hyperparameters were initially selected based on previous research (Donahue et al., 2019)

Name	Value
Input data type	16-bit PCM (requantised to 32-bit float)
Audio sampling rate	16000
Model data type	32-bit float
Audio length	8192 samples (0.557 seconds)
Num channels (c)	1
Batch size (b)	64
Number of resolutions (n)	9
Stride (s)	2
Phase Shuffle (p)	2
Loss	Wasserstein-GP (Gulrajani et al., 2017)
GP coefficient (λ)	10
D updates per G (r)	3
Optimiser	Adam ($\alpha = 1e-4$, $\beta_1 = 0.5$, $\beta_2 = 0.9$)

Table 5.1: Network training hyperparameters for evaluation.

and then fine-tuned through a trial and error process to achieve a balance between generation quality, computational capacity, and training time. To investigate the relationship between the latent space dimensionality (d_z) and generation quality, each architecture is trained four times with d_z values of 3, 16, 64, 100. The input data type is mono ($c = 1$) 16-bit PCM audio sampled at 16kHz and requantised to 32-bit floating point data. The audio length (T) is set to 8192 samples—which is equivalent to 0.557 seconds at $16k$Hz sampling rate. To allow for the generation of T samples, each model consists of 9 resolutions with a scaling factor 2, thus at each stage of G and D, audio is upsampled or downsampled by a factor of 2 respectively. The model dimensionality parameter corresponds with how many trainable filters are used during training, and can be used to increase or decrease the size and complexity of the model. Models are trained using a gradient penalty coefficient ($\lambda = 10$) and D is updated 3 times per G update.

5.2.3 Evaluation Metrics

The primary objective of the proposed model is to generate novel samples with statistical properties that closely match those of the training data. In contrast to typical machine learning models, GANs lack a single objective function that can be used to evaluate their performance. GANs are trained with an adversarial loss that does not provide a direct measure of the quality of the generated samples, and the generations do not necessarily correspond to any specific, predefined ground truth. This makes it challenging to evaluate the quality of generated samples and compare between different GAN models. As a result, evaluating the quality of generated samples often requires generating a large number of examples and comparing their overall distributions. Manual human evaluation would be impractical and time-consuming due to the large number of examples involved. To address the challenge of evaluating the quality of generated samples in the absence of a predefined ground truth, several reference-free evaluation metrics have been proposed to compare the distribution of generated samples to that of the real data. Following (Donahue et al., 2019; Engel et al., 2019), three sets of performance measurements are used in the evaluation: inception score (IS), nearest neighbour distance (NND), and Fréchet audio

distance (FAD). Each metric provides a different insight to the performance of the models. The metrics were calculated on a set of 20000 generations produced by each model.

Inception Score

The inception score (IS) (Salimans et al., 2016) is an evaluation metric that has been proposed to evaluate the performance of GANs. IS computes the KL divergence between the conditional class distribution $p(y|x)$ and the marginal class distribution $p(y)$ over the generated data, and is defined as:

$$IS = exp(\mathbb{E}_{x \sim p_g} \mathcal{D}_{KL}(p(y|x) \parallel p(y))), \tag{5.3}$$

where $x \sim p_g$ indicates that x is a sample from p_g. Following (Donahue et al., 2019), the conditional class distribution $p(y|x)$ is obtained by applying a pre-trained audio classifier to the audio generations. The audio classifier is a CNN architecture similar to the square-filter AIRC model (Section 4.2.1), that is trained on the dataset of labelled drum sounds (kick, snare, cymbal) from Section 5.2.1. Audio is input into the classifier as log-scaled Mel spectrograms (Section 3.4.5) with 128 Mel-frequency bands. The Mel spectrograms are processed through four convolutional and pooling layers. The final layer is a fully-connected layer with three outputs and a softmax activation function, depicting the presence of each class as decimal probabilities. A train–validation–test split of $80\%/10\%/10\%$ is applied to the dataset and early stopping is performed on the minimum negative log-likelihood of the validation set. The classifier achieves 97% accuracy on the test set.

Given model scores $p(y|x)$, the IS is calculated over 20000 generated samples. The IS score seeks to capture two properties: audio quality, whether the generations be classified into a specific class, and audio diversity, whether a diverse range of different classes can be generated. A high IS score implies the generative model can generate distinct audio from each class (i.e., low entropy for conditional posteriors) and there is variety in the generator output (i.e., high entropy for the marginal over all classes). This is an indication that a model can produce generations that capture the semantic modes of the real data distribution. IS will penalise models whose generations cannot be classified into a single class with high confidence, as well as models whose examples belong to only a few of all the possible classes. Salimans et al. (2016) suggest, in their experiments, IS correlates well with the human judgment of image quality; however, IS fails to capture intra-class diversity, meaning that a class-conditional model that memorises one example per class will achieve a high IS (Barratt and Sharma, 2018).

Nearest Neighbour Comparisons

Nearest neighbour comparisons can serve as a useful tool to evaluate the quality of generative models by assessing the extent to which a high inception score is a result of either limited intra-class diversity or overfitting of the training data. Specifically, such comparisons allow for an assessment of the diversity and variability of generated samples within each class, thus providing a means of diagnosing mode collapse and other pathologies that may arise in the course of model training. Donahue et al. (2019) proposed two metrics to determine if a high inception score has been caused by either of these two undesirable cases. The metric $|D|_{\text{self}}$ quantifies the diversity of a set of 1000 examples by measuring the average Euclidean distance to their nearest neighbor within the set, excluding the example itself. A higher value of $|D|_{\text{self}}$ signifies greater diversity among the samples. On the other hand, $|D|_{\text{train}}$ measures the average Euclidean distance of a set of 1000 examples to their closest neighbor within the real training set. A distance of 0 indicates that the generative model only reproduces examples from the

training set. However, a high value of $|D|_{\text{train}}$ would suggest that the generative model has deviated from the original training objective. Hence, a balance must be achieved to ensure that the generated samples possess sufficient diversity while remaining faithful to the training set. As in IS, distances are calculated using Mel spectrograms. $|D|_{\text{self}}$ and $|D|_{\text{train}}$ are reported on relative to those of the real test set in Section 5.3.

Fréchet Audio Distance

The Fréchet Audio Distance (FAD) (Kilgour et al., 2019) has been widely adopted due to its sensitivity to small changes in the real distribution (i.e., noise), computational efficiency, and close correlation with human perception. In contrast to the inception score, FAD can detect intra-class diversity and penalise a model that is affected by mode collapse. The FAD compares embedding statistics generated on an evaluation set with embedding statistics generated on the training data. To compute the FAD, a pre-trained audio classification network (Hershey et al., 2017) is used to embed the real and generated audio into a lower-dimensional space. The real and generated audio embeddings are then summarised as multivariate Gaussian distributions by calculating the means and covariances. The Fréchet distance between the two distributions is then calculated as follows:

$$FAD = ||\mu_b - \mu_e||^2 + tr(\Sigma_e + \Sigma_b - 2\sqrt{\Sigma_b \Sigma_e}), \tag{5.4}$$

where (μ_e , Σ_e) and (μ_b , Σ_b) are the mean and covariances of the embedding of real and generated data respectively and tr is a trace of the matrix. A lower FAD signifies that the synthetic and real data distributions are more similar.

The audio classification network used for generating the embeddings is based on the VGG architecture (Simonyan and Zisserman, 2015) and is trained on a large dataset of YouTube videos with over 3000 classes (Abu-El-Haija et al., 2016).

5.3 Results and Discussion

This section presents the results of the evaluation, with a comprehensive discussion of the generation quality. This entails an examination of the various attributes of the generated audio, while also considering the potential applications and limitations associated with the proposed approach.

5.3.1 Evaluation Results

Table 5.2 presents the results of the three evaluation metrics: inception Score (IS), nearest neighbour distances (NND), and Fréchet audio distance (FAD). The table presents the results of the proposed generative models (`CW-GAN` and `CSW-GAN`) trained with different values of the latent dimensionality (d_z), along with the unconditional baseline model (`UW-GAN`) that was trained using $d_z = 100$. The metrics are computed on generated drum sounds produced by the different architecture configurations and compared with the real training and test data. Across most model configurations, the IS closely matches that of the real data, indicating that the generated audio samples are diverse and representative of the entire data distribution while still being identifiable as belonging to specific classes. Models trained with lower d_z values generally achieved lower IS, suggesting that higher d_z values may be necessary to capture

	Fréchet Audio Distance				*Inception Score*	*Nearest Neighbour*	
Experiment	Overall	Kick	Snare	Cymbal	Overall	$\lvert D\rvert_{\text{self}}$	$\lvert D\rvert_{\text{train}}$
Real (train)	0.00	0.00	0.00	0.00	2.93	0.97	0.00
Real (test)	0.04	0.05	0.10	0.11	2.86	1.00	1.00
UW-GAN	4.80	-	-	-	2.66	1.40	7.61
CW-GAN ($d_z = 100$)	2.74	3.98	3.99	1.55	2.73	1.21	5.31
CW-GAN ($d_z = 64$)	2.97	3.38	4.18	2.54	2.70	1.33	3.11
CW-GAN ($d_z = 16$)	4.53	5.08	5.90	3.66	2.72	1.47	7.66
CW-GAN ($d_z = 3$)	5.32	4.31	7.73	5.18	2.65	1.27	7.85
CSW-GAN ($d_z = 100$)	**1.48**	**2.06**	**1.88**	**1.41**	**2.80**	1.18	5.07
CSW-GAN ($d_z = 64$)	1.98	2.17	2.97	2.00	2.79	1.29	6.10
CSW-GAN ($d_z = 16$)	2.78	4.14	4.54	3.44	2.73	1.24	7.70
CSW-GAN ($d_z = 3$)	4.29	5.17	5.30	4.49	2.61	1.16	8.57

Table 5.2: Results for experiments comparing the real and generated data for different model configurations, where bold indicates the best scores for FAD and Inception Score. A higher inception score demonstrates that a model can produce generations that capture the semantic modes of the real data distribution. $\lvert D\rvert_{\text{self}}$ nearest neighbour values indicate intra-class diversity relative to the real test data. $\lvert D\rvert_{\text{train}}$ nearest neighbour indicates the distance between the training data and generated data. The Fréchet audio distance (FAD) is reported for the generative models under mixed conditions (*overall*), as well as intra-class FAD for the individual conditions of *kick*, *snare*, and *cymbal*. A lower FAD indicates that the generated and real data distributions are more similar.

the semantic modes of the real data distribution. All experiments produce $\lvert D\rvert_{\text{self}}$ and $\lvert D\rvert_{\text{train}}$ that is higher than the real test data, indicating that the high IS is not a result of mode collapse or overfitting to the training data. The $\lvert D\rvert_{\text{self}}$ values are reduced for models with lower d_z values, which indicates a correlation between the size of d_z and the amount of variety a model can generate. $\lvert D\rvert_{\text{train}}$ is also higher for models with a lower d_z values, indicating that generations deviate further from those in the real data when using a smaller latent dimensionality.

The overall FAD is reported for the generative models, as well as intra-class FAD for the individual conditions of kick, snare, and cymbal. The conditional models generally demonstrate a lower FAD compared to the unconditional baseline model, indicating higher quality in generating perceptually similar samples to the real data. These results reveal that the conditioning improved the ability of the model to learn the complexities of the training data distribution with respect to each class. Furthermore, the effectiveness of the conditions suggests that they can be a valuable tool for drum synthesis, allowing for the targeted generation of drum sounds from a specific category. The exception to this trend is observed in the conditional models trained using $d_z = 3$, which exhibit a higher FAD compared to the unconditional baseline model. This finding provides support for the hypothesis that the use of a low value for the latent dimensionality d_z may lead to lower performance due to the limited expressiveness of the latent space. Overall the best performing model, with scores highlighted in bold, is the style-based generator with a latent dimensionality of $d_z = 100$. The style-based generator architecture considerably improves the quality of generations and provides greater control over the synthesis process.

.

5.3.2 Generation Quality

The highest-performing models, which are the style-based and conditional models with $d_z = 100$, were scaled up by increasing the number of parameters and adding two additional layers to generate audio at a sample rate of 44.1 kHz. Audio examples for each model are provided on the accompanying webpage.[1] Alongside the perceptually-motivated evaluation metrics (Section 5.3.1), which allow for comparison between different models, informal listening tests were conducted to assess the generation quality of audio samples from each class. It is worth noting that subtle differences between some of the models would be challenging for human evaluation to consistently discern. In contrast, the perceptually-motivated evaluation metrics offer a more consistent and objective measure for comparing the performance of these models. While formal human evaluations could provide additional insights, the subjective nature of audio perception and the specific context of SBEM production make it challenging to design and conduct such evaluations. Nevertheless, the audio examples provided on the accompanying website offer a qualitative perspective on the generated audio, allowing readers to form their own interpretation and complement the quantitative evaluation metrics used in this study.

Incorporating class labels as a conditioning factor into the generative model results in an improvement in overall audio quality and removes overlap between different classes of audio samples. The style-based architecture produces generations of audio that are generally more pleasing to the ear, with the snare and cymbal sounds in particular exhibiting a higher level of realism and richness. Real snare drums and cymbals produce an inharmonic sound that is characterised by a broad frequency spectrum, with energy distributed across a wide range of frequencies. In electronic music production, synthesised white noise can be used to approximate the sound they produce. By introducing random noise at each layer of the style-based generator network, the model is able to learn the characteristics of more complex and realistic snare drums and cymbals, with subtle variations in tonal balance, decay, and resonance. Moreover, the noise introduces stochastic variation to the generations, which mirrors the natural variation and unpredictability present in acoustic drums and analogue synthesisers. This variation is important, as it means that no two sounds are exactly the same, adding depth and nuance to the synthesised audio. During inference, the amount of noise provided to each layer of the network can be adjusted without affecting the overall structure of the generations, allowing further fine-grained control over synthesis.

Figure 5.4 presents examples of two generated instances for each of kicks, snares, and cymbals. The figure demonstrates the diversity and richness of the synthesised drum sounds, showcasing system capacity to generate distinct instances of each drum type while maintaining their characteristic timbral qualities. Kick drums provide the essential low-frequency foundation in SBEM and can be characterised by an initial high-frequency sound that drops in pitch and amplitude over a short period of time (Section 2.3.1). The fundamental frequency of a kick drum typically ranges between 50–145Hz (Rossing et al., 2014). Upon examining the first pair of spectrograms in Figure 5.4, it becomes evident that the generated kick drums exhibit these characteristics. Both examples display a rapid decrease in frequency, with the loudest frequencies falling within a similar low-frequency range. On the other hand, snare drums can be characterised by sharp staccato sound and burst of high-frequency noise (Section 2.3.1). Typical fundamental frequency ranges of acoustic snare drums are between 100–200Hz (Owsinski, 2017). Similar traits are discernible in the spectrograms of the generated snare drum examples, featuring a prominent peak at approximately 200Hz, enveloped by a spectrum of broadband noise. Cymbals are distinctive

[1] `https://jake-drysdale.github.io/research.html`

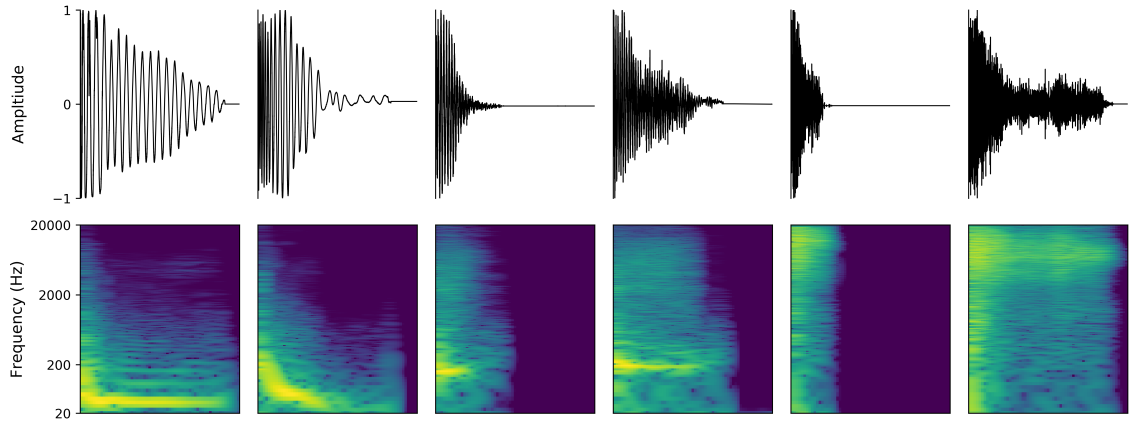

Figure 5.4: Example waveforms (top) and spectrograms (bottom) for two generated instances of kick, snare, and cymbal drum sounds, respectively.

in that they often lack a well-defined fundamental tone. Instead, they produce a broad spectrum of frequencies, contributing to a sustained wash of high-frequency overtones and complex resonances (Section 2.3.1). These characteristics are also evident in the spectrogram examples, showcasing a closed hi-hat and a crash cymbal.

As can be heard from the audio examples, some artifacts remain as a result of the upsampling layers in the generator network; however, their impact on the overall quality of the drum sounds is minimal. In most cases, these artifacts can be removed with simple post-processing techniques, such as amplitude fading or equalisation. Current methods for upsampling within neural audio synthesis models are known for introducing unwanted artifacts (Pons et al., 2021), and further research into more robust methods could prove essential in enhancing audio quality, especially when working with limited training datasets. While the quality of the generated audio can be enhanced through network expansion, additional training data, and prolonged training periods, the audio examples produced by the current system showcase its ability to synthesise audio using a lightweight network with a dataset size comparable to the personal sample collection of an SBEM producer. From a creative standpoint, the imperfections and artifacts found in the generations may be appealing to producers looking to achieve a low-fidelity sound, which is desirable some styles modern hip-hop (Winston and Saywood, 2019). An additional desirable attribute of the synthesised audio is its coherency, which gives the impression that all sounds originate from a single source. This effect is akin to recording drums in a specific room with a particular reverb time or using an analogue drum synthesiser with distinct electronic components. The coherence of the generated audio is highly desirable in music production, as producers frequently seek to achieve it through bus or group processing (Izotope, 2021). Combining sounds from disparate sources can often result in a disjointed sound, making the ability to generate coherent audio from a single model a valuable tool for music producers.

5.4 Latent Space Exploration

The evaluation has confirmed that the proposed conditional style-based GAN (`CSW-GAN`) is effective in generating a diverse range of high-quality drum samples. As mentioned in Section 2.1.3, techniques for transforming samples have largely remained unchanged since the early 1990s, constrained by the intrinsic

transformational potential tied to the timbral and rhythmic qualities of the source material. GANs offer a promising avenue for overcoming the limitations of traditional sample transformation techniques by providing new ways of manipulating samples. This section investigates various systematic approaches for interacting with the network and navigating the generative space. The aim is to exert influence over the generated content and showcase novel methods of sample manipulation. These approaches primarily involve creative exploration and subjective artistic choices, which may vary significantly between individuals. Nevertheless, these techniques provide new opportunities for timbral modifications and enable producers to discover and create unique samples using the system.

The proposed system allows for the generation of a specific type of drum sample through an input latent vector and conditional value. Continuous exploration and fine-tuning of the possible generations can be achieved by varying the values of the latent vector. The latent dimensions are akin to synthesis parameters, in that changing the values of the dimensions yields changes to the outcome of the generation. While a higher latent dimensionality can lead to better quality and diverse generated audio, it also poses a significant challenge for controllability due to its complexity and large number of dimensions. Specifically, there are far too many dimensions to effectively explore, demanding that the values for each dimension much be adjusted separately. A system with a smaller latent dimensionality would be more efficient in terms of interactively exploring and manipulating the sample generation space. For instance, models trained with a latent dimensionality of $d_z = 3$ only require navigation through three dimensions, making it easier to control and manipulate generations.

5.4.1 Interpolation

The proposed system learns to map points in the latent space to the generated waveforms. The structure of the latent space can be explored by interpolating between two known locations. Spherical interpolation (slerp) (Shoemake, 1985) is the most appropriate way of interpolating the latent space Z of GAN with a Gaussian prior (White, 2016). Slerp treats the interpolation as a curved path on an n-dimensional hypersphere, helping to avoid sampling from locations that are highly unlikely given the prior of the model. Interpolation can also be performed in the intermediate latent space W; however, linear interpolation is preferred as the vectors in W are not normalised in any way (Karras et al., 2019).

Figure 5.5 illustrates navigating through the latent space W using linear interpolation to transition between different kick drums, snares, and cymbals. For each drum class, a linear path consisting of 10 steps is created between two w vectors. The generator is then supplied with each w vector along the path, resulting in the output of a corresponding waveform. Changes in audio characteristics are continuous without abrupt variations during traversal of the latent space. Larger steps in the latent space are perceptually equivalent to smoothly mixing amplitudes between distinct drum sounds whereas smaller adjustments result in subtle changes in timbral characteristics. Minute adjustments can be made in any direction within the latent space to fine-tune a waveform or generate slight variations. Subtle variations in timbre have the potential to produce humanised variation in programmed drum sequences, thereby providing a more natural sensibility. Furthermore, sampling from improbable points in the w space can produce distorted and corrupted audio signals, which may present more abstract possibilities (Broad et al., 2020).

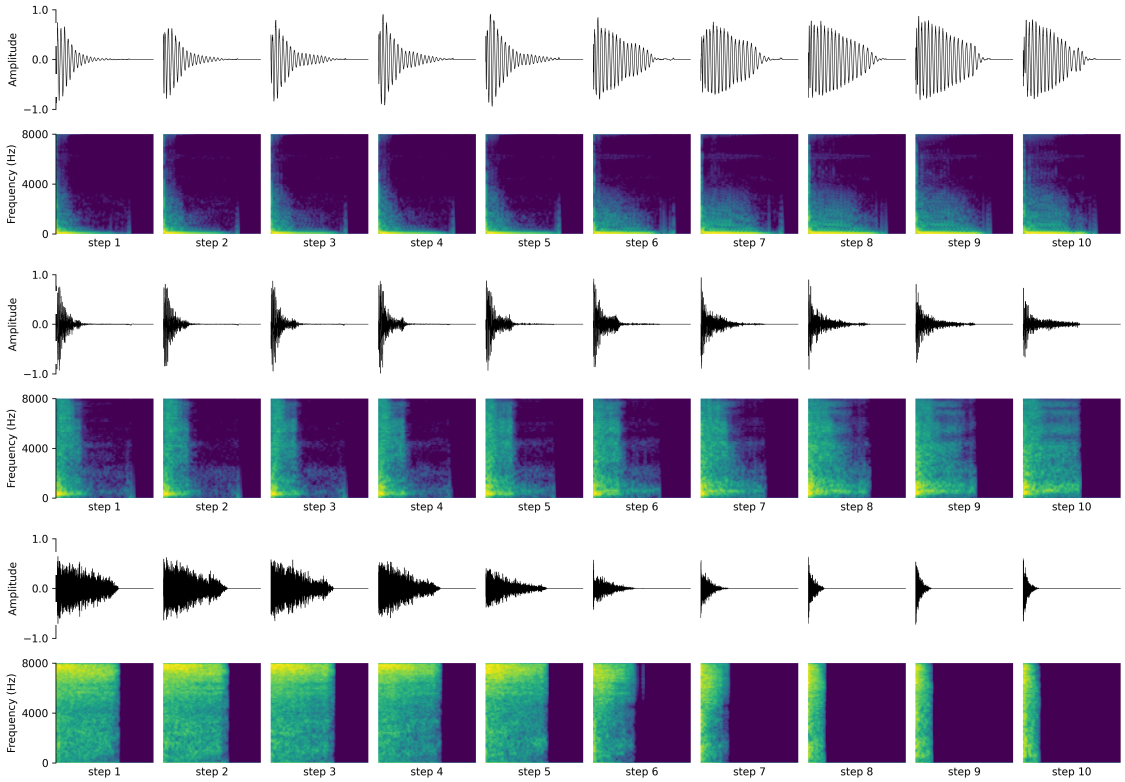

Figure 5.5: Latent space interpolation for kick, snare, and cymbal drum sounds, respectively. The figure displays waveform and spectrogram representations of 10 intermediate steps between two points in the latent space for each drum type.

5.4.2 Layer-wise Control

Generating audio with a specific latent vector w can be fine-tuned through layer-wise edits. This involves modifying only the w inputs within a selected range of layers, while keeping the other layers unchanged. This approach allows for more refined control over audio synthesis, particularly in drum sound generation, as it enables minor adjustments to the sound without altering its main features. For instance, if a certain sound is identified as desirable in the latent space, the earlier layers of the network can be frozen to preserve the essential characteristics of the sound, while the later layers can be adjusted to explore finer details. Combining this approach with the synthesis parameters obtained from dimensionality reduction (See Section 5.4.4) enhances control precision.

Figure 5.6 illustrates the impact of fixing different layers of the network while interpolating between two points in the latent space. As the number of fixed layers increases, the preserved characteristics of the drum become more pronounced. This offers insights into the features governed by distinct network layers and the interaction between latent space parameters and various network layers. For instance, step 8 in the last row of Figure 5.6 closely resembles the original generation (step 1) but incorporates some higher frequency transient content from step 8 in the first row, which corresponds to the second point in the latent space interpolation. This suggests that the high-frequency content is influenced by the later layers of the network. By assigning distinct latent vectors to different layers, this approach enables the generation of hybrid drum samples, seamlessly merging features from diverse sounds. In a music production context, one could manipulate the latent space to produce a desired sound and

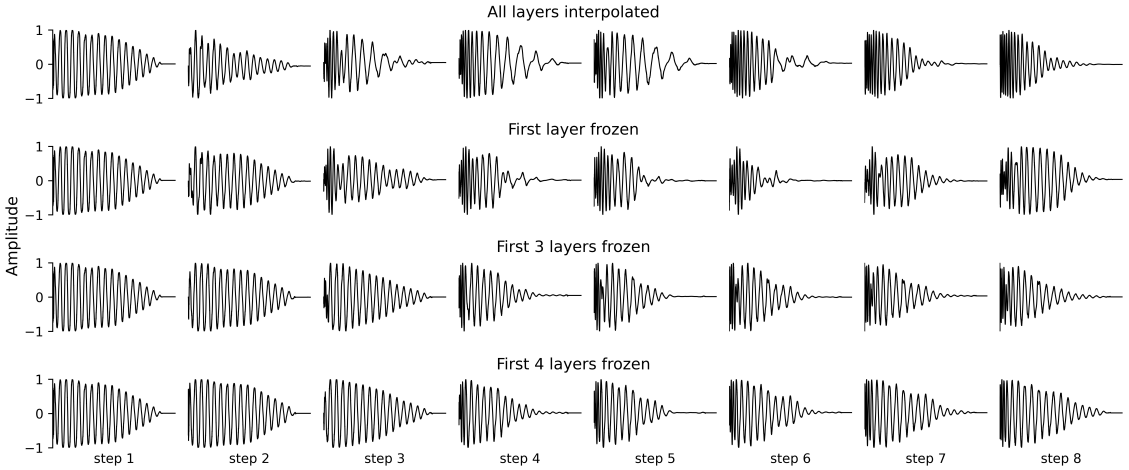

Figure 5.6: Visualising interpolation between two points in the latent space with varying fixed network layers. The figure displays four rows, each representing the effect of fixing an increasing number of layers during the interpolation process. Each row consists of 8 intermediate steps between the two points. As more layers are fixed, a larger portion of the original generation characteristics is retained in the generated output.

then automate the latent vectors supplied to the later layers of the generator network, creating dynamic variations while maintaining the fundamental characteristics. This fusion of distinct attributes results in unique and tailored samples, allowing for precise fine-tuning and honing in on desired sonic elements.

5.4.3 Arithmetic

Latent vector arithmetic is another technique that allows for control over the generated content by performing arithmetic operations within the latent space. This approach leverages the semantic relationships encoded in the latent space, leading to shared transformations across the generated samples. By adding, subtracting, or combining latent vectors in other ways, it is possible to create new samples that exhibit desired properties or characteristics. Two latent vectors, which generate two different drum samples, can be combined and provided to the generator network to create a hybrid of the two sounds. This technique is similar to the spectral *mutation* technique described by Zölzer et al. (2002), in which the magnitude and phase of one or two time-frequency representations are modified and combined to create a hybrid sound

Figure 5.7 demonstrates the effects of combining w vectors using various arithmetic operations. The first column displays the original w vectors, while the subsequent columns present the resulting waveforms obtained by combining the vectors in different ways. The combination of vectors can be controlled by adjusting scaling factors, for instance, by computing $(w1 * 0.3) + (w2 * 0.7)$. This opens up interesting creative possibilities for producers to explore when designing drum samples, as the latent vector values can be manipulated during a live coding performance (Collins et al., 2003) or automated within a DAW.

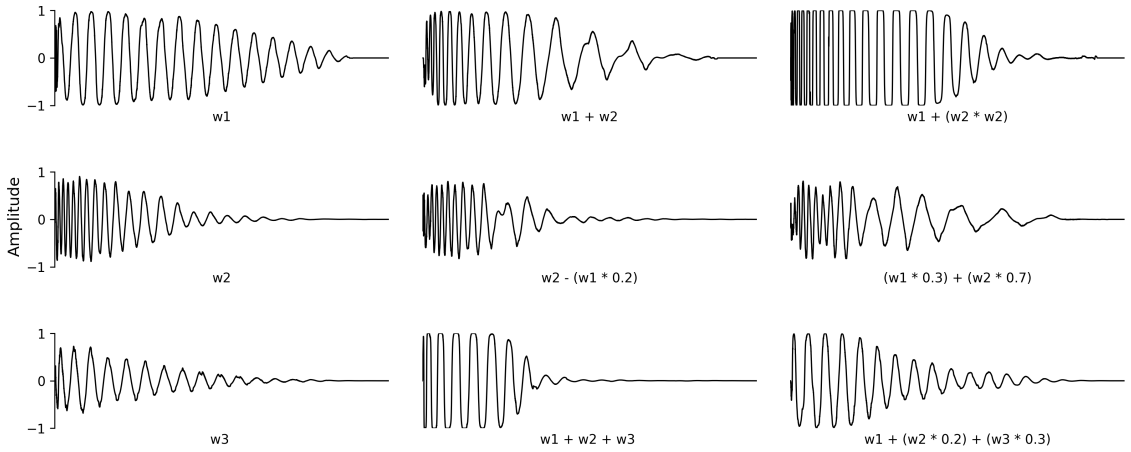

Figure 5.7: Latent space arithmetic operations applied to kick drums, illustrating the effects of combining different latent codes w through addition, subtraction, and multiplication

5.4.4 Dimensionality Reduction

Dimensionality reduction is a technique used in machine learning and data analysis to reduce the number of features or variables in a dataset while retaining as much of the relevant information as possible. The latent space Z is a distribution of vectors generated from a Gaussian distribution that does not inherently possess semantic meaning. Conversely, the intermediate space W is obtained by transforming the vectors from the latent space Z through a non-linear mapping function. This transformation enables the manipulation of the w vectors to control specific aspects of the generated audio. Dimensionality reduction techniques can be applied to W to reduce its high dimensionality and extract features that contribute to the variations in the generated audio. By reducing the dimensionality of W, users can more effectively explore the generation space and achieve greater control over the synthesised drum sounds.

Principal Component Analysis

Principal component analysis (PCA) is a well-established technique utilised within the field of data analysis for dimensionality reduction. PCA identifies a set of orthogonal axes, known as principal components, that successively capture the maximum variance in the data. The principal components are akin to new uncorrelated features that are ordered by their contribution to the total variance in the data. This process allows for the preservation of the most important information contained within the intermediate latent space $\mathcal{W}$ of the style-based generator, while also enhancing controllability for audio synthesis. PCA is applied within the intermediate latent space $\mathcal{W}$ to derive a set of coordinates that emphasise variation in timbre.

Following Härkönen et al. (2020), principal axes of $p(w)$ are identified with PCA. To this end, N random vectors are sampled from Z and the corresponding w values are computed using the mapping network M. PCA can be computed on the w values to obtain a basis V for W. After performing PCA, six principal components were found to explain 90% of the variance in the intermediate space W. The principal components can be utilised to control the generator by varying PCA coordinates scaled by control parameter p such that $w' = w + Vp$. Each p_i entry of p is a separate synthesis parameter that is initialised with zeros until modified by a user.

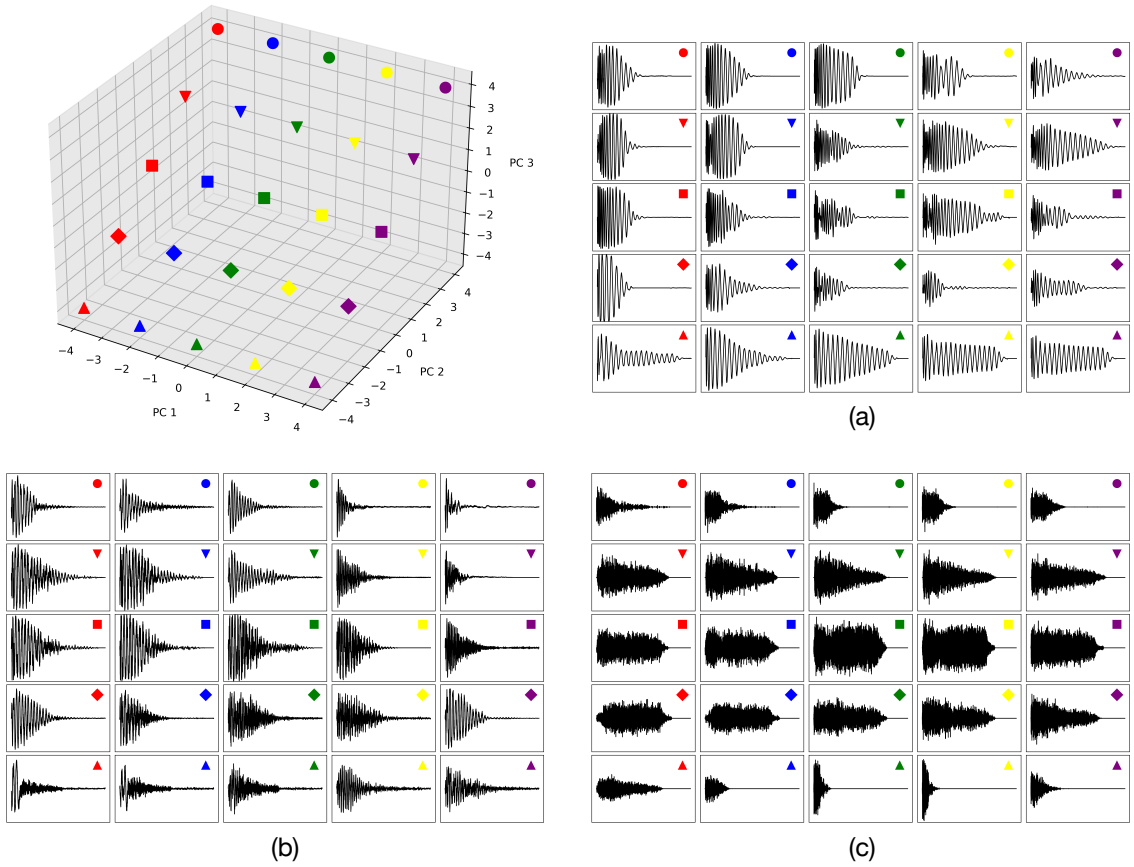

Figure 5.8: Exploration of the first three principal components (PCs) in the latent space for drum sound generation. The 3D scatter plot (top left) displays axes representing the direction of each PC. Drum sounds are generated at points along linear paths traversing the PC space, with paths color-coded to indicate their respective PCs. The resulting generated audio is visualised in corresponding plots for kick (a), snare (b), and cymbal (c) samples.

Figure 5.8 demonstrates the effects of manipulating the first three principal components of the w vector on the characteristics of kick (a), snare (b), and cymbal (c) samples. The original generations, with unmodified w vectors, occupy the centre and are labelled by a green square. The 3D scatter plot depicts the direction of each principal component, while the coloured shapes indicate the combination of directions used to manipulate the drum samples. By modifying the principal components, crucial aspects of the generation can be adjusted, including the attack, decay, and frequency content, as demonstrated in the figure. The visualisation reveals how manipulating the principal components can affect the characteristics of the generated samples, and provides insight into the underlying structure of the audio sample space.

Uniform Manifold Approximation and Projection

Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2018) is a manifold learning technique for dimensionality reduction. In contrast to PCA, which aims to preserve the pairwise distance structure among all data samples, other dimensionality reduction methods such as UMAP and t-SNE (van der Maaten and Hinton, 2008) emphasise the preservation of local distances over global distances. While PCA produces a linear transformation of the original data that is easier to interpret, UMAP and

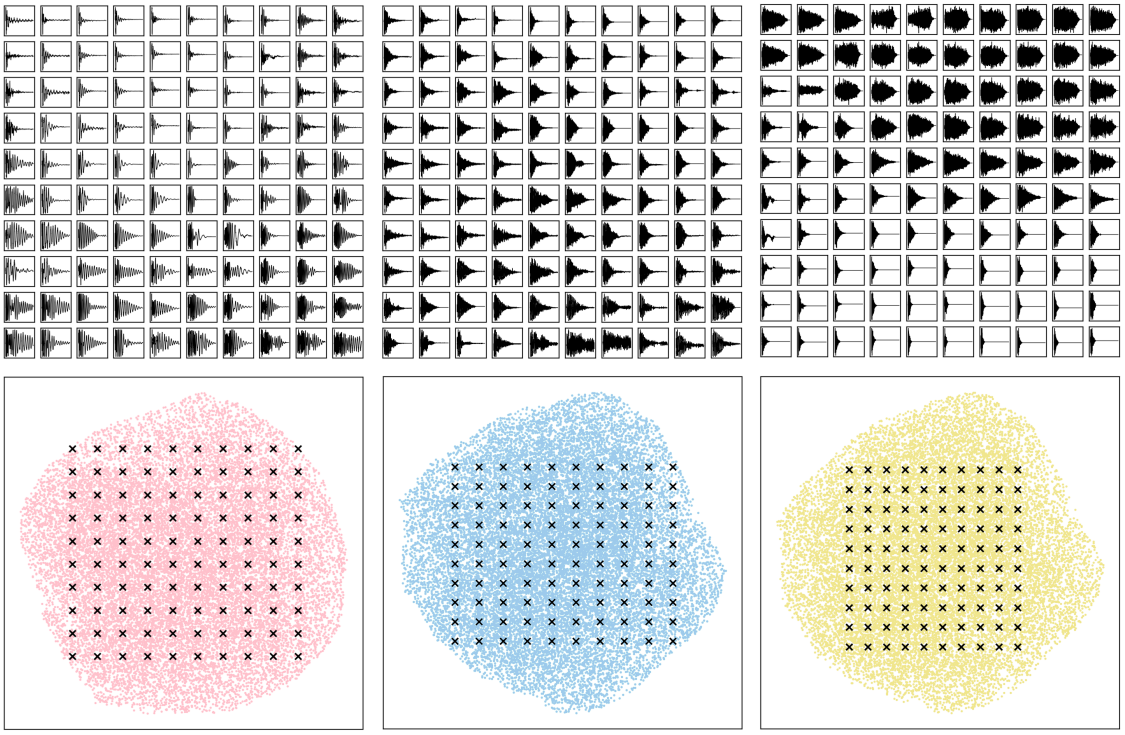

Figure 5.9: UMAP embeddings and waveform generations of drum samples. The top section features three individual UMAP embeddings for 20000 intermediate latent vectors of kick (left), snare (centre), and cymbal (right) drum samples. Black crosses within the embeddings mark the points used for generating the corresponding sounds, which are displayed as waveform generations in the lower section.

t-SNE are capable of generating low-dimensional representations of data that better capture non-linear relationships. UMAP has the added advantage of being faster computationally than t-SNE and is capable of preserving the distance between points that are far in the embedding space. Additionally, UMAP supports inverse transformations, in which high-dimensional data can be approximated given a location in the low-dimensional embedding space.

UMAP estimates a manifold by constructing a fuzzy topological representation of the high-dimensional data, which is then optimised to produce a low-dimensional embedding that preserves both the local and global structure of the data (McInnes et al., 2018). UMAP has several hyperparameters that impact the resulting embedding, including the dimensionality of the low-dimensional embedding space, the size of the neighbourhood used to estimate the manifold structure of the data, and the minimum distance apart that points are allowed to be. To demonstrate how UMAP can be used to control drum sample generation, a two-dimensional manifold mapping with 5 neighbours and a minimum distance of 0.2 is created from 20000 64-dimensional w vectors for individual classes kick, snare, and cymbal. The hyperparameters were determined through a series of preliminary experiments, resulting in a uniform and comprehensible space for navigation.

Figure 5.9 presents the resulting two-dimensional UMAP embeddnigs for each class of drum samples. By applying the inverse UMAP transform, higher-dimensional w vectors can be derived from the two-dimensional space, facilitating the generation of new drum samples. The black crosses in the figure highlight positions within the embedding space from which samples were generated. The lower portion of the figure provides examples of drum samples produced from these specific locations in the embeddings.

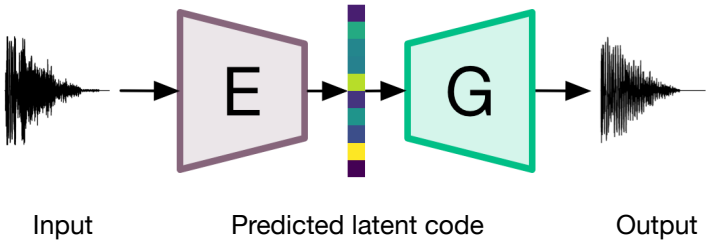

Figure 5.10: Overview of encoding an input into the latent space and regenerating a drum sample.

The examples demonstrate that the UMAP embeddings organise different drum generations according to their timbral characteristics in a coherent way. For instance, in the kick drum space, moving downwards along the y-axis produces kick drums with a longer decay, while moving along the x-axis affects the initial transients, with the right side having more detailed and prominent transients. Similar observations can be made for other types of drums as well, for example, the cymbal embedding effectively distributes different types of cymbals (e.g., hi-hats and crash cymbals).

5.4.5 Embedding Existing Audio into the Latent Space

Music producers often employ various techniques (e.g., layering, pitch-shifting, and time-stretching) to create variations of existing drum recordings (Section 2.2.3). To allow for the manipulation of existing audio material using the proposed system, a separate encoder is trained to embed a given waveform into the intermediate latent space of the pre-trained generator. By estimating the latent vector w for an existing sound, the input can be reconstructed by the generator model and variations can be made by modifying w using any of the aforementioned exploration techniques (i.e., interpolation, varying principal coordinates, and layer-wise editing).

Following recent advances in the image domain (Xia et al., 2023), an encoder network E is trained separately to embed a given waveform into the intermediate latent space of the pre-trained generator. Figure 5.10 illustrates encoding audio input and regenerating an output given the predicted latent vector. The predicted latent vectors are fed into the generator to synthesise drum sounds with similar characteristics to the input waveform. E replicates the architecture of the discriminator network D; however, the model is unconditional and its final dense layer has been modified to match the dimensionality of w, which has a size of $d_w = 100$. Using a dataset complied of 10000 drum sounds generated with pre-trained network G, E is trained to minimise the mean square error (MSE) between the ground truth latent vectors w and the predicted latent vectors $\hat{w}$, as expressed by the following formula:

$$\mathcal{L}_E = \text{MSE}(w, \hat{w}) = ||w - \hat{w}||^2 2. \tag{5.5}$$

A train–validation–test split of $80\%/10\%/10\%$ is applied to the generated dataset and early stopping is performed on the validation set. E is achieves 87% accuracy on the test set.

Figure 5.11 presents examples of beat-box (i.e., vocal percussion) drum sounds and their reconstruction using the proposed method. While the reconstructions may not be perfect, they share key characteristics with their counterparts, such as the amplitude envelope shape and frequency content. This approach enables synthesis control by using sounds as input, even for timbres not encountered during training.

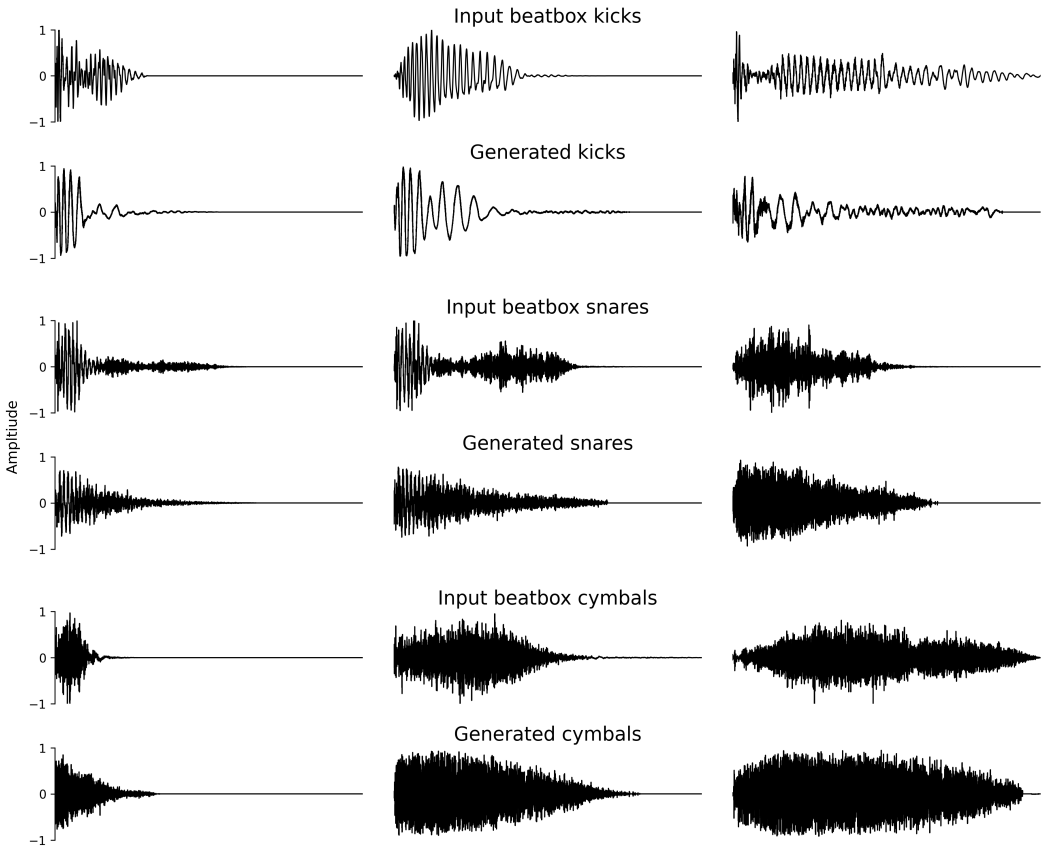

Figure 5.11: Beatbox drum sounds regenerated using the encoder and generator. Top row shows kick drum sounds, middle row shows snare drum sounds, and bottom row shows cymbal sounds. For each drum type, the input sound is mapped into the latent space using the encoder and then regenerated using the generator to create the output sound.

The encoder identifies generations in the latent space that exhibit shared characteristics with the input sounds, effectively creating a form of tone transfer. The proposed functionality has been evaluated by Nistal et al. (2022) and subsequently integrated into the DrumGAN VST.[2] This feature can provide music producers with greater flexibility and creativity in generating unique timbres by manipulating synthesis with input audio.

5.5 Additional Experiments

Building upon the evaluation in Section 5.3, which demonstrated that the system performs well when conditioned on distinct drum classes, this section outlines a series of supplementary experiments involving the use of different conditioning techniques within the proposed system. These additional experiments aim to further showcase the versatility and potential for customisation of the system. These conditioning approaches enable the system to learn additional features from the input data, resulting in synthesised sounds that are better aligned with the intended attributes. Specifically, the experiments focus on the effects of conditioning on the positions of $\frac{1}{16}$-note segments, enabling the generation of one-bar drum loops (Section 5.5.1), and multimodal recording parameters, allowing for the generation of snare

[2] `https://www.steinberg.net/vst-instruments/backbone`

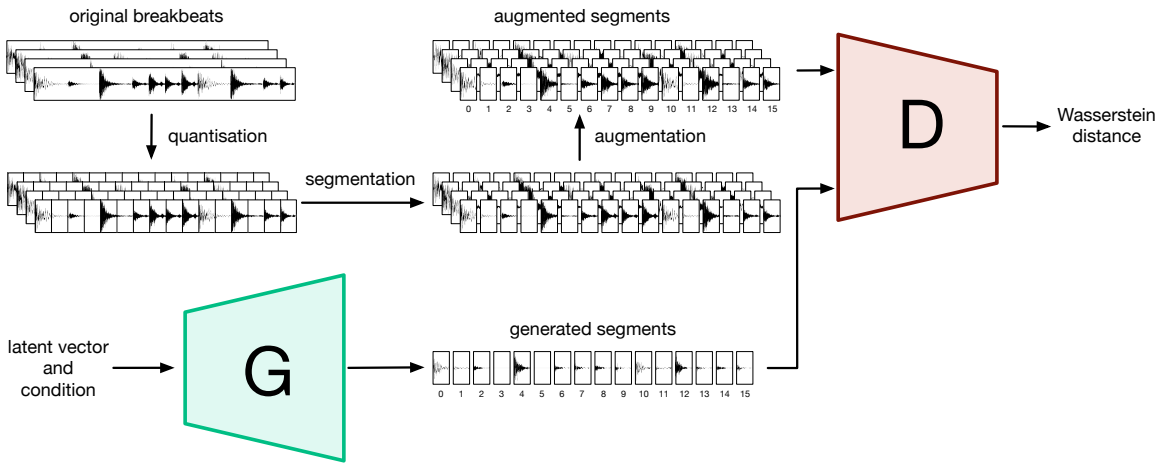

Figure 5.12: Overview of drum loop generation system.

drum sounds with targeted characteristics (Section 5.5.2). Furthermore, a prototype graphical user interface is presented, which integrates all the synthesis techniques outlined previously in this chapter. The outputs of these experiments provide valuable insights into the capabilities and limitations of the proposed system, and highlight potential directions for future research.

5.5.1 Generating Drum Loops

While synthesising one-shot drum samples has its applications, SBEM producers often prefer to work with longer instances of drum recordings, such as breakbeats (Section 2.1.2). This approach allows for the preservation of rhythmic properties, which can provide a more natural and cohesive feel in comparison to manually programmed drum patterns (Section 2.2.3). However, the implementation of GANs for generating audio waveforms with increased durations faces challenges due to the larger models and extended training times necessary to produce high-resolution audio. Despite the benefits of fast parallelised training and inference, the need for increased memory to load the model constrains the length of audio samples that can be generated. In order to generate longer instances of audio, such as breakbeats, one possible workaround is to segment the longer loop into smaller sections, each labelled according to its position, and use these labels as conditioning information for the system. By conditioning the system in this way, it can generate a sequence of smaller audio segments that can be concatenated to produce a longer audio sequence, effectively circumventing the memory limitations of the model.

Figure 5.12 provides an overview of the drum loop generation method. Labels are used to condition the system on an integer value, which determines the desired 16th-note segment. The first condition corresponds to the first $\frac{1}{16}$-note segment of each drum loop, the second condition to the second $\frac{1}{16}$-note segment, continuing in this manner for subsequent segments. Synthesis is controlled by an input latent vector and condition, enabling continuous exploration and interpolation of the generated waveforms at different timing positions. To train the system, twenty drum loops created from popular breakbeats, such as *Amen* (The Winstons, 1969), *Funky Drummer* (James Brown, 1970), *Think* (Lyn Collins 1972), and *Apache* (Incredible Bongo Band, 1973), were selected. Each drum loop is reduced to the length of one bar and quantised in Ableton Live using a 16th-note resolution. The quantised breakbeats were then sliced into $\frac{1}{16}$-note segments, resulting in an initial 20 examples per condition. To ensure consistent

training, the drum loops are required to be at a uniform tempo. Consequently, a tempo of 161.5 beats per minute was selected for this purpose. This tempo was chosen based on the sample rate of 44.1kHz, which results in each $\frac{1}{16}$-note segment consisting of 4096 samples—the nearest power of 2 that aligns with the symmetric structure of the generator and discriminator networks. To increase the size of the dataset, the individual drum segments were augmented using common music production techniques for manipulating breakbeats such as pitch shifting, re-sampling, and distortion (Section 2.2.3.

To generate a drum loop, the system generates a waveform for each of the 16 conditions using the same latent code for each condition. The system is reminiscent of a drum machine, empowering users to resequence and substitute audio at various locations. By traversing the latent space, a wide range of characteristics from different breakbeats can be explored. The interpolation of the latent space enables smooth transitions in the generated output, where large steps can be used to select different breakbeats and small steps to make subtle adjustments in timbre or rhythmic pattern. The provided audio examples for this experiment can be accessed through the corresponding webpage.[3] A technique commonly used by music producers is to alternate between multiple breakbeats over the course of a composition (Hockman, 2014). The unique feature of this system lies in its ability to facilitate seamless morphing between multiple breakbeats. This feature can be further leveraged to create mashups of different breakbeats using different latent codes for each condition. Specifically, the capacity of the system for resequencing and substituting audio at different locations enables the combination of multiple breakbeats, thus generating unique, hybrid drum loop variations.

The resulting system can be combined with the breakbeat detection system (Section 4.5), thus creating a production tool that can synthesise new drum sounds based on breakbeats detected in existing music. As an example, breakbeats that have been extracted from a collection of funk and jazz recordings could be utilised as training data for the generative model. To generate drum loops, the system can be conditioned on labels that indicate the position of the segmented breakbeats. For one-shot drum generation, an automatic drum transcription system (Southall et al., 2017) could be deployed to label the segments according to drum type (i.e., kick, snare, and cymbal). This pipeline presents a unique method for generating drum sounds that are consistent with the characteristics of a specified dataset of existing music.

5.5.2 Multimodal Conditioning

The proposed GAN architecture can be modified to generate data that exhibits diverse features or properties by conditioning it on input data labelled with multiple tags, each representing a unique condition. Given that sufficient labels are available, this approach enables further control over the generated output through various inputs and allows for the exploration of condition variations through the latent space. Such control is particularly beneficial for audio synthesis, as it provides more extensive control over the synthesis process. By incorporating multiple conditions, the generated audio can be customised to meet specific requirements, resulting in a more refined and adaptable output.

To support multiple conditions, the style-based generator (Section 5.1.1) is modified to accept multiple inputs. Each input is passed through a dedicated embedding layer, generating an individual encoding for every condition. The outputs from these layers are subsequently concatenated with the latent vector z, forming a unified representation that encapsulates both the latent features and condition-specific

[3] `https://jake-drysdale.github.io/research.html`

information. This composite representation is then fed through the mapping network M, yielding an intermediate latent vector w. This vector serves to control the synthesis process, enabling more refined and customisable data generation that takes into account the characteristics of multiple conditions.

To achieve this, a comprehensive dataset with multiple labels for each audio instance is required. Cheshire (2020) curated an extensive acoustic snare drum dataset (SDDS),[4] featuring multi-velocity recordings of ten different snare drums. Each drum was captured with 53 studio microphones, using various commercial dampening techniques. For this experiment, three recording characteristics are selected from the SDDS to condition the model: snare type, microphone position, and dampening. The snare type varies in dimensions, shell and head materials, as well as batter head tuning, reflecting the diverse construction of snare drums. The microphone position represents multiple placements that emulate various real-world recording techniques, capturing the nuances of different recording setups. Finally, the dampening characteristic refers to the dampening product, such as Big Fat Snare Drum, Snare Weight, Moongel, which are employed to alter the timbre of the snare. Snare batter head dampening is a common practice when recording drums (Seymour, 2010; D'Virgilio, 2014), which serves to modify the tonal characteristics of a snare by influencing aspects such as decay time, overtones, and frequency content.

A prototype graphical user interface (GUI)[5] has been developed to exhibit the potential of the system for generating drum samples with multiple conditions and the aforementioned synthesis techniques (Section 5.4), as showcased in Figure 5.13. This GUI integrates the `CSW-GAN` and facilitates interaction and manipulation of the network to create a wide range of customisable snare drums. The buttons on the top left facilitate the selection of the snare type condition (e.g., Maple, Premier, Yamaha). Below those buttons, the dampening condition (e.g., Big Fat Snare, Moon Gel, undampened) can be selected. Further below, the mixer section allows the amplitude of various microphone positions (e.g., overhead, top mic, bottom mic) to be manipulated using sliders. This is accomplished by generating five separate snare drums, each corresponding to one of the five microphone positions, and enabling the amplitude of each generation to be controlled by individual sliders. The generated outputs are then combined, and their overall amplitude can be adjusted using the master slider. The bottom set of sliders provides the ability to explore variations by navigating the latent space using the first five principal components, which correspond to the most substantial variance. Preliminary listening tests reveal that the first principal component controls velocity, while the others manage various characteristics related to the different microphones used during recording.

The system enables the generation of various snare drums by utilising conditions to fix specific characteristics of the snare drum recordings and explore variations by navigating the latent space. This approach emulates a mixing scenario where recordings from different microphones are present. The principal components simplify the navigation process by reducing the dimensionality of the latent space from 100 dimensions to 5 dimensions. These components influence characteristics of the drums that were not explicitly provided as conditions during training, such as velocity, microphone type, and other intricate features learned by the network. Although the current implementation conditions the system based on snare recording parameters available from the dataset, the system could also be adapted to accommodate other types of information for different applications. For instance, metadata from online sample libraries like Splice, which include tags related to genre, pitch, and duration, could be

[4] `http://dmtlab.bcu.ac.uk/matthewcheshire/audio/sdds`
[5] `https://jake-drysdale.github.io/tools.html`

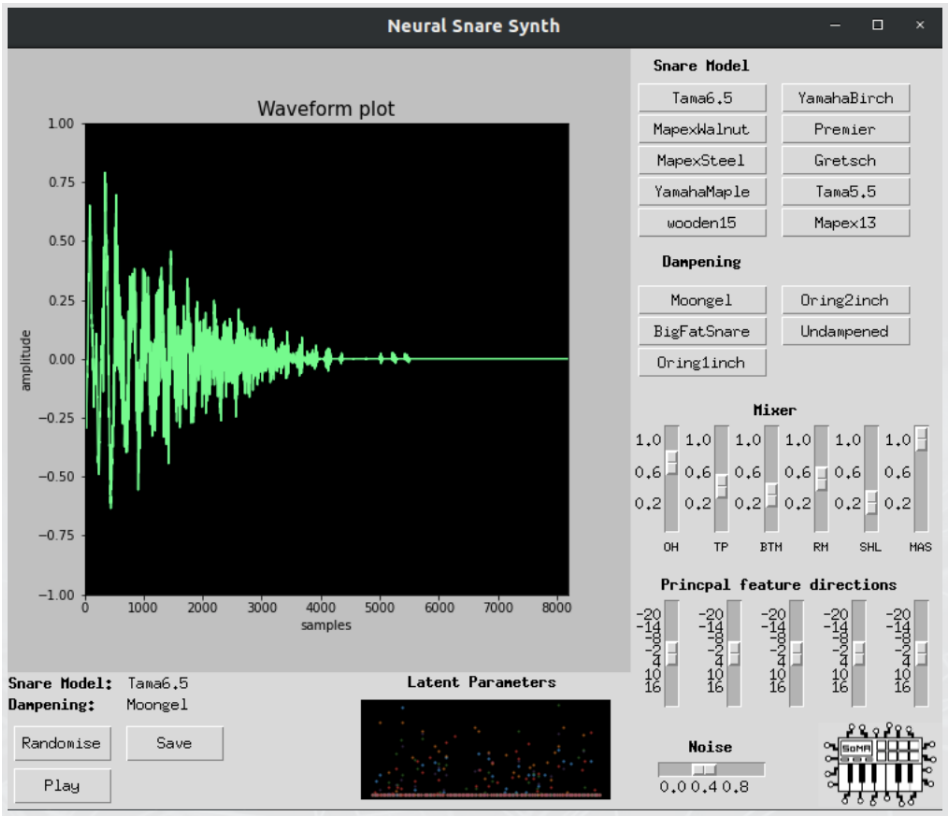

Figure 5.13: A prototype GUI for snare drum synthesis. Top left buttons select snare type condition, while below, the dampening condition can be chosen. The mixer section provides sliders for controlling amplitudes of various microphone positions. The bottom set of sliders allows for latent space navigation using the first five principal components.

used as conditioning inputs to cater to distinct needs. In cases where a collection of sounds lack such metadata, a automatic tagging system, such as the system proposed in Chapter 2, could be employed. This model could be trained on a large collection where metadata is available, and subsequently used to automatically label the collection of samples.

5.6 Chapter Summary

This chapter introduced a novel deep learning system developed to generate customisable drum sounds based on a sample collection. The system demonstrated the ability to seamlessly transition between generated sounds, thereby enhancing drum sample selection and manipulation. While existing tools for navigating sample collections have allowed sounds with similar characteristics to be positioned closely together in a space, the system presented in this chapter enables continuous exploration of intermediate samples, facilitating the discovery of sounds that possess combined characteristics of neighbouring samples. Through the implementation of label conditioning, individual layer control, and dimensionality reduction, the system facilitated efficient navigation of the latent space. The evaluation indicated that these modifications led to improvements in audio quality, as assessed by widely-used generative model evaluation metrics. These enhancements provided a more deterministic and adaptable approach to generating drum sounds while maintaining a high level of precision and versatility. Moreover, an

encoder model was developed to project incoming sounds onto the latent space, enabling synthesis to be controlled by audio input. Subsequent investigations demonstrated the versatility of the system when conditioned on various types of information, enabling different applications such as generating drum loops and adjusting snare drum recording parameters. A prototype graphic user interface was presented, demonstrating how the model and identified synthesis parameters could be integrated into a plugin format.

Overall, this chapter has provided insights into the performance of a deep generative model trained on a dataset that emulates the drum sound collection typically used by SBEM producers. The study demonstrates that the system can learn and generalise effectively from limited data, which makes it a promising tool for producers with their own unique sample collections. By substituting the training data with their personal sample collection, SBEM producers could tailor the system to their own styles and creative needs. The next chapter will conclude the thesis with a summary of the main findings of this and the previous chapters. This is followed by recommendations for future work to improve of the performances of the systems proposed in this thesis.

Chapter 6

Conclusions

Expounded in Chapters 1 and 2, music sampling has radically reshaped the landscape of music creation. It has brought a new level of versatility to the art of composition, enabling musicians to integrate a plethora of sounds from a wide variety of sources into their work. This revolutionary approach has given birth to an unprecedented age of innovation and creativity in the music industry. However, the modern era of digitised music access and production tools, coupled with extensive sample libraries, has introduced a fresh set of challenges. The volume and complexity of options available can become a hindrance, introducing a multitude of tasks that can disrupt the creative process. These complexities can divert the focus of musicians from pure creativity to the laborious management of digital resources. The primary focus of this thesis was the application of deep learning techniques to SBEM. The research introduced novel systems designed to foster a more intuitive and efficient interaction with music samples. These proposed methodologies were aimed towards enabling producers to more effectively utilise the vast resources available to them, and alleviating the technical burdens and complexities that have emerged in the current digital music landscape.

In the rapidly evolving landscape of SBEM production, innovative tools and techniques are invaluable. Several systems have been developed throughout this thesis to address evolving needs of producers and enhance their creative workflow. These systems provide practical solutions to common challenges faced by SBEM producers, ultimately streamlining the sample discovery process. A deep learning system for analysing music samples through automatic instrumentation role classification (AIRC) was introduced. The system is capable of automatically identifying the roles of various instruments within a given audio sample, serving as a valuable tool for organising sample collections, analysing the structure of SBEM, and identifying material suitable for sampling in existing recordings. By efficiently identifying instrumental roles associated with audio samples in large collections, the system significantly reduces the time and effort required for producers to curate and locate the ideal sound materials for their creative endeavors. Additionally, a deep generative model for synthesising drum samples was developed. By incorporating several architectural modifications and applying dimensionality reduction techniques to the latent space, a novel method for interacting with collections of drum samples was established. Furthermore, the system simplifies the synthesis procedure by harnessing a low-dimensional latent space, resulting in a more intuitive approach with fewer parameters to manage than conventional synthesisers. Evaluation results from the proposed systems indicated that the integrated deep learning techniques can effectively aid in the analysis and generation of electronic music samples. While these systems exhibit substantial potential, it is important to recognise that there is still scope for improvement and further investigation

within the wider realm of music analysis and generation. By continuing research and development in this area, not only will the capabilities of music analysis and generation systems be expanded, but artists and producers will be empowered with more advanced and intuitive tools for engaging with their creative work. In the following sections, a comprehensive summary of the key contents and contributions of the main chapters in this thesis is provided, along with a discussion of potential avenues for future research.

6.1 Summary

Chapter 4 introduced a novel deep learning system designed for automatic instrumentation role classification (AIRC), enabling the identification of samples based on their specific functions within SBEM (e.g., bass, chords, drums). The chapter began by presenting the methodology, implementation, and training details of various deep learning architecture configurations considered for AIRC. As the data used in AIRC encompassed a range of musical audio types, from tonal melodies to experimental sound effects, this motivated the exploration of architecture configurations suited for diverse sound classification tasks. Each configuration employed different front-end filter shapes to learn a representation from spectrograms and applied various pooling operations to derive the final predictions by summarising the information learnt by the network. The versatility of the system was demonstrated by conducting three distinct experiments, each representing a different SBEM production task: AIRC, loop activation transcription (LAT), and detecting samples in existing recordings.

The first experiment identified the optimal architecture configuration for AIRC. The models were trained and evaluated using the Freesound loop dataset (FSLD), a large public collection of loops and corresponding instrumentation role annotations extracted from the Freesound website. A novel data augmentation technique that incorporated typical music production techniques for mixing loops was introduced to address the limited coverage of multi-label annotations within the FSLD. The results from the AIRC evaluation revealed that the optimal configuration for this task was the square filter convolutional neural network with auto-pooling (`SF-CNN-AUTO`). This configuration significantly outperformed the baseline model, demonstrating substantially higher scores in both area under the receiver operating characteristic curve (ROC-AUC) and area under the precision-recall curve (PR-AUC).

In the second experiment, AIRC was employed to generate high-level summaries of SBEM arrangements by identifying the different instrumentation roles present throughout a composition. The performance of the proposed system was compared to previous approaches to loop activation transcription (LAT), a task that involves predicting the locations in which loops of a particular instrumentation role occur throughout a piece of music. When trained using the augmentation strategy, the AIRC system demonstrated competitive performance compared to the previous approaches for LAT, while offering a significantly faster runtime. A major advantage of the AIRC system is its capability to operate without depending on loops being precise duplicates, while simultaneously ensuring clear separation between instrumentation roles. This distinguished it from the previously proposed methods, which necessitate exact repetition of loops, constraining their adaptability in comparison. In order to demonstrate the ability of the AIRC system to accurately estimate instrumentation roles in authentic music scenarios, it was evaluated on a dataset of full-length SBEM compositions with annotated structural roles. The evaluation results revealed that AIRC is effective in transcribing instrumentation role activations in real-world SBEM compositions, highlighting the potential for practical applications in music production and analysis of SBEM compositions.

In the third experiment, a novel approach for the automatic retrieval of samples from existing recordings was introduced, utilising the proposed AIRC system. The proposed architecture configurations were evaluated in their ability to identify the location of breakbeats in a dataset of funk, soul, and jazz recordings. Despite being trained on data sourced from a community-based collection of user-uploaded loops and samples (FSLD), the models were successfully applied to fully produced music recordings from various genres and styles. The results demonstrated the potential of the AIRC system for identifying *sample-able* material from large collections of existing music.

In Chapter 5, a novel deep learning system for synthesising drum samples was introduced. The system serves as a method for creating new, highly controllable samples from a collection, offering the ability to seamlessly morph between generations of these samples. The chapter discussed the implementation details of a conditional generative adversarial network (GAN) architecture for drum sample synthesis and the procedures involved in the training process. It also proposed modifications to the generator network that enhance interactions with the system and evaluated the impact of these components with perceptually motivated evaluation metrics. The conditional style-based WaveGAN (`CSW-GAN`) architecture emerged as the best performing model, demonstrating substantial enhancements in audio quality. Compared to the baseline model, it achieved improved scores in terms of Inception score (IS) and Fréchet audio distance (FAD). Furthermore, the chapter explored an analysis of the attributes of the generated audio, examining potential applications and limitations of the system. Techniques for engaging with the system and investigating the generation space were proposed, aiming to achieve more deterministic control over the output. This involved exploring the latent space of the `CSW-GAN` through an unsupervised process using a range of dimensionality reduction techniques to identify synthesis parameters. Moreover, an encoder model was developed to project incoming sounds onto the latent space, enabling synthesis to be controlled by arbitrary sounds from various sources. Subsequent investigations demonstrated the versatility of the system when conditioned on various types of information, enabling different applications such as generating drum loops and adjusting snare drum recording parameters. A prototype graphic user interface was presented, demonstrating how the model and identified synthesis parameters could be integrated into a plugin format.

6.2 Contributions

As highlighted in Chapter 2, historical constraints associated with memory, financial resources, and a limited selection of tools made music production less accessible. Nevertheless, these very limitations often encouraged producers to refine their skills with specific tools and techniques, thereby fostering creativity and innovation. In contrast, modern advancements have alleviated many of these constraints, but the vast array of options can sometimes impede creativity by overwhelming producers with an abundance of audio material or intricate production techniques. Through a series of experiments, this thesis demonstrated the development and implementation of novel deep learning tools tailored specifically for SBEM. By providing efficient and scalable methods, these tools would enable producers to concentrate on their creative tasks, rather than spending time laboriously organising their collections to find the desired sounds while in a creative flow.

The first significant contribution of this thesis is the design and optimisation of a deep learning system capable of analysing the structural role a sample occupies within a SBEM composition through AIRC (Chapter 4). The system excels in determining the instrumentation roles of electronic music samples,

whether these represent singular roles—such as chords, melody, bass, drums, and sound effects—or a combination thereof. Evaluation results highlighted the efficacy of the proposed system in labelling collections of samples, analysing the structure of SBEM, and identifying samples within existing music. Additionally, a novel data augmentation technique was introduced, utilising SBEM production techniques to combine samples. This technique enhanced the representation of samples with multiple roles and demonstrated improved performance when transcribing the activations of loops in the Artificial dataset.

The proposed system presents substantial potential for the automatic labelling of large, unstructured sample collections. This could prove beneficial for online platforms like Splice or Freesound, which currently operate based on manual labelling procedures. It is particularly advantageous in the context of user-created sample collections, given its ability to process samples originating from an assortment of sources, such as vinyl records and digital recordings. Moreover, such collections may contain a vast number of audio files that would necessitate considerable human effort to organise manually.

Utilising the AIRC system, the structure of existing recordings can be analysed in the form of an instrumentation role activation map (IRAM), yielding valuable insights into arrangement and composition, or identifying sections suitable for sampling. This presents a powerful solution for deconstructing existing recordings into their constituent elements, including instrumentation roles and their corresponding locations, with numerous potential applications in music production and performance. The system can be employed to visualise the fundamental structure of a song, automatically generate tracks using a set of loops and a reference song, provide visual cues for DJs to anticipate upcoming song events, or support MIR systems that rely on structural information, such as automatic DJ systems and music mashups.

Additionally, the AIRC system can be used to detect material suitable for sampling in existing music, offering SBEM producers an effective tool for uncovering unique elements to incorporate into their own creative work. The system offers an automated approach to crate digging, and could serve as an invaluable tool for artists to quickly identify potential samples within extensive digital recording collections. The system could be utilised with an SBEM producer's personal collection of tracks across various genres, generating a new sample library from the existing recordings. Furthermore, the thresholds can be fine-tuned to be more or less stringent, which can be particularly useful for narrowing down potential candidates in larger collections or increasing the pool of candidates in smaller collections.

The second key contribution of this thesis is the development of a deep generative model for synthesising drum samples, designed to emulate the data on which it is trained, while simultaneously offering the capability to discover intermediate samples between existing ones. While methods for organising similar drum samples based on perceptual audio features already exist, the proposed approach maps a collection of samples to a compact latent space. This enables the generation of new samples by leveraging the statistical properties of the training data, offering the ability to seamlessly morph and interpolate between generations. Evaluation results highlighted the capacity of the proposed model to generate a diverse range of class-specific drum sounds, and exploration of the latent space revealed that the system can serve as a music production tool that fosters creative sound experimentation. Through the application of dimensionality reduction, layer-wise editing, latent space interpolation, and encoding incoming sounds into the latent space, the ability to generate unique and customisable drum sounds with a high degree of precision and control is demonstrated. The system can also be conditioned on various types of descriptive information, enabling numerous applications such as generating drum loops and adjusting snare drum recording parameters. Additionally, a prototype graphical user interface was developed, showcasing how interactions with the model through the conditions and derived synthesis

parameters can be seamlessly integrated into an audio plugin. Open-source implementations of the systems are provided on the accompanying webpages, complete with pre-trained weights, training data, and command-line interfaces for inference and training the system on new data.[1][2]

Although the primary focus of this thesis revolved around SBEM, the modular deep learning framework can be easily adapted for other tasks related to MIR and audio generation. By supplying alternative training data and labels, the presented systems can be tailored to accommodate various scenarios and individual user requirements. The aspiration is that these systems will enrich the growing body of research dedicated to supporting music production, particularly for producers who skillfully reinterpret music from the past.

.

6.3 Future Work

The research presented in this thesis has investigated the potential of deep learning to assist in SBEM production; however, there are still numerous directions to be explored in order to enhance the presented systems. This section discusses some possible directions for future work.

Efficiency and Sustainability in Deep Learning Systems

Over time, the constraints on memory and computational resources have continuously evolved. In the past, limitations were primarily imposed by the capacity of hardware samplers. Today, however, the quality and duration of audio generated by deep learning systems are largely dictated by computing power and GPU memory. While training with larger amounts of data and models can enhance audio quality and duration, it also comes with the cost of increased energy consumption and carbon footprint—a growing concern in the era of deep learning. Therefore, a potential direction for future work could involve the design of more efficient models that require fewer resources. Such an approach would not only help in reducing energy consumption but would also make these systems more accessible for devices with limited computational capacity such as mobile phones and embedded systems, including Eurorack modules or Raspberry Pi devices.

Training Data

The performance of deep learning systems presented in Chapters 4 and 5 are largely dependent on the quality and volume of the data used during training. This underscores the importance of careful curation of training data as one of the most effective strategies to improve future systems. Traditional methods of data collection and human annotation can often be labour-intensive and time-consuming. Therefore, an alternative approach could involve leveraging information from readily available online databases, which often contain a wealth of pre-labelled and categorised data. For instance, Splice offers a vast library of high-quality audio samples labelled with tags, which could be used to train deep learning models for sample classification and generation. However, to ethically and legally utilise this data, it is necessary to acquire the appropriate permissions or licenses. Similarly, WhoSampled offers a unique resource by cataloging instances of music sampling, complete with timestamps. This detailed database provides invaluable insights into the use and context of samples within a track. Such information could be instrumental in enhancing the performance of sample detection systems.

[1] `https://github.com/SoMA-group/style-drumsynth`
[2] `https://jake-drysdale.github.io/tools.html`

Data Augmentation

Although the augmentation technique presented in Chapter 4 incorporates features such as key matching and time-stretching to enhance loop combination, human producers might still have a superior ability to discern the best loop pairings. Future research could aim to refine this technique to better mimic the methods producers employ when combining loops. For instance, introducing equalisation and balancing amplitude levels of combined loops could potentially improve system performance. Furthermore, a model could be employed to access the compatibility of loops before combining them (Chen et al., 2020). These enhancements would be particularly advantageous when training models on other datasets that may lack representations of samples with multiple roles.

Data augmentation could further enhance the systems presented in Chapter 5, particularly when training on the personal drum sample collection of a music producer, which might be insufficient in size. Training GANs with limited data often results in discriminator overfitting, leading to training divergence. Karras et al. (2020a) proposed an adaptive discriminator augmentation mechanism that markedly stabilises training under data-limited conditions for images. This technique holds the potential to be adapted for audio data, facilitating training even on limited datasets.

Transfer Learning

Another compelling direction for future research involves exploring the potential of transfer learning. This approach involves applying the knowledge acquired from solving one task to a related task. While deep generative models typically require large volumes of data for effective training, transfer learning could provide an avenue for models to be fine-tuned using smaller datasets. This method can save significant computational resources and time, making it an efficient strategy for improving the performance of these systems. This approach could enable a balance between the quality of the audio generations and the aesthetic preferences of the user. For instance, if a producer has a specific style or genre they prefer, the system presented in Chapter 5 could be fine-tuned using a subset of data that aligns with this preference. This way, the output of the model would be more likely to match the unique style of producer, leading to a more personalised user experience.

Improved Control and Creative Interactions

In Chapter 5, the deep generative model demonstrated high performance when conditioned on a variety of drum types. The additional experiments implemented new strategies, such as position-based drum segment conditioning and multimodal snare generation under multiple conditions. Given appropriate labelled data, the conditioning possibilities for the system are endless. Future research could explore conditioning strategies based on specific drum types, such as tom-toms, claps, ride cymbals, and bongos; MIDI information like drum patterns and velocity; or variations in recording techniques, including microphone positioning, drum tuning, and room reverberation. Furthermore, exploring methods to dissociate and manipulate the underlying factors of data variation (i.e., latent space disentanglement) would be worthwhile. When combined with conditioning, this could empower users to maintain certain characteristics while continuously exploring other factors of variation.

Another promising avenue for exploration involves developing methods that allow generative models to yield novel and creative outputs that diverge from their training data. Broad et al. (2021a) describe several techniques for active divergence in the image domain, which could potentially be adapted for audio data. These approaches could significantly expand the possibilities for creative manipulation of samples, a crucial aspect in genres such as hardcore, jungle, and drum and bass.

Improved Evaluation Metrics for DGMs

Evaluating generative models remains a complex issue, particularly for GANs, which lack paired inputs and outputs and require reference-free metrics. The evaluation metrics used in Chapter 5, such as the inception score and Frechét audio distance, are currently standard in the field. However, these only provide a general overview of system performance. Future work could focus on developing new metrics that assess more specific audio characteristics in different DGM-based systems, such as timbre, pitch, and amplitude envelopes. Moreover, the development of metrics to assess continuity within the latent space could prove beneficial, facilitating the monitoring of smoothness and disentanglement therein. While such a metric has been established in the image domain (Karras et al., 2021), adapting it for audio information would necessitate significant modifications.

User Studies

The evaluations conducted in Chapters 4 and 5 have yielded important insights into both the capabilities and limitations of the proposed deep learning systems within the scope of SBEM production. These insights serve as a foundation for the further development and refinement of these systems. For future work, it would be beneficial to explore methodologies for conducting user studies. Such studies could concentrate on discerning how users interact with the deep learning systems, pinpointing any challenges or obstacles they encounter, and collecting feedback on their overall experience. The insights gleaned from such studies could then be used to refine the systems and better integrate them into user workflows. By tailoring the deep learning systems to align more closely with the needs and preferences of users, it is anticipated that user interactions and overall effectiveness could be significantly enhanced.

Audio Plug-ins

Another direction for future work is to embed the system presented in Chapter 5 into an audio plug-in format (e.g., a virtual studio instrument) that can be utilised within a DAW. A prototype GUI was presented to demonstrate how the proposed approaches of interacting with the system could be integrated into an audio plug-in; however, future work could evaluate different interfaces with established SBEM producers to inform and improve the breadth of the design goals. Furthermore, the experimental dataset used for training could be replaced with the personal sample collections of SBEM producers and custom tags could be defined for conditioning.

6.4 Final Thoughts

This project originated from my personal curiosity to explore innovative methods for automating and enhancing labor-intensive and monotonous processes encountered when creating music with samples. From my own experience, producing electronic music and working with samples is often an improvised process. Very rarely do I have an exact idea in our mind of what I want to create; usually, it's just a feeling or a source of inspiration (e.g., other music, a film, a life event), a set of tools, and a drive to experiment with sound. With the multitude of music production techniques, composition methods, sample libraries, and other musical resources available online, creating high-quality, original music can be challenging. Through the continued development of new tools, I envision alleviating the technical burden and automating laborious tasks, making the process of producing electronic music more intuitive and allowing producers to improvise with greater ease.

Machine learning algorithms are often designed and trained to flawlessly accomplish a specific task. However, one of the distinct aspects of music is that there is no definitive right answer, leaving it up to the producer to determine what resonates with their creative vision. While there may not be a single correct approach, producers often adhere to certain rules or workflows that align with the genre or style they are expressing themselves through. These established conventions provide a sense of familiarity and consistency, which can be particularly appealing to listeners, especially those on the dance floor who seek to connect with the music in a visceral way. Balancing the desire for innovation with the need for familiarity is an ongoing challenge for music producers, as they strive to create works that resonate with audiences. By leveraging machine learning techniques that can analyse and understand these intricate patterns and structures, producers may be better equipped to navigate this delicate balance and create compositions that are both engaging and innovative.

Despite speculations about AI potentially dominating a large portion of the creative industry, I hope it will instead expand the scope of creative possibilities and establish unprecedented avenues for musical expression. Consequently, the process of music production may undergo a significant transformation, leading us to encounter ways of creating music that were once thought inconceivable. Owing to the vast array of passionate and skilled musicians and producers, the limits of creative potential will persistently be challenged and expanded. As technology and AI increasingly intertwine with the music production process, artists will gain a greater capacity to push the boundaries of their craft, ushering in continued growth and evolution in the music domain. Much like the revolutionary impact of digital sampling capabilities on music creation, AI and deep learning could similarly redefine the methods of crafting and consuming music. My hope is that this work will inspire future contributions, fostering an enhanced understanding of music and offering greater creative control in music production.

Bibliography

Abdoli, S., Cardinal, P. and Koerich, A. L. (2019), End-to-End Environmental Sound Classification Using a 1D Convolutional Neural Network, in *Expert Systems with Applications, 136(1)*, pp. 252–63. (cit. on p. 45.)

Ableton (2012), DJ Olive: The Audio Janitor Takes Live Dubwise, Accessed 3 January 2023, `https://www.ableton.com/en/pages/artists/dj_olive`. (cit. on p. 83.)

Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B. and Vijayanarasimhan, S. (2016), YouTube-8M: A Large-Scale Video Classification Benchmark, in *CoRR abs/2006.00751*. (cit. on p. 98.)

Ahmed, A., Benford, S. and Crabtree, A. (2012), Digging in the Crates: An Ethnographic Study of DJs' Work, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, *Austin, Texas, USA*, pp. 1805–14. (cit. on pp. 18 and 19.)

Akata, Z., Perronnin, F., Harchaoui, Z. and Schmid, C. (2015), Label-embedding for Image Classification, in *IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(7)*, pp. 1425–38. (cit. on p. 63.)

Alain, G., Chevalier-Boisvert, M., Osterrath, F. and Piche-Taillefer, R. (2020), DeepDrummer: Generating Drum Loops using Deep Learning and a Human in the Loop, in *Proceedings of the Joint Conference on AI Music Creativity*, pp. 81–91. (cit. on p. 17.)

Aljanaki, A., Soleymani, M., Wiering, F., Veltkamp, R., Larson, M., Ionescu, B., Anguera, X. et al. (2014), A Multimodal Approach to Drop Detection in Electronic Dance Music, in *MediaEval Multimedia Benchmark Workshop*, *Barcelona, Spain*. (cit. on p. 33.)

Andersen, K. and Knees, P. (2016), Conversations with Expert Users in Music Retrieval and Research Challenges for Creative MIR, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *New York City, USA*, pp. 122–28. (cit. on pp. 3, 4, 9, 19, 20, and 34.)

Andreas, J., Eric, H., Nicola, M., Rachel, B., Aparna, K. and Tillman, W. (2017), Singing Voice Separation with Deep U-net Convolutional Networks, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 23–27. (cit. on p. 36.)

Aouameur, C., Esling, P. and Hadjeres, G. (2019), Neural Drum Machine: An Interactive System for Real-time Synthesis of Drum Sounds, in *Proceedings of the International Conference on Computational Creativity (ICCC)*, *Charlotte, USA*, pp. 92–9. (cit. on pp. 38 and 89.)

Arewa, O. (2005), From JC Bach to Hip hop: Musical Borrowing, Copyright and Cultural Context, in *North Carolina Law Review, 84(1)*, p. 547. (cit. on p. 10.)

Arık, S. Ö., Jun, H. and Diamos, G. (2018), Fast Spectrogram Inversion using Multi-head Convolutional Neural Networks, in *IEEE Signal Processing Letters, 26(1)*, pp. 94–8. (cit. on p. 90.)

Arjovsky, M. and Bottou, L. (2017), Towards Principled Methods for Training Generative Adversarial Networks, in *Proceedings of the International Conference on Learning Representations (ICLR)*, *Toulon, France*. (cit. on p. 61.)

Arjovsky, M., Chintala, S. and Bottou, L. (2017), Wasserstein GAN, in *Proceedings of the International Conference on Machine Learning (ICML)*, *Sydney, Australia*, pp. 214–23. (cit. on pp. 38, 61, and 62.)

Bakator, M. and Radosav, D. (2018), Deep Learning and Medical Diagnosis: A Review of Literature, in *Multimodal Technologies and Interaction, 2(3)*, pp. 1–12. (cit. on p. 41.)

Barratt, S. T. and Sharma, R. (2018), A Note on the Inception Score, in *Corr*, volume CoRR: abs/1801.01973. (cit. on p. 97.)

Bartlett, B. and Bartlett, J. (2009), *Practical Recording Techniques: The Step-by-Step Approach to Professional Audio Recording*, Focal Press, Waltham, USA. (cit. on pp. 15, 25, and 27.)

Benetos, E., Dixon, S., Duan, Z. and Ewert, S. (2018), Automatic Music Transcription: An Overview, in *IEEE Signal Processing Magazine, 36(1)*, pp. 20–30. (cit. on pp. 3 and 42.)

Bengio, Y., Courville, A. and Vincent, P. (2013), Representation Learning: A Review and New Perspectives, in *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8)*, pp. 1798–828. (cit. on p. 64.)

Bengio, Y., Ducharme, R. and Vincent, P. (2001), A Neural Probabilistic Language Model, in *Advances in Neural Information Processing Systems, 13(1)*, pp. 932–38. (cit. on pp. 37 and 57.)

Bernardo, G. and Bernardes, G. (2021), Leveraging Compatibility and Diversity in Computational Music Mashup Creation, in *Proceedings of the International Audio Mostly Conference*, *Trento, Italy*, p. 248–55. (cit. on p. 36.)

Berns, S. and Colton, S. (2020), Bridging Generative Deep Learning and Computational Creativity, in *Proceedings of the International Conference on Computational Creativity (ICCC)*, *Coimbra, Portugal*, pp. 406–9. (cit. on p. 65.)

Bertin-Mahieux, T., Ellis, D. P., Whitman, B. and Lamere, P. (2011), The Million Song Dataset, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Miami, USA*, pp. 591–96. (cit. on p. 32.)

Bilbao, S., Desvages, C., Ducceschi, M., Hamilton, B., Harrison-Harsley, R., Torin, A. and Webb, C. (2019), Physical Modeling, Algorithms, and Sound Synthesis: The NESS Project, in *Computer Music Journal, 43(3)*, pp. 15–30. (cit. on p. 29.)

Bilbao, S. and Webb, C. J. (2013), Physical Modeling of Timpani Drums in 3D on GPGPUs, in *Journal of the Audio Engineering Society (JAES), 61(10)*, pp. 737–48. (cit. on p. 29.)

Bitton, A., Esling, P., Caillon, A. and Fouilleul, M. (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. 64 and 65.)

Böck, S., Krebs, F. and Widmer, G. (2016), 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. 255–61. (cit. on p. 85.)

Bogdanov, D., Wack, N., Gómez, E., Gulati, S., Herrera, P., Mayor, O., Roma, G., Salamon, J., Zapata, J. and Serra, X. (2013), Essentia: An Open-Source Library for Sound and Music Analysis, in *Proceedings of the ACM International Conference on Multimedia*, *Barcelona, Spain*, p. 855–58. (cit. on p. 34.)

Bogdanov, D., Won, M., Tovstogan, P., Porter, A. and Serra, X. (2019), The MTG-Jamendo Dataset for Automatic Music Tagging, in *International Conference on Machine Learning ICLM*, *Long Beach, USA*. (cit. on p. 32.)

Brandes, L. F. (2007), From Mozart to Hip-Hop: The Impact of Bridgeport v. Dimension Films on Musical Creativity, in *UCLA Entertainment Law Review, 14(1)*, pp. 93–128. (cit. on p. 10.)

Breiman, L. (2001), Random Forests, in *Machine Learning, 45(1)*, pp. 5–32. (cit. on p. 34.)

Bristow-Johnson, R. (1996), Wavetable Synthesis 101, A Fundamental Perspective, in *Audio Engineering Society (AES) Convention*, *Los Angeles, USA*. (cit. on p. 16.)

Broad, T., Berns, S., Colton, S. and Grierson, M. (2021a), Active Divergence with Generative Deep Learning-A Survey and Taxonomy, in *Proceedings of the International Conference on Computational Creativity (ICCC)*, *Mexico City, Mexico*, pp. 227–36. (cit. on pp. 65 and 120.)

Broad, T. and Grierson, M. (2021), Searching for an (Un)stable Equilibrium: Experiments in Training Generative Models without Data, in *Proceedings of the Machine Learning for Creativity and Design Workshop at the Conference on Neural Information Processing Systems (NIPS)*, *Vancouver, Canada*. (cit. on p. 66.)

Broad, T., Leymarie, F. F. and Grierson, M. (2020), Amplifying The Uncanny, in *Proceedings of the Conference on Computation, Communication, Aesthetics and X (xCoAx)*, *Graz, Austria*, pp. 33–44. (cit. on pp. 66 and 102.)

Broad, T., Leymarie, F. F. and Grierson, M. (2021b), Network Bending: Expressive Manipulation of Deep Generative Models, in *Proceedings of the International Conference on Artificial Intelligence in Music, Sound, Art and Design (Evo-MUSART)*, pp. 20–36. (cit. on p. 66.)

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. et al. (2020), Language Models are Few-shot Learners, in *Advances in Neural Information Processing Systems (NIPS), 33(1)*, pp. 1877–901. (cit. on p. 2.)

Bruford, F., Barthet, M., McDonald, S. and Sandler, M. B. (2019), Groove Explorer: An Intelligent Visual Interface for Drum Loop Library Navigation, in *Joint Proceedings of the ACM IUI Workshops co-located with the ACM Conference on Intelligent User Interfaces*, *Los Angeles, USA*. (cit. on p. 35.)

Burkholder, J. P. (1994), The Uses of Existing Music: Musical Borrowings as a Field, in *Notes, the Quarterly Journal of the Music Library Association, 50(3)*, pp. 851–70. (cit. on pp. 4 and 10.)

Butler, D. (2014), 'Way out-of This World!' Delia Derbyshire, Doctor Who and the British Public's Awareness of Electronic Music in the 1960s, in *Critical Studies in Television, 9(1)*, pp. 62–76. (cit. on p. 1.)

Butler, M. J. (2006), *Unlocking the Groove: Rhythm, Meter, and Musical Design in Electronic Dance Music*, Indiana University Press, Bloomington, USA. (cit. on pp. 23 and 24.)

Carr, C. and Zukowski, Z. (2018), Generating Albums with SamplerNN to Imitate Metal, Rock, and Punk Bands, in *CoRR abs/1811.06633*. (cit. on p. 42.)

Casey, M. and Slaney, M. (2006a), The Importance of Sequences in Musical Similarity, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, *Toulouse, France*, pp. 5–8. (cit. on p. 35.)

Casey, M. and Slaney, M. (2007), Fast Recognition of Remixed Music Audio, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, *Honolulu, Hawaii*, pp. 1425–8. (cit. on p. 34.)

Casey, M. A. and Slaney, M. (2006b), Song Intersection by Approximate Nearest Neighbor Search, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Victoria, Canada*, pp. 144–9. (cit. on p. 34.)

Chai, J., Zeng, H., Li, A. and Ngai, E. W. (2021), Deep Learning in Computer Vision: A Critical Review of Emerging Techniques and Application Scenarios, in *Machine Learning with Applications (MLWA), 6(1)*, pp. 1–13. (cit. on p. 2.)

Chandna, P., Ramires, A., Serra, X. and Gómez, E. (2021), LoopNet: Musical Loop Synthesis Conditioned on Intuitive Musical Parameters, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, *Toronto, Canada*, pp. 3395–99. (cit. on p. 17.)

Chang, V. (2009), Records That Play: The Present Past in Sampling Practice, in *Popular Music, 28(2)*, pp. 143–59. (cit. on pp. 3 and 11.)

Charnas, D. (2022), *Dilla Time: The Life and Afterlife of J Dilla, the Hip-hop Producer Who Reinvented Rhythm*, Swift Press, London, UK. (cit. on p. 11.)

Chen, B.-Y., Smith, J. B. and Yang, Y.-H. (2020), Neural Loop Combiner: Neural Network Models for Assessing the Compatibility of Loops, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Montréal, Canada*, pp. 424–31. (cit. on pp. 36 and 120.)

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I. and Abbeel, P. (2016), InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets, in *Proceedings of the Neural Information Processing Systems (NIPS)*, *Barcelona, Spain*, p. 2180–88. (cit. on p. 64.)

Cheshire, M. (2020), Snare Drum Data Set (SDDS): More Snare Drums Than You Can Shake a Stick At, in *Audio Engineering Society (AES) Convention*. (cit. on p. 112.)

Ching, J., Ramires, A. and Yang, Y. (2020), Instrument Role Classification: Auto-tagging for Loop Based Music, in *Proceedings of the Joint Conference on AI Music Creativity*, *Stockholm, Sweden*, pp. 196–202. (cit. on pp. 32, 70, 72, and 74.)

Choi, K., Fazekas, G. and Sandler, M. (2016), Automatic Tagging Using Deep Convolutional Neural Networks, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *New York City, USA*, pp. 805–11. (cit. on p. 31.)

Choi, K., Fazekas, G., Sandler, M. and Cho, K. (2017), Convolutional Recurrent Neural Networks for Music Classification, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, *New Orleans, USA*, pp. 2392–96. (cit. on p. 31.)

Chowning, J. M. (1973), The Synthesis of Complex Audio Spectra by Means of Frequency Modulation, in *Journal of the Audio Engineering Society (JAES), 21(7)*, pp. 526–34. (cit. on p. 29.)

Clevert, D., Unterthiner, T. and Hochreiter, S. (2016), Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs), in *Proceedings of the International Conference on Learning Representations (ICLR)*, *San Juan, Puerto Rico*, pp. 1–14. (cit. on p. 72.)

Cocharro, D., Sioros, G., Caetano, M. F. and Davies, M. E. P. (2014), Real-time Manipulation of Syncopation in Audio Loops, in *Joint Proceedings of the International Computer Music Conference (ICMC) and the Sound and Music Computing Conference (SMC)*, *Athens, Greece*, pp. 536–41. (cit. on p. 17.)

Coleman, G. (2007), Mused: Navigating the Personal Sample Library, in *Proceedings of the International Computer Music Conference (ICMC)*, *Copenhagen, Denmark*, pp. 324–7. (cit. on p. 35.)

Collins, M. (2014), *In the Box Music Production: Advanced Tools and Techniques for Pro Tools*, CRC Press, Boca Ranton, USA. (cit. on p. 17.)

Collins, N. (2001a), Algorithmic Composition Methods for Breakbeat Science, in *Proceedings of Music Without Walls*, *Leicester, UK*, pp. 21–3. (cit. on p. 13.)

Collins, N. (2001b), Further Automatic Breakbeat Cutting Methods, in *Proceedings of the Generative Art Conference*, *Milan, Italy*. (cit. on p. 13.)

Collins, N. (2002), Interactive Evolution of Breakbeat Cut Sequences, in *Proceedings of Cybersonica, Institute of Contemporary Arts*, *London, UK*. (cit. on p. 13.)

Collins, N. (2012), Influence in Early Electronic Dance Music: An Audio Content Analysis Investigation, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Porto, Portugal*, pp. 1–6. (cit. on p. 30.)

Collins, N., McLean, A., Rohrhuber, J. and Ward, A. (2003), Live Coding in Laptop Performance, in *Organised Sound, 8(3)*, pp. 321–30. (cit. on p. 104.)

Collins, N., Ruzicka, V. and Grierson, M. (2020), Remixing AIs: Mind Swaps, Hybrainity, and Splicing Musical Models, in *Proceedings of the Joint Conference on AI Music Creativity*. (cit. on p. 66.)

Davies, H. (1996), A History of Sampling, in *Organised Sound, 1(1)*, pp. 3–11. (cit. on p. 11.)

Davies, M. E. P., Hamel, P., Yoshii, K. and Goto, M. (2014), AutoMashUpper: Automatic Creation of Multi-song Music Mashups, in *, 22(12)*, pp. 1726–37. (cit. on pp. 33, 36, and 69.)

Davis, J. and Goadrich, M. (2006), The Relationship Between Precision-Recall and ROC Curves, in *Proceedings of the International Conference on Machine Learning (ICML)*, *Pittsburgh, Pennsylvania*, pp. 233–40. (cit. on p. 74.)

De Castro, L. N. and Timmis, J. (2002), An Artificial Immune Network for Multimodal Function Optimization, in *Proceedings of the Congress on Evolutionary Computation (CEC), 1(1)*, *Honolulu, USA*, pp. 699–704. (cit. on p. 36.)

Delgado, A., Saitis, C., Benetos, E. and Sandler, M. (2022), Deep Conditional Representation Learning for Drum Sample Retrieval by Vocalisation, in *CoRR abs/2204.04651*. (cit. on p. 34.)

Devlin, J., Chang, M., Lee, K. and Toutanova, K. (2019), BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL-HLT)*, *Minneapolis, USA*, pp. 4171–86. (cit. on pp. 2 and 31.)

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A. and Sutskever, I. (2020), Jukebox: A Generative Model for Music, in *CoRR abs/2005.00341*. (cit. on pp. 3 and 42.)

Dieleman, S. and Schrauwen, B. (2014), End-to-end Learning for Music Audio, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, *Florence, Italy*, pp. 6964–8. (cit. on pp. 31 and 45.)

Dittmar, C., Hildebrand, K. F., Gärtner, D., Winges, M., Müller, F. and Aichroth, P. (2012), Audio Forensics Meets Music Information Retrieval—A Toolbox for Inspection of Music Plagiarism, in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, *Bucharest, Romania*, pp. 1249–53. (cit. on p. 34.)

Dittmar, C. and Müller, M. (2016), Reverse Engineering the Amen Break—Score-informed Separation and Restoration Applied to Drum Recordings, in *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP), 24(9)*, pp. 1535–47. (cit. on p. 14.)

Donahue, C., McAuley, J. J. and Puckette, M. S. (2019), Adversarial Audio Synthesis, in *Proceedings of the International Conference on Learning Representations (ICLR)*, *New Orleans, USA*. (cit. on pp. 38, 90, 92, 93, 95, 96, and 97.)

Drysdale, J., Ramires, A., Serra, X. and Hockman, J. (2022), Improved Automatic Instrumentation Role Classification and Loop Activation Transcription, in *Proceedings of the International Conference on Digital Audio Effects (DAFX)*, *Vienna, Austria*, pp. 264–71. (cit. on p. 70.)

Drysdale, J., Tomczak, M. and Hockman, J. (2020), Adversarial Synthesis of Drum Sounds, in *Proceedings of the International Conference on Digital Audio Effects (DAFX)*, *Vienna, Austria*, pp. 167–72. (cit. on p. 92.)

Drysdale, J., Tomczak, M. and Hockman, J. (2021), Style-based Drum Synthesis with GAN Inversion, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. (cit. on p. 92.)

Duchi, J., Hazan, E. and Singer, Y. (2011), Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, in *Journal of Machine Learning Research (JMLR), 12(61)*, pp. 2121–59. (cit. on p. 49.)

Dupont, S., Dubuisson, T., Urbain, J., Sebbe, R., d'Alessandro, N. and Frisson, C. (2009), Audiocycle: Browsing Musical Loop Libraries, in *International Workshop on Content-Based Multimedia Indexing (CBMI)*, *Chania, Crete*, pp. 73–80. (cit. on p. 35.)

D'Virgilio, N. (2014), How to Control Drum Sustain with Dampening, Accessed 12 March 2021, `https://www.sweetwater.com/insync/how-to-control-drum-sustain-with-dampening`. (cit. on p. 112.)

Emmerson, S. (2018), *The Routledge Research Companion to Electronic Music: Reaching Out with Technology*, Routledge, Abingdon, UK. (cit. on p. 10.)

Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C. and Roberts, A. (2019), GANSynth: Adversarial Neural Audio Synthesis, in *Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, Louisiana, USA*. (cit. on pp. 3, 37, 38, 42, 90, and 96.)

Engel, J., Hantrakul, L. H., Gu, C. and Roberts, A. (2020), DDSP: Differentiable Digital Signal Processing, in *Proceedings of the International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia*. (cit. on p. 37.)

Engel, J., Resnick, C., Roberts, A., Dieleman, S., Norouzi, M., Eck, D. and Simonyan, K. (2017), Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders, in *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1068–77. (cit. on pp. 5, 37, and 42.)

Engeln, L., Le, N. L., McGinity, M. and Groh, R. (2021), Similarity Analysis of Visual Sketch-Based Search for Sounds, in *Proceedings of the International Audio Mostly Conference, Trento, Italy*, p. 101–8. (cit. on p. 35.)

Esling, P., Chemla-Romeu-Santos, A. and Bitton, A. (2018a), Bridging Audio Analysis, Perception and Synthesis with Perceptually-regularized Variational Timbre Spaces, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Paris, France*, pp. 175–81. (cit. on pp. 37, 42, and 65.)

Esling, P., Chemla-Romeu-Santos, A. and Bitton, A. (2018b), Generative Timbre Spaces: Regularizing Variational Auto-encoders with Perceptual Metrics, in *Proceedings of the International Conference on Digital Audio Effects (DAFX), Aveiro, Portugal*, pp. 369–76. (cit. on p. 65.)

Ewoodzie Jr, J. C. (2017), *Break Beats in the Bronx: Rediscovering Hip-hop's Early Years*, UNC Press Books. (cit. on p. 2.)

FitzGerald, D., Cranitch, M. and Coyle, E. (2006), Sound Source Separation Using Shifted Non-Negative Tensor Factorisation, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toulouse, France*, pp. 653–6. (cit. on p. 33.)

Fletcher, N. H. and Rossing, T. D. (1998), *The Physics of Musical Instruments*, Springer, New York, USA. (cit. on p. 26.)

Font, F., Roma, G. and Serra, X. (2013), Freesound Technical Demo, in *ACM International Conference on Multimedia*, pp. 411–412. (cit. on p. 74.)

Fontana, F. and Rocchesso, D. (1998), Physical Modeling of Membranes for Percussion Instruments, in *Acta Acustica United with Acustica, 84(3)*, pp. 529–42. (cit. on p. 29.)

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. 14.)

Fried, O., Jin, Z., Oda, R. and Finkelstein, A. (2014), AudioQuilt: 2D Arrangements of Audio Samples using Metric Learning and Kernelized Sorting, in *International Conference on New Interfaces for Musical Expression (NIME), London, UK*, pp. 281–6. (cit. on p. 34.)

Fukushima, K. (1980), Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position, in *Biological Cybernetics, 36(4)*, pp. 193–202. (cit. on p. 45.)

Gatys, L. A., Ecker, A. S. and Bethge, M. (2016), Image Style Transfer using Convolutional Neural Networks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, *Las Vegas, USA*, pp. 2414–23. (cit. on p. 41.)

Gillet, O. and Richard, G. (2005), Drum Loops Retrieval from Spoken Queries, in *Journal of Intelligent Information Systems (JIIS), 24(2)*, pp. 159–77. (cit. on p. 34.)

Glazyrin, N. (2014), Towards Automatic Content-Based Separation of DJ Mixes into Single Tracks, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Taipei, Taiwan*, pp. 149–54. (cit. on p. 33.)

Glorot, X. and Bengio, Y. (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. 249–56. (cit. on p. 50.)

Goodfellow, I. (2016), NIPS 2016 Tutorial: Generative Adversarial Networks, in *Proceedings of the Neural Information Processing Systems (NIPS)*, *Barcelona, Spain*. (cit. on p. 57.)

Goodfellow, I., Bengio, Y. and Courville, A. (2016), *Deep Learning*, MIT Press, Cambridge, USA. (cit. on pp. 2, 56, and 60.)

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014), Generative Adversarial Nets, in *Proceedings of the Neural Information Processing Systems (NIPS)*, *Montréal, Canada*, pp. 2672–80. (cit. on pp. 37, 38, and 60.)

Greenwald, J. (2002), Hip-Hop Drumming: The Rhyme May Define, but the Groove Makes You Move, in *Black Music Research Journal, 22(2)*, pp. 259–71. (cit. on pp. 24, 25, 26, and 27.)

Griffin, D. and Lim, J. (1984), Signal Estimation from Modified Short-time Fourier Transform, in *IEEE Transactions on Acoustics, Speech, and Signal Processing, 32(2)*, pp. 236–43. (cit. on p. 90.)

Griffin, R. (2016), *David Bowie: The Golden Years*, Omnibus Press, London, USA. (cit. on p. 1.)

Grigorescu, S., Trasnea, B., Cocias, T. and Macesanu, G. (2020), A Survey of Deep Learning Techniques for Autonomous Driving, in *Journal of Field Robotics (JFR), 37(3)*, pp. 362–86. (cit. on p. 41.)

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. and Courville, A. C. (2017), Improved Training of Wasserstein GANs, in *Proceedings of the Neural Information Processing Systems (NIPS)*, *Long Beach, California, USA*, pp. 5767–77. (cit. on pp. 38, 62, and 96.)

Gururani, S. and Lerch, A. (2017), Automatic Sample Detection in Polyphonic Music, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Suzhou, China*, pp. 264–71. (cit. on p. 34.)

Härkönen, E., Hertzman, A., Lehtinen, J. and Paris, S. (2020), GANSpace: Discovering Interpretable GAN Controls, in *Proceedings of the Neural Information Processing Systems (NIPS)*, pp. 9841–50. (cit. on pp. 65 and 105.)

He, K., Zhang, X., Ren, S. and Sun, J. (2015), Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034. (cit. on pp. 50 and 73.)

Heittola, T., Klapuri, A. and Virtanen, T. (2009), Musical Instrument Recognition in Polyphonic Audio Using Source-Filter Model for Sound Separation, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Kobe, Japan*, pp. 327–32. (cit. on p. 30.)

Hennequin, R., Khlif, A., Voituret, F. and Moussallam, M. (2020), Spleeter: A Fast and Efficient Music Source Separation Tool with Pre-Trained Model, in *Journal of Open Source Software, 5(50)*, pp. 1–4. (cit. on pp. 3, 5, and 42.)

Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B. et al. (2017), CNN Architectures for Large-Scale Audio Classification, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, USA*, pp. 131–5. (cit. on p. 98.)

Hewitt, M. (2009), *Composition for Computer Musicians*, Course Technology, Huntington Beach, USA. (cit. on p. 24.)

Heyman, M. (2021), Recreating the Beatles: The Analogues and Historically Informed Performance, in *Journal of Popular Music Studies (JPMS), 33(2)*, pp. 77–98. (cit. on p. 1.)

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S. and Lerchner, A. (2017), beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework, in *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1–22. (cit. on p. 64.)

Hockman, J. (2014), *An Ethnographic and Technological Study of Breakbeats in Hardcore, Jungle and Drum & Bass*, PhD Thesis, McGill University. (cit. on pp. 2, 4, 11, 12, 13, 14, 16, 19, 21, 22, 23, 24, 26, 27, 69, 83, and 111.)

Hockman, J., Davies, M. E. P. and Fujinaga, I. (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. 167–74. (cit. on pp. 13 and 85.)

Hockman, J. A. and Davies, M. (2015), Computational Strategies for Breakbeat Classification and Resequencing in Hardcore, Jungle, and Drum and Bass, in *Proceedings of the International Conference on Digital Audio Effects (DAFX), Trondheim, Norway*. (cit. on pp. 4, 12, 14, and 70.)

Holmes, T. (2012), *Electronic and Experimental Music: Technology, Music, and Culture*, Routledge, Abingdon, UK. (cit. on pp. 1, 10, and 16.)

Huang, C.-Z. A., Koops, H. V., Newton-Rex, E., Dinculescu, M. and Cai, C. J. (2020), AI Song Contest: Human-AI Co-creation in Songwriting, in *CoRR abs/2010.05388*. (cit. on p. 5.)

Huang, H., li, z., He, R., Sun, Z. and Tan, T. (2018), IntroVAE: Introspective Variational Autoencoders for Photographic Image Synthesis, in *Advances in Neural Information Processing Systems (NIPS), 31(1)*, pp. 52–63. (cit. on pp. 59 and 89.)

Huang, J., Wang, J.-C., Smith, J. B. L., Song, X. and Wang, Y. (2021), Modeling the Compatibility of Stem Tracks to Generate Music Mashups, in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 187–95. (cit. on p. 36.)

Huang, X. and Belongie, S. (2017), Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy*, pp. 1501–10. (cit. on p. 65.)

Ioffe, S. and Szegedy, C. (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. 448–56. (cit. on pp. 52 and 72.)

Islam, M. M., Karray, F., Alhajj, R. and Zeng, J. (2021), A Review on Deep Learning Techniques for the Diagnosis of Novel Coronavirus (COVID-19), in *IEEE Access, 9(1)*, pp. 30,551–72. (cit. on p. 2.)

Izotope (2021), Mix Bus 101: Why, When, and How to Group Tracks into a Bus, Accessed 11 December 2022, `https://www.izotope.com/en/learn/mix-buses-101.html`. (cit. on p. 101.)

Jackson, G. (2016), Modern Approaches: Sampling, Accessed 27 August 2022, `https://daily.redbullmusicacademy.com/2016/07/modern-approaches-sampling`. (cit. on pp. 18 and 19.)

Jahanian, A., Chai, L. and Isola, P. (2020), On the "Steerability" of Generative Adversarial Networks, in *Proceedings of the International Conference on Learning Representations (ICLR)*, *Addis Ababa, Ethiopia*. (cit. on p. 65.)

Johnson, E. S. (1987), Protecting Distinctive Sounds: The Challenge of Digital Sampling, in *Journal on Law and Technology, 2(1)*, p. 273. (cit. on p. 10.)

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S. and Saul, L. K. (1999), An Introduction to Variational Methods for Graphical Models, in *Machine Learning, 37(1)*, pp. 183–233. (cit. on p. 59.)

Joyce, J. (2022), Paradox: Breakbeat Mastery, Accessed 5 January 2023, `https://www.ableton.com/en/blog/paradox-breakbeat-mastery`. (cit. on pp. 12 and 22.)

Kapur, A., Benning, M. and Tzanetakis, G. (2004), Query-by-Beat-Boxing: Music retrieval for the DJ, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Barcelona, Spain*, pp. 170–77. (cit. on p. 34.)

Karplus, K. and Strong, A. (1983), Digital Synthesis of Plucked-String and Drum Timbres, in *Computer Music Journal, 7(2)*, pp. 43–55. (cit. on pp. 29 and 90.)

Karras, T., Aila, T., Laine, S. and Lehtinen, J. (2017), Progressive Growing of GANs for Improved Quality, Stability, and Variation, in *Proceedings of the International Conference on Learning Representations (ICLR)*, *Toulon, France*. (cit. on pp. 38 and 62.)

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J. and Aila, T. (2020a), Training Generative Adversarial Networks with Limited Data, in *Advances in Neural Information Processing Systems (NIPS), 33(1)*, pp. 12,104–14. (cit. on p. 120.)

Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J. and Aila, T. (2021), Alias-free Generative Adversarial Networks, in *Advances in Neural Information Processing Systems (NIPS), 34(1)*, pp. 852–63. (cit. on pp. 2, 65, and 121.)

Karras, T., Laine, S. and Aila, T. (2019), A Style-based Generator Architecture for Generative Adversarial Networks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, *Long Beach, California, USA*, pp. 4401–10. (cit. on pp. 41, 65, 93, 94, and 102.)

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J. and Aila, T. (2020b), Analyzing and Improving the Image Quality of StyleGAN, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, *Seattle, Washington, USA*, pp. 8110–19. (cit. on pp. 38, 65, and 93.)

Katz, M. (2012), *Groove Music: The Art and Culture of the Hip-hop DJ*, Oxford University Press on Demand, Oxford, UK. (cit. on p. 12.)

Kilgour, K., Zuluaga, M., Roblek, D. and Sharifi, M. (2019), Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms, in *In Proceedings of the Interspeech*, *Graz, Austria*, pp. 2350–4. (cit. on p. 98.)

Kim, S. (2015), A-Track, Accessed 27 January 2023, `https://www.interviewmagazine.com/music/a-trak`. (cit. on p. 83.)

Kim, T., Choi, M., Sacks, E., Yang, Y.-H. and Nam, J. (2020), A Computational Analysis of Real-World DJ Mixes using Mix-To-Track Subsequence Alignment, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Montréal, Canada*, pp. 764–70. (cit. on p. 33.)

Kim, W. and Nam, J. (2020), Drum Sample Retrieval from Mixed Audio via a Joint Embedding Space of Mixed and Single Audio Samples, in *Audio Engineering Society (AES) Convention*. (cit. on p. 35.)

Kingma, D. P. and Ba, J. (2015), Adam: A Method for Stochastic Optimization, in *Proceedings of the International Conference on Learning Representations (ICLR)*, *San Diego, USA*. (cit. on pp. 49 and 94.)

Kingma, D. P. and Welling, M. (2014), Auto-Encoding Variational Bayes, in *Proceedings of the International Conference on Learning Representations (ICLR)*, *Alberta, Canada*. (cit. on pp. 37, 58, and 59.)

Kitahara, T., Iijima, K., Okada, M., Yamashita, Y. and Tsuruoka, A. (2015), A Loop Sequencer that Selects Music Loops Based on the Degree of Excitement, in *Proceedings of the Sound and Music Computing Conference (SMC)*, *Maynooth, Ireland*, pp. 435–8. (cit. on p. 35.)

Kladder, J. (2016), Maschine-itivity: How I Found My Creativity Using a Digital Sampling Device to Compose Music, in *Journal of Music, Technology & Education (JMTE), 9(3)*, pp. 289–313. (cit. on p. 15.)

Knees, P. and Andersen, K. (2016), Searching for Audio by Sketching Mental Images of Sound: A Brave New Idea for Audio Retrieval in Creative Music Production, in *Proceedings of the ACM on International Conference on Multimedia Retrieval (ICMR)*, *New York City, USA*, pp. 95–102. (cit. on p. 35.)

Kohonen, T. (2001), *Self-Organizing Maps*, Springer, Berlin, Germany. (cit. on p. 34.)

Kong, Z., Ping, W., Huang, J., Zhao, K. and Catanzaro, B. (2020), DiffWave: A Versatile Diffusion Model for Audio Synthesis, in *Proceedings of the International Conference on Learning Representations (ICLR)*, *Vienna, Austria*. (cit. on p. 3.)

Kruse, A. J. (2016), Being Hip-Hop: Beyond Skills and Songs, in *General Music Today, 30(1)*, pp. 53–8. (cit. on p. 21.)

Landy, L. (2012), *Making Music with Sounds*, Routledge, Abingdon, UK. (cit. on p. 16.)

Lattner, S. (2022), SampleMatch: Drum Sample Retrieval by Musical Context, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Bengaluru, India*, pp. 781–8. (cit. on p. 35.)

Lauriola, I., Lavelli, A. and Aiolli, F. (2022), An Introduction to Deep Learning in Natural Language Processing: Models, Techniques, and Tools, in *Neurocomputing, 470(1)*, pp. 443–56. (cit. on p. 2.)

Lavault, A., Roebel, A. and Voiry, M. (2022), StyleWaveGAN: Style-based Synthesis of Drum Sounds using Generative Adversarial Networks for Higher Audio Quality, in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pp. 234–38. (cit. on p. 38.)

Law, E., West, K., Mandel, M., Bay, M. and Stephen Downie, J. (2009), Evaluation of algorithms using games: The case of music tagging, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 387–392. (cit. on p. 32.)

Lee, J., Park, J., Kim, K. L. and Nam, J. (2018), SampleCNN: End-to-End Deep Convolutional Neural Networks Using Very Small Filters for Music Classification, in *Applied Sciences, 8(1)*. (cit. on p. 31.)

Lee, K. and Nam, J. (2019), Learning a Joint Embedding Space of Monophonic and Mixed Music Signals for Singing Voice, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Delft, Netherlands*, pp. 295–302. (cit. on p. 36.)

Lehner, B., Widmer, G. and Sonnleitner, R. (2014), On the Reduction of False Positives in Singing Voice Detection, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, *Florence, Italy*, pp. 7480–4. (cit. on p. 35.)

Levy, M., Sandler, M. and Casey, M. (2006), Extraction of High-Level Musical Structure from Audio Data and Its Application to Thumbnail Generation, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, *Toulouse, France*, pp. 13–6. (cit. on p. 32.)

Li, J. (2022), Recent Advances in End-to-End Automatic Speech Recognition, in *APSIPA Transactions on Signal and Information Processing, 11(1)*, pp. 1–38. (cit. on p. 2.)

Linn, R. (1994), *MPC3000 MIDI Production Center: Software Version 3.0 Operator's Manual*, Akai Electric Company, LTD. (cit. on p. 11.)

López-Serrano, P. (2019), *Analyzing Sample-based Electronic Music using Audio Processing Techniques*, Ph.D. thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU). (cit. on pp. 4 and 17.)

López-Serrano, P., Dittmar, C., Driedger, J. and Müller, M. (2016), Towards Modeling and Decomposing Loop-Based Electronic Music, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *New York City, USA*, pp. 502–8. (cit. on pp. 33, 69, 78, 80, and 82.)

López-Serrano, P., Dittmar, C. and Müller, M. (2017), Finding Drum Breaks in Digital Music Recordings, in *Proceedings of the International Symposium on Computer Music Multidisciplinary Research (CMMR)*, *Porto, Portugal*, pp. 111–22. (cit. on pp. 14, 17, 18, 35, 36, 83, and 85.)

Louie, R., Engel, J. and Huang, C.-Z. A. (2022), Expressive Communication: Evaluating Developments in Generative Models and Steering Interfaces for Music Creation, in *International Conference on Intelligent User Interfaces (IUI)*, *Helsinki, Finland*, pp. 405–17. (cit. on p. 42.)

Lu, W., Wang, J.-C., Won, M., Choi, K. and Song, X. (2021), SpecTNT: a Time-Frequency Transformer for Music Audio, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 396–403. (cit. on p. 31.)

Magazine, S. (2004), Dr. Dre - The Ultimate Interview, in *Scratch Magazine*, p. 73–80. (cit. on p. 11.)

Main, A., Grierson, M., Yamada-Rice, D. and Murr, J. (2022), Augmenting Personal Creativity with Artificial Intelligence: Workshop Proposal for Creativity and Cognition, in *Creativity and Cognition*, *Venice, Italy*, p. 462–65. (cit. on p. 42.)

Makhzani, A., Shlens, J., Jaitly, N. and Goodfellow, I. J. (2015), Adversarial Autoencoders, in *CoRR abs/1511.05644*. (cit. on p. 59.)

Mandel, M. I. and Ellis, D. (2005), Song-Level Features and Support Vector Machines for Music Classification, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *London, UK*, pp. 594–9. (cit. on p. 30.)

Mandel, M. I., Pascanu, R., Eck, D., Bengio, Y., Aiello, L. M., Schifanella, R. and Menczer, F. (2011), Contextual Tag Inference, in *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 7(1)*, pp. 1–18. (cit. on p. 31.)

Marcuse, S. (1975), *A Survey of Musical Instruments*, Harper, New York City, USA. (cit. on p. 26.)

Marques, J. and Moreno, P. J. (1999), A Study of Musical Instrument Classification Using Gaussian Mixture Models and Support Vector Machines, in *Cambridge Research Laboratory Technical Report Series (CRL), 4(1)*. (cit. on p. 30.)

Marrington, M. et al. (2017), Composing with the Digital Audio Workstation, in *The Singer-Songwriter Handbook*, pp. 77–89. (cit. on pp. 14 and 15.)

Martín-Gutiérrez, D., Peñaloza, G. H., Belmonte-Hernández, A. and García, F. Á. (2020), A Multimodal End-to-End Deep Learning Architecture for Music Popularity Prediction, in *IEEE Access, 8(1)*, pp. 39,361–74. (cit. on pp. 3 and 42.)

McFee, B., Salamon, J. and Bello, J. P. (2018), Adaptive Pooling Operators for Weakly Labeled Sound Event Detection, in *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP), 26(11)*, pp. 2180–93. (cit. on pp. 71 and 73.)

McInnes, L., Healy, J., Saul, N. and Großberger, L. (2018), UMAP: Uniform Manifold Approximation and Projection, in *Journal of Open Source Software, 3(29)*, p. 861. (cit. on pp. 106 and 107.)

Mehrabi, A., Choi, K., Dixon, S. and Sandler, M. (2018), Similarity Measures for Vocal-Based Drum Sample Retrieval Using Deep Convolutional Auto-Encoders, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, *Calgary, Canada*, pp. 356–60. (cit. on p. 34.)

Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., Courville, A. C. and Bengio, Y. (2017), SampleRNN: An Unconditional End-to-End Neural Audio Generation Model, in *Proceedings of the International Conference on Learning Representations (ICLR)*, *Toulon, France*. (cit. on p. 58.)

Miranda, E. (2012), *Computer Sound Design: Synthesis Techniques and Programming*, Routledge, Abingdon, UK. (cit. on pp. 28 and 29.)

Mirza, M. and Osindero, S. (2014), Conditional Generative Adversarial Nets, in *CoRR abs/1411.1784*. (cit. on p. 63.)

Moore, B. C. (2012), *An Introduction to the Psychology of Hearing*, Brill, Leiden, Netherlands. (cit. on p. 55.)

Morey, J. and McIntyre, P. (2014), The Creative Studio Practice of Contemporary Dance Music Sampling Composers, in *Dancecult: Journal of Electronic Dance Music Culture, 6(1)*, pp. 41–60. (cit. on pp. 2, 9, 17, 18, 22, 23, 24, and 83.)

Müller, M. (2007), Dynamic Time Warping, in *Information Retrieval for Music and Motion*, pp. 69–84. (cit. on p. 34.)

Ndou, N., Ajoodha, R. and Jadhav, A. (2021), Music Genre Classification: A Review of Deep-Learning and Traditional Machine-Learning Approaches, in *IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, *Toronto, Canada*, pp. 1–6. (cit. on pp. 3 and 42.)

Nistal, J., Aouameur, C., Velarde, I. and Lattner, S. (2022), DrumGAN VST: A Plugin for Drum Sound Analysis/Synthesis with Autoencoding Generative Adversarial Networks, in *Proceedings of the International Conference on Machine Learning (ICML)*, *Baltimore, Maryland, USA*. (cit. on p. 109.)

Nistal, J., Lattner, S. and Richard, G. (2020), DrumGAN: Synthesis of Drum Sounds with Timbral Feature Conditioning Using Generative Adversarial Networks, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Montréal, Canada*, pp. 590–7. (cit. on pp. 38 and 65.)

Ong, B. S. and Streich, S. (2008), Music Loop Extraction from Digital Audio Signals, in *IEEE International Conference on Multimedia and Expo (ICME)*, *Hannover, Germany*, pp. 681–4, IEEE. (cit. on p. 35.)

Otter, D. W., Medina, J. R. and Kalita, J. K. (2020), A Survey of the Usages of Deep Learning for Natural Language Processing, in *IEEE Transactions on Neural Networks and Learning Systems, 32(2)*, pp. 604–24. (cit. on p. 41.)

Owsinski, B. (2017), *The Recording Engineer's Handbook: 4th Edition*, Bobby Owsinski Media Group. (cit. on p. 100.)

Pampalk, E., Hlavac, P. and Herrera, P. (2004), Hierarchical Organization and Visualization of Drum Sample Libraries, in *Proceedings of the International Conference on Digital Audio Effects (DAFX)*, *Naples, Italy*, pp. 3–8. (cit. on p. 34.)

Pearce, A., Brookes, T. and Mason, R. (2017), Timbral Attributes for Sound Effect Library Searching, in *Audio Engineering Society (AES) International Conference on Semantic Audio*, *Erlangen, Germany*. (cit. on p. 38.)

Petrella, N. (2002), *The Ultimate Guide to Cymbals*, Carl Fischer, New York City, USA. (cit. on p. 26.)

Plumerault, A., Borgne, H. L. and Hudelot, C. (2020), Controlling Generative Models with Continuous Factors of Variations, in *Proceedings of the International Conference on Learning Representations (ICLR)*, *Addis Ababa, Ethiopia*. (cit. on p. 65.)

Pons, J. (2019), *Deep Neural Networks for Music and Audio Tagging*, Ph.D. thesis, Universitat Pompeu Fabra. (cit. on p. 77.)

Pons, J., Nieto, O., Prockup, M., Schmidt, E., Ehmann, A. and Serra, X. (2018), End-to-end Learning for Music Audio Tagging at Scale, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Paris, France*, pp. 637–44. (cit. on pp. 31, 45, 70, 71, and 74.)

Pons, J., Pascual, S., Cengarle, G. and Serrà, J. (2021), Upsampling Artifacts in Neural Audio Synthesis, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, *Toronto, Canada*, pp. 3005–9. (cit. on pp. 63 and 101.)

Pons, J., Slizovskaia, O., Gong, R., Gómez, E. and Serra, X. (2017), Timbre Analysis of Music Audio Signals with Convolutional Neural Networks, in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, *Kos island, Greece*, pp. 2744–8. (cit. on pp. 71 and 73.)

Prétet, L., Hennequin, R., Royo-Letelier, J. and Vaglio, A. (2019), Singing Voice Separation: A Study on Training Data, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, *Brighton, UK*, pp. 506–10. (cit. on p. 36.)

Producer's Edge (2008), Issue 03 Pete Rock, in *Producer's Edge*, p. 112–15. (cit. on pp. 11 and 18.)

Rabenstein, R. and Trautmann, L. (2001), Digital Sound Synthesis by Physical Modelling, in *Proceedings of the International Symposium on Image and Signal Processing and Analysis (ISPA) in Conjunction with the International Conference on Information Technology Interfaces (ITI)*, *Pula, Croatia*, pp. 12–23. (cit. on p. 29.)

Radford, A., Metz, L. and Chintala, S. (2016), Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, in *Proceedings of the International Conference on Learning Representations (ICLR)*, *San Juan, Puerto Rico*. (cit. on p. 38.)

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. et al. (2019), Language Models are Unsupervised Multitask Learners, in *OpenAI Blog, 1(8)*, p. 9. (cit. on p. 2.)

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M. and Sutskever, I. (2021), Zero-Shot Text-to-Image Generation, in *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 8821–31. (cit. on p. 41.)

Ramires, A., Chandna, P., Favory, X., Gómez, E. and Serra, X. (2020a), Neural Percussive Synthesis Parameterised by High-Level Timbral Features, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, *Barcelona, Spain*, pp. 786–90. (cit. on p. 38.)

Ramires, A., Font, F., Bogdanov, D., Smith, J. B. L., Yang, Y.-H., Ching, J., Chen, B.-Y., Wu, Y.-K., Wei-Han, H. and Serra, X. (2020b), The Freesound Loop Dataset and Annotation Tool, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Montréal, Canada*, pp. 288–93. (cit. on pp. 32, 70, 74, and 75.)

Ramires, A., Juras, J., Parker, J. D. and Serra, X. (2022), A Study of Control Methods for Percussive Sound Synthesis Based on GANs, in *Proceedings of the International Conference on Digital Audio Effects (DAFX)*, *Vienna, Austria*, pp. 224–31. (cit. on p. 38.)

Ramires, A. and Serra, X. (2019), Data Augmentation for Instrument Classification Robust to Audio Effects, in *Proceedings of the International Conference on Digital Audio Effects (DAFX)*, *Birmingham, UK*. (cit. on pp. 75 and 76.)

Ratcliffe, R. (2014), A Proposed Typology of Sampled Material within Electronic Dance Music, in *Dancecult: Journal of Electronic Dance Music Culture, 6(1)*, pp. 97–122. (cit. on p. 16.)

Reuter, A. (2022), Who let the DAWs out? The Digital in a New Generation of the Digital Audio Workstation, in *Popular Music and Society, 45(2)*, pp. 113–28. (cit. on p. 14.)

Reynolds, S. (2011), *Retromania: Pop Culture's Addiction to Its Own Past*, Faber and Faber, London, UK. (cit. on pp. 2, 9, and 18.)

Reynolds, S. (2012), *Energy Flash: A Journey Through Rave Music and Dance Culture*, Soft Skull Press, Berkeley, USA. (cit. on p. 13.)

Rocha, B., Bogaards, N. and Honingh, A. (2013), Segmentation and Timbre Similarity in Electronic Dance Music, in *Proceedings of the Sound and Music Computing Conference (SMC)*, *Stockholm, Sweden*. (cit. on p. 33.)

Rodgers, T. (2003), On the Process and Aesthetics of Sampling in Electronic Music Production, in *Organised Sound, 8(3)*. (cit. on pp. 3, 9, and 17.)

Roma, G., Green, O. and Tremblay, P. A. (2019), Adaptive Mapping of Sound Collections for Data-driven Musical Interfaces., in *International Conference on New Interfaces for Musical Expression (NIME)*, *Porto Alegre, Brazil*, pp. 313–18. (cit. on p. 35.)

Roma, G. and Serra, X. (2015), Music Performance by Discovering Community Loops, in *Proceedings of the Web Audio Conference (WAV)*, *Paris, France*. (cit. on p. 35.)

Rose, T. (1994), *Black Noise: Rap Music and Black Culture in Contemporary America*, Wesleyan University Press, Middletown, USA. (cit. on p. 9.)

Rossing, T. D. (2001), *Science of Percussion Instruments*, World Scientific Publishing, Singapore. (cit. on p. 26.)

Rossing, T. D., Bork, I., Zhao, H. and Fystrom, D. O. (1992), Acoustics of Snare Drums, in *The Journal of the Acoustical Society of America (JASA), 92(1)*, pp. 84–94. (cit. on p. 26.)

Rossing, T. D., Moore, R. F. and A., W. P. (2014), *The Science of Sound*, Pearson Publishing, London, UK. (cit. on pp. 25 and 100.)

Rouard, S. and Hadjeres, G. (2021), CRASH: Raw Audio Score-based Generative Modeling for Controllable High-resolution Drum Sound Synthesis, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 579–84. (cit. on p. 38.)

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986), Learning Representations by Back-Propagating Errors, in *Nature, 323(1)*, pp. 533–36. (cit. on p. 48.)

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X. and Chen, X. (2016), Improved Techniques for Training GANs, in *Advances in Neural Information Processing Systems (NIPS), 29(1)*, pp. 2234–42. (cit. on p. 97.)

Sanjek, D. (1992), Don't Have to DJ No More: Sampling and the "Autonomous" Creator, in *Cardozo Arts Entertainment Law Journal (AELJ), 10(1)*, pp. 607–23. (cit. on p. 10.)

Sarroff, A. and Casey, M. A. (2014), Musical Audio Synthesis Using Autoencoding Neural Nets, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Taipei, Taiwan*. (cit. on p. 37.)

Scarfe, T., Koolen, W. and Kalnishkan, Y. (2014), Segmentation of Electronic Dance Music, in *International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications*. (cit. on p. 33.)

Schloss, J. G. (2014), *Making Beats: The Art of Sample-based Hip-hop*, Wesleyan University Press, Middletown, USA. (cit. on pp. 17 and 23.)

Schwarz, D., Beller, G., Verbrugghe, B. and Britton, S. (2006), Real-time Corpus-based Concatenative Synthesis with Catart, in *Proceedings of the International Conference on Digital Audio Effects (DAFX)*, *Montréal, Canada*, pp. 279–82. (cit. on p. 35.)

Seetharaman, P. and Pardo, B. (2016), Simultaneous Separation and Segmentation in Layered Music, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *New York City, USA*, pp. 495–501. (cit. on p. 33.)

Serra, X. (2007), State of the Art and Future Directions in Musical Sound Synthesis, in *IEEE Workshop on Multimedia Signal Processing (MMSP)*, *Chania, Greece*, pp. 9–12. (cit. on p. 30.)

Sewell, A. (2013), *A Typology of Sampling in Hip-hop*, Ph.D. thesis, Indiana University. (cit. on p. 16.)

Seymour, M. (2010), Engineer's Guide To Tuning and Damping Drums, Accessed 12 March 2021, `https://www.soundonsound.com/techniques/engineers-guide-tuning-and-damping-drums`. (cit. on p. 112.)

Shelvock, M. T. (2020), *Cloud-based Music Production: Sampling, Synthesis, and Hip-hop*, CRC Press, Boca Raton, USA. (cit. on pp. 17 and 19.)

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryareting the Latent Space of GANs for Semantin, R. et al. (2018), Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, *Calgary, Canada*, pp. 4779–83. (cit. on p. 90.)

Shen, Y., Gu, J., Tang, X. and Zhou, B. (2020), Interpreting the Latent Space of GANs for Semantic Face Editing, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9243–52. (cit. on p. 65.)

Shen, Y., Gu, J., Tang, X. and Zhou, B. (2022), Musika! Fast Infinite Waveform Music Generation, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Bengaluru, India*, pp. 543–50. (cit. on pp. 3 and 42.)

Shen, Y. and Zhou, B. (2021), Closed-form Factorization of Latent Semantics in GANs, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1532–40. (cit. on p. 65.)

Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D. and Wang, Z. (2016), Real-time Single Image and Video Super-resolution using an Efficient Sub-pixel Convolutional Neural Network, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, *Las Vegas, USA*, pp. 1874–83. (cit. on p. 63.)

Shi, Z. and Mysore, G. J. (2018), LoopMaker: Automatic Creation of Music Loops from Pre-Recorded Music, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, *Montréal, Canada*. (cit. on pp. 17 and 36.)

Shier, J., McNally, K. and Tzanetakis, G. (2017), Sieve: A Plugin for the Automatic Classification and Intelligent Browsing of Kick and Snare Samples, in *Workshop on Intelligent Music Production (WIMP)*, *Manchester, UK*. (cit. on p. 34.)

Shier, J., McNally, K., Tzanetakis, G. and Brooks, K. G. (2021), Manifold Learning Methods for Visualization and Browsing of Drum Machine Samples, in *Journal of the Audio Engineering Society (JAES), 69(1)*, pp. 40–53. (cit. on p. 35.)

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. 36.)

Shoemake, K. (1985), Animating Rotation with Quaternion Curves, in *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, *New York City, USA*, p. 245–54. (cit. on p. 102.)

Simonyan, K. and Zisserman, A. (2015), Very Deep Convolutional Networks for Large-Scale Image Recognition, in *Proceedings of the International Conference on Learning Representations (ICLR)*, *San Diego, USA*. (cit. on p. 98.)

Sinnreich, A. (2010), *Mashed Up: Music, Technology, and the Rise of Configurable Culture*, University of Massachusetts Press, Amherst, USA. (cit. on p. 10.)

Slaney, M., Weinberger, K. and White, W. (2008), Learning a Metric for Music Similarity, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Philadelphia, USA*, pp. 313–18. (cit. on p. 30.)

Smaragdis, P. (2004), Non-Negative Matrix Factor Deconvolution: Extraction of Multiple Sound Sources from Monophonic Inputs, in *Proceedings of the International Conference on Independent Component Analysis and Signal Separation (ICA)*, *Granada, Spain*, pp. 494–9. (cit. on p. 33.)

Smith, J. B. and Goto, M. (2018), Nonnegative Tensor Factorization for Source Separation of Loops in Audio, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, *Calgary, Canada*, pp. 171–75. (cit. on pp. 17, 33, 36, 69, 78, 80, and 82.)

Smith, J. B., Kawasaki, Y. and Goto, M. (2019), Unmixer: An Interface for Extracting and Remixing Loops., in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Delft, Netherlands*, pp. 824–31. (cit. on pp. 17 and 33.)

Smith, J. B. L., Kato, J., Fukayama, S., Percival, G. and Goto, M. (2017), The CrossSong Puzzle: Developing a Logic Puzzle for Musical Thinking, in *Journal of New Music Research (JNMR), 46(3)*, pp. 213–28. (cit. on p. 36.)

Smith, J. O. (2010), *Physical Audio Signal Processing for Virtual Musical Instruments and Digital Audio Effects*, W3K Publishing. (cit. on p. 28.)

Snoman, R. (2012), *The Dance Music Manual: Tools, Toys and Techniques*, CRC Press, Boca Raton, USA. (cit. on pp. 19, 23, and 24.)

Solberg, R. (2014), "Waiting for the Bass to Drop": Correlations Between Intense Emotional Experiences and Production Techniques in Build-up and Drop Sections of Electronic Dance Music, in *Dancecult, 6(1)*, pp. 61–82. (cit. on p. 24.)

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S. and Poole, B. (2021), Score-Based Generative Modeling through Stochastic Differential Equations, in *Proceedings of the International Conference on Learning Representations (ICLR)*. (cit. on p. 38.)

Sordo, M. et al. (2012), *Semantic Annotation of Music Collections: A Computational Approach*, Ph.D. thesis, Universitat Pompeu Fabra. (cit. on p. 31.)

Southall, C. (2019), *Automatic Drum Transcription Using Deep Learning*, Ph.D. thesis, Birmingham City University. (cit. on pp. 26 and 27.)

Southall, C., Stables, R. and Hockman, J. (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. 606–12. (cit. on p. 111.)

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014), Dropout: A Simple Way to Prevent Neural Networks from Overfitting, in *The Journal of Machine Learning Research (JMLR), 15(1)*, pp. 1929–58. (cit. on p. 52.)

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. 27.)

Stillar, G. (2005), Loops as Genre Resources, in *Folia Linguistica, 39(1)*, pp. 197–212. (cit. on p. 17.)

Stoller, D., Ewert, S. and Dixon, S. (2018), Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Paris, France*, p. 334–40. (cit. on p. 38.)

Streich, S. and Ong, B. S. (2008), A Music Loop Explorer System, in *Proceedings of International Computer Music Conference (ICMC)*, *Belfast, Ireland*. (cit. on p. 35.)

Sutton, R. (1986), Two Problems with Back Propagation and Other Steepest Descent Learning Procedures for Networks, in *Proceedings of the Annual Conference of the Cognitive Science Society*, *Amherst, USA*, pp. 823–32. (cit. on p. 49.)

The Economist (2011), Seven Seconds of Fire, in *The Economist*, p. 145–6. (cit. on p. 13.)

Tieleman, T., Hinton, G. 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*, pp. 26–31. (cit. on p. 49.)

Tolstikhin, I., Bousquet, O., Gelly, S. and Schölkopf, B. (2018), Wasserstein Auto-Encoders, in *Proceedings of the International Conference on Learning Representations (ICLR)*, *Vancouver, Canada*. (cit. on p. 38.)

Tomczak, M., Goto, M. and Hockman, J. (2020), Drum Synthesis and Rhythmic Transformation with Adversarial Autoencoders, in *Proceedings of the ACM International Conference on Multimedia*, *Seattle, USA*, p. 2427–35. (cit. on p. 65.)

Topel, S. S. and Casey, M. A. (2011), Elementary Sources: Latent Component Analysis for Music Composition, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, *Miami, Florida*, pp. 579–84. (cit. on p. 35.)

Torin, A., Hamilton, B. and Bilbao, S. (2014), An Energy Conserving Finite Difference Scheme for the Simulation of Collisions in Snare Drums., in *Proceedings of the International Conference on Digital Audio Effects (DAFX)*, *Erlangen, Germany*, pp. 145–52. (cit. on p. 29.)

Toulson, R. (2021), *Drum Sound and Drum Tuning: Bridging Science and Creativity*, CRC Press, Boca Raton, USA. (cit. on p. 27.)

Turquois, C., Hermant, M., Gómez-Marín, D. and Jordà, S. (2016), Exploring the Benefits of 2D Visualizations for Drum Samples Retrieval, in *Proceedings of the ACM on Conference on Human Information Interaction and Retrieval (CHIIR)*, *Carrboro, USA*, p. 329–32. (cit. on p. 35.)

Tzanetakis, G. and Cook, P. (2002), Musical Genre Classification of Audio Signals, in *IEEE Transactions on Speech and Audio Processing, 10(5)*, pp. 293–302. (cit. on pp. 30 and 31.)

Tzirakis, P., Zhang, J. and Schuller, B. W. (2018), End-to-End Speech Emotion Recognition Using Deep Neural Networks, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, Canada*, pp. 5089–93. (cit. on p. 45.)

Uria, B., Murray, I. and Larochelle, H. (2014a), A Deep and Tractable Density Estimator, in *Proceedings of the International Conference on Machine Learning (ICML), Bejing, China*, pp. 467–75. (cit. on p. 37.)

Uria, B., Murray, I. and Larochelle, H. (2014b), A Deep and Tractable Density Estimator, in *Proceedings of the International Conference on Machine Learning (ICML), Beijing, China*, pp. 467–75. (cit. on p. 57.)

Vályi, G. (2010), *Digging in the Crates: Practices of Identity and Belonging in a Translocal Record Collecting Scene*, Ph.D. thesis, Goldsmiths, University of London. (cit. on pp. 18 and 19.)

Van Balen, J., Haro, M., Serra, J. et al. (2012), Automatic Identification of Samples in Hip Hop Music, in *Proceedings of the International Symposium on Computer Music Modeling and Retrieval (CMMR), London, UK*, pp. 544–51. (cit. on pp. 33 and 34.)

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. and Kavukcuoglu, K. (2016), WaveNet: A Generative Model for Raw Audio, in *Proceedings of the ISCA Speech Synthesis Workshop, Sunnyvale, USA*, p. 125. (cit. on pp. 37 and 58.)

van der Maaten, L. and Hinton, G. E. (2008), Visualizing Data using t-SNE, in *Journal of Machine Learning Research (JMLR), 9(1)*, pp. 2579–605. (cit. on p. 106.)

Vande Veire, L. and De Bie, T. (2018), From Raw Audio to a Seamless Mix: Creating an Automated DJ System for Drum and Bass, in *EURASIP Journal on Audio, Speech, and Music Processing*, pp. 1–21. (cit. on pp. 33 and 69.)

Veal, M. (2013), *Dub: Soundscapes and Shattered Songs in Jamaican Reggae*, Wesleyan University Press, Middletown, USA. (cit. on p. 1.)

Villani, C. (2009), *Optimal Transport: Old and New*, Springer, Berlin, Germany. (cit. on p. 61.)

Voulodimos, A., Doulamis, N., Doulamis, A. and Protopapadakis, E. (2018), Deep Learning for Computer Vision: A Brief Review, in *Computational Intelligence and Neuroscience*. (cit. on p. 41.)

Wang, A. et al. (2003), An Industrial Strength Audio Search Algorithm, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, volume 2003, *Baltimore, USA*, pp. 7–13. (cit. on p. 34.)

Weiss, H. (2016), *Out of the Shadows*, Accessed 27 January 2023, `https://www.interviewmagazine.com/music/dj-shadow`. (cit. on p. 83.)

White, T. (2016), Sampling Generative Networks: Notes on a Few Effective Techniques, in *CoRR abs/1609.04468*. (cit. on p. 102.)

Whitney, J. L. (2013), *Automatic Recognition of Samples in Hip-hop Music Through Non-negative Matrix Factorization*, Master's thesis, University of Miami. (cit. on p. 34.)

WhoSampled (2023), Amen, Brother by The Winstons, Accessed 3 January 2023, `https://www.whosampled.com/The-Winstons/Amen,-Brother/`. (cit. on p. 13.)

Winston, E. and Saywood, L. (2019), Beats to Relax/Study to: Contradiction and Paradox in Lofi Hip hop, in *Journal of the International Association for the Study of Popular Music (IASPM), 9(2)*, pp. 40–54. (cit. on p. 101.)

Won, M., Choi, K. and Serra, X. (2021a), Semi-Supervised Music Tagging Transformer, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 769–76. (cit. on p. 31.)

Won, M., Chun, S., Nieto, O. and Serra, X. (2020a), Data-Driven Harmonic Filters for Audio Representation Learning, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 536–40. (cit. on pp. 31, 32, 71, 72, and 73.)

Won, M., Chun, S. and Serra, X. (2019), Toward Interpretable Music Tagging with Self-attention, in *CoRR abs/1906.04972*. (cit. on p. 31.)

Won, M., Ferraro, A., Bogdanov, D. and Serra, X. (2020b), Evaluation of CNN-Based Automatic Music Tagging Models, in *CoRR abs/2006.00751*. (cit. on pp. 32, 70, 71, 74, and 77.)

Won, M., Oramas, S., Nieto, O., Gouyon, F. and Serra, X. (2021b), Multimodal Metric Learning for Tag-based Music Retrieval, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, Canada*, pp. 591–95. (cit. on p. 32.)

Wu, C.-W., Dittmar, C., Southall, C., Vogl, R., Widmer, G., Hockman, J., Müller, M. and Lerch, A. (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. 3 and 42.)

Xia, W., Zhang, Y., Yang, Y., Xue, J.-H., Zhou, B. and Yang, M.-H. (2023), GAN Inversion: A Survey, in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 45(3)*. (cit. on p. 108.)

Yadati, K., Larson, M. A., Liem, C. C. and Hanjalic, A. (2014), Detecting Drops in Electronic Dance Music: Content Based Approaches to a Socially Significant Music Event, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Taipei, Taiwan*, pp. 143–48. (cit. on pp. 33 and 82.)

Yang, G., Yang, S., Liu, K., Fang, P., Chen, W. and Xie, L. (2021), Multi-Band MelGAN: Faster Waveform Generation for High-Quality Text-to-Speech, in *IEEE Spoken Language Technology Workshop (SLT)*, pp. 492–8. (cit. on p. 2.)

Zeiler, M. D. and Fergus, R. (2014), Visualizing and Understanding Convolutional Networks, in *Proceedings of the European Conference on Computer Vision (ECCV), Zurich, Switzerland*, pp. 818–33. (cit. on p. 62.)

Zölzer, U., Amatriain, X., Arfib, D., Bonada, J., De Poli, G., Dutilleux, P., Evangelista, G., Keiler, F., Loscos, A., Rocchesso, D. et al. (2002), *DAFX-Digital Audio Effects*, John Wiley & Sons, Hoboken, USA. (cit. on p. 104.)

Appendix A

Publications

ADVERSARIAL SYNTHESIS OF DRUM SOUNDS

Jake Drysdale, Maciek Tomczak, Jason Hockman

Digital Media Technology Lab (DMT Lab)
Birmingham City University
Birmingham, United Kingdom
jake.drysdale, maciek.tomczak, jason.hockman@bcu.ac.uk

ABSTRACT

Recent advancements in generative audio synthesis have allowed for the development of creative tools for generation and manipulation of audio. In this paper, a strategy is proposed for the synthesis of drum sounds using generative adversarial networks (GANs). The system is based on a conditional Wasserstein GAN, which learns the underlying probability distribution of a dataset compiled of labeled drum sounds. Labels are used to condition the system on an integer value that can be used to generate audio with the desired characteristics. Synthesis is controlled by an input latent vector that enables continuous exploration and interpolation of generated waveforms. Additionally we experiment with a training method that progressively learns to generate audio at different temporal resolutions. We present our results and discuss the benefits of generating audio with GANs along with sound examples and demonstrations.

1. INTRODUCTION

Sample-based electronic music (EM) describes a variety of genres that emerged through advancements in audio production and digital sampling technologies. EM is mainly created through the use of digital audio workstation (DAW) software for arranging and manipulating short audio recordings, commonly referred to as samples. Early sampling technologies (e.g., Akai S950) were limited by a small amount of memory; however, this constraint stimulated creativity, artistic choices, and new genres of music. Considering the abundance of free and affordable audio sample libraries available at present, there is the potential for an EM producer's personal collection of samples to become unwieldy and therefore difficult to navigate and maintain.

Sample selection is an integral part of the EM production workflow and is one of the key skills harnessed by EM producers. The selection of samples in this context is a meticulous retrieval task involving careful listening for key subjective attributes (e.g., warmth, boominess) of particular timbral features. Online sample libraries such as Splice[1] and Loopmasters[2] have well-annotated databases with high quality sounds; however, when a producer is searching a collection for an exact sample or a sample with certain characteristics (e.g., bass-heavy kick), the sound selection process can be tedious and labor-intensive.

[1] https://splice.com/
[2] https://www.loopmasters.com/

In this paper, a system is presented that allows EM producers to interactively generate and fine tune novel audio sounds based on their own personal collections. The system is based on a generative adversarial network, which learns a mapping between a collection (i.e., dataset) of labelled drum sounds and a low-dimensional latent space that provides high-level control of the input data distribution.

1.1. Background

Advancements in generative modelling have allowed for the development of novel tools for the generation and manipulation of audio. Generative models learn the underlying probability distribution of a given dataset and produce new data based on example observations. Generative methodologies include generative adversarial networks (GANs) [1], autoencoders [2] and autoregressive networks [3]. Autoencoders map high-dimensional data distributions onto low-dimensional latent spaces and reconstruct the output from this representation using a decoder. Several generative models using autoencoders have been proposed for the task of generalised musical audio generation including autoregressive (AR) models (e.g., [4, 5]) and non-AR models (e.g. [6, 7, 8]). AR models for raw audio synthesis have the capacity to generate high fidelity audio, yet this comes at the cost of slow generation and the inability to learn compact latent space representations. An alternative solution is found in GANs, a subset of non-AR generative models, which map low-dimensional latent spaces to complex data distributions through an adversarial training strategy [1]. The generator learns to produce realistic synthesized data from a prior distribution, while the discriminator learns to correctly classify real and synthetic data. GANs can be conditioned on additional information (e.g., pitch, instrument class) enabling high-level control over data generation [9]. Unlike AR models, GANs are capable of parallelised training and generation. However, GANs require much larger models to generate longer audio recordings, becoming computationally expensive. Thus, GANs are well-suited for the synthesis of short audio recordings such as drum sounds.

Donahue et al. [10] were the first apply adversarial learning to musical audio using a modified deep convolutional GAN [11] that operates on raw audio data. Alternatively, Engel et al. [12] proposed GANSynth, an adversarial approach to audio synthesis that utilised recent improvements in the training stability of GANs [13, 14, 15]. Musical notes are conditioned with labels representing the pitch content and are modelled as log magnitude and instantaneous frequency spectrograms, which are used to approximate the time-domain signal. More recently, Engel et al. [16] achieved high resolution audio generation without the need for large AR models or adversarial losses through a modular approach to generative audio modeling that integrates digital signal processing elements into a neural network.

Specific to the generation of drum sounds, Aouameur et al.

[8] used a conditional Wasserstein autoencoder to generate audio spectrograms that are inverted to audio through a multi-head CNN. Ramires et al. [17] synthesized percussion sounds with high-level control over timbral characteristics using Wave-u-net [18]. Tomczak et al. [19] proposed a method for joint synthesis and rhythm transformation of drum sounds by combining the use of adversarial autoencoders with a Wasserstein GAN adversarial framework.

1.2. Motivation

In this paper, a system for synthesising drum samples is presented, which is suitable for generating novel drums sounds based on a producers personal sample collection. The system is designed to be lightweight, in that it can learn to generate high-quality audio when trained using a small amount of data. High-level conditioning organises drum sounds into specific categories, while a compact latent space with low dimensionality is used for intuitive synthesis control to output a variety of different drum sounds. In addition, interpolating the compact latent space of a learned generative model provides an intuitive way for EM producers to morph between generated drum samples when composing new grooves and rhythms.

The system is realised through a conditional Wasserstein generative adversarial network trained with a small dataset of labelled drums sounds. Conditioning is achieved with the three main percussion instruments from the common drum kit—that is, kick drum, snare drum, and cymbals—and it can generate a diverse range of sounds when trained on a relatively small dataset of short audio recordings.

By varying the input latent vector, large or subtle variations can be made to the timbral characteristics of the output. In order to reduce training time, a progressive growing training methodology similar to [13] is considered, in which audio is incrementally generated at increasingly higher temporal resolutions.

The remainder of this paper is structured as follows: Section 2 presents our proposed method for drum sound generation. Section 3 presents our training procedure and dataset processing, and Section 4 provides the results and discussion. Conclusions and suggestions for future work are presented in Section 5.

2. METHOD

The proposed approach to drum synthesis builds upon the architecture of WaveGAN [10] but is designed specifically for conditional audio generation of a variety of different drum sounds. Figure 1 presents a general overview of the proposed system. Generator G is trained to generate audio signals given a latent vector z and a conditional variable y, and discriminator D is trained to estimate the Wasserstein distance between the generated and observed distributions. Both networks are optimised simultaneously until G can produce drum samples that are indistinguishable from the observed training data.

The original GAN framework as proposed by [1] defines an adversarial game between generator network G and discriminator network D. G is used to learn mappings from a noise space Z to drum data space X. $Z = \mathbb{R}^{d_z}$, where d_z is a hyperparameter that controls the dimensionality of Z. Latent variables $z \in Z$ are sampled from a known prior $p(z)$, which is modelled with a simple distribution (e.g., Gaussian, Uniform). X is the drum data space that represents the input to D or output of G. As training data, drum samples D are drawn from a real distribution $p_D(x)$. By

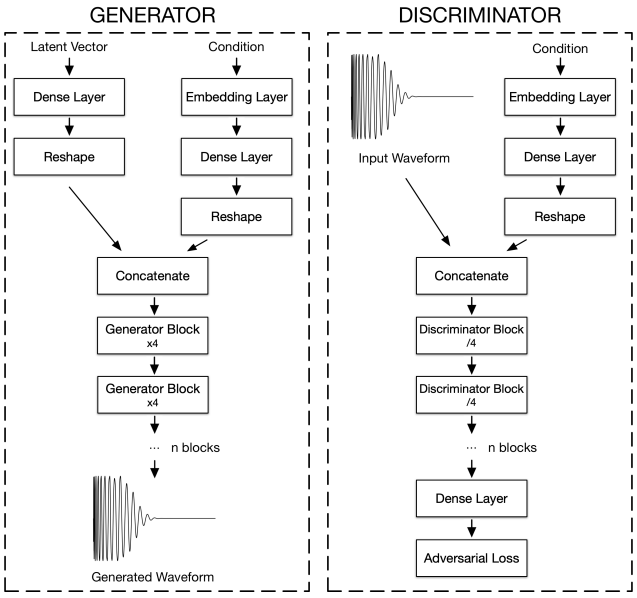

Figure 1: *Overview of proposed system for drum synthesis: Generator G (left) is trained to generate audio given a latent vector z and conditioning variable y. Discriminator D (right) is trained to minimise the Wasserstein distance between the generated distribution and the observed distribution.*

sampling from $p(z)$, G can be used to output drums that represent a synthetic distribution $q(x)$. Following the more general formulation introduced in [20], the GAN learning problem aims at finding a min-max optimisation of objective V between the pair of G and D (i.e., Nash equilibrium), of the value function defined as:

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_D(x)}[f(D(x))] + \mathbb{E}_{z\sim p(z)}[f(-D(G(z)))], \quad (1)$$

where $\mathbb{E}[\cdot]$ denotes expectation, and $f : \mathbb{R} \to \mathbb{R}$ is a concave function. G is trained to output $q(x)$ as close to $p_D(x)$ as possible. D is trained to distinguish between real data P_X and synthesised data $q(x)$. Convergence occurs when G can mislead D by generating synthesized samples that are indistinguishable from real samples.

Training GANs correctly utilising the original formulation is difficult and prone to mode collapse, resulting in reduced sample variability. To help stabilise training, Arjovsky et al. [14] suggest minimising the Wasserstein distance between the generated and observed distributions.

D is modified to emit an unconstrained real number rather than a probability value to recover the traditional GAN [1] formulation $f(x) = -log(1 + exp(-x))$, where f is the logistic loss. This convention slightly differs from the standard formulation in that the discriminator outputs the real-valued *logits* and the loss function would implicitly scale this to a probability. The Wasserstein GAN is achieved by taking $f(x) = x$. Within this formulation, f has to be a 1-Lipschitz function and D is trained to assist in computing the Wassertein distance, rather than to classify samples as real or fake. To enforce the Lipschitz constraint, Arjovsky et al. [14] suggest the application of a simple clipping function to restrict the maximum weight value in f. To avoid subsequent difficulties in optimisation (i.e., exploding or vanishing gradients), the authors

in [15] utilised a gradient penalty parameterised by penalty coefficient λ to enforce the constraint.

In the conditional formulation of the GAN, the G and D networks use additional input layers with labels y. The updated objective function can be stated as:

$$\min_G \max_D V(G,D) = \mathbb{E}_{x,y\sim p_D(x,y)}[f(D(x))] + \mathbb{E}_{y\sim p(y),z\sim p(z)}[f(-D(G(z,y),y))], \quad (2)$$

where $p(y)$ is the prior conditioning distribution. Conditioning the system on labels allows for targeted generation of drum sounds from a specific category. Methods for categorical conditioning commonly involve encoding conditional labels as one-hot vectors and concatenating them with the latent code [9]; however, this can lead to undesirable behaviour such as cross-over between classes. Following [21], an embedding space Y is used to condition the model on external information, where $Y = \mathbb{R}^{d_Y}$, where d_Y is a hyperparameter used to control the dimensionality of Y.

2.1. Model Details

In order to learn an instrument-specific encoding, conditioning variable y is passed through an embedding layer with a dimensionality w, such that each of the three drum classes are mapped to a different w-element vector representation that is learned by G ($w = 50$). The embedding layer and latent vector are then scaled to the initial size of the network using a dense layer, then concatenated together and passed through a series of upsampling blocks to output a generated waveform. Each upsampling block consists of one-dimensional nearest neighbour upsampling with a stride of 4, a one-dimensional convolutional layer with a kernel length of 25, and a ReLU activation. Thus, at each block the number of audio samples is increased by a factor of 4 with the output layer passed through a $tanh$ activation.

Discriminator network D mirrors the architecture in G. D takes an audio signal and conditioning variable y. In D, y is passed to an identical embedding layer to that in G and is scaled to the size of the input waveform using a dense layer and reshaping. This representation is then concatenated with the input waveform and passed through a series of downsampling blocks. Each downsampling block consists of a convolutional layer with a stride of 4 and kernel length of 25, a leaky ReLU activation (α = 0.2). Thus, at each stage of the discriminator the input waveform is decreased by a factor of 4. The final layer of D is a dense layer with a linear activation function that outputs the authenticity of the input audio sample through the Wasserstein distance.

Upsampling in generator networks is known to cause periodic checkerboard artifacts when synthesising images [22]. When generating raw audio, checkerboard artifacts can be perceived as pitched noise that degrades the overall audio quality. An optimisation problem can occur when *D* learns to reject generated audio with artifact frequencies that always occur at a particular phase. Donahue et al. [10] introduced a phase shuffle module that randomly perturbs the phase at each layer of *D*. Phase shuffle forces *D* to become invariant to the phase of the input waveform and is controlled by hyperparameter s that perturbs the phase of a layer's activations by -s to s samples (s = 2).

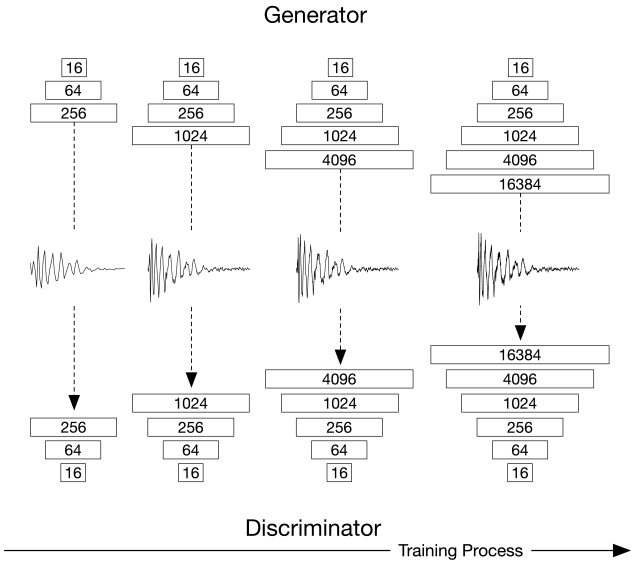

Figure 2: *Progressive growing procedure, in which D and G begin learning with low resolution audio resolution of 256 samples. As training advances new layers are added to the models to incrementally increase the number of samples by a multiple of 4 thus, learning higher frequencies as training progresses.*

3. TRAINING

3.1. Network training

In order to optimise Equation 2 we use alternating updates between networks G and D. At each training iteration, the parameters of network D are updated k times for each G parameter update ($k = 5$). The model is trained using the Adam optimiser [23] with a learning rate $2e$–4, $\beta_1 = 0.5$, $\beta_2 = 0.99$ for a 2000 epochs and 50000 iterations in total, where each iteration takes a mini-batch of 64 examples. The model is trained using a gradient penalty coefficient (λ = 10). n upsampling and downsampling blocks are used to allow for the generation of T samples of audio. Following [10], the latent dimensionality d_z was initially set to 100 and a second model is trained with a lower dimensionality ($d_z = 3$) to explore the tradeoff between dimensionality and audio quality.

3.2. Progressive Growing

To reduce the length of training time, a progressive growing procedure is adopted during training. Following [13], the model is initially trained with downsampled input audio data, then learns to generate output at samplerates of incrementally higher quality. Figure 2 depicts the progressive growing procedure for networks D and G, which are trained on low resolution audio until stable. Additional layers are then added to support more audio samples and thus higher samplerates can be used to sample the audio. Higher frequencies are learned in successive epochs as training progresses. As in [10], the output size of layers grows in increments of 4 until the desired samplerate of is met. When training completes at its current resolution, networks are incrementally grown by adding a new set of layers to increase the resolution each time by a multiple of 4. Skip connections are used to connect the new block to the input of D or output of G and the newly added

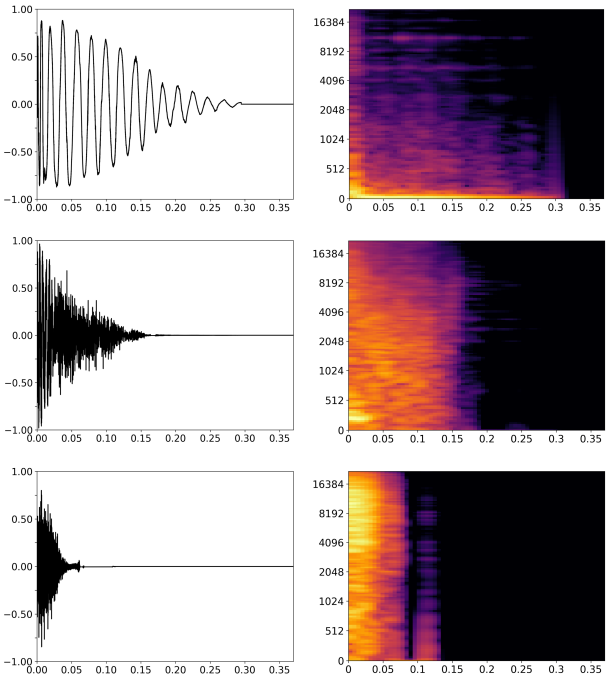

Figure 3: *Example waveform generations (left) and corresponding Mel-scaled log frequency spectrograms (right) for kick drum (top), snare drum (middle) and cymbal (bottom).*

layers are linearly faded in to prevent shocks from the sudden addition of a larger layer. Fading between layers is controlled by parameter r, which is linearly interpolated from 0 to 1. Learning the earlier simple models first helps to stabilise training and progressively learn finer high frequency details. G and D both start training with a short audio length of 256 samples. As training advances, we incrementally add layers to both G and D to increase the number of samples used in the generated audio.

3.3. Dataset

For all experiments, both networks are trained using raw audio waveforms. We compiled a dataset of drums sounds D selected from a wide variety of sample libraries (including Loopmasters and Splice). Sounds were categorised manually into p domains comprising kick, snare and cymbal samples. Each domain contains 3000 individual samples resulting in a total dataset size of 9000 samples. All samples are mono 16-bit PCM audio files sampled at 44.1kHz. Prior to preprocessing, the mean sample length of each audio file in the dataset was 18234 samples (i.e., 0.41s). In accordance with the network architecture in [10], we choose the training data input length to the nearest power of two ($T = 16384$) to satisfy the symmetric structure of networks G and D. Each training sample is trimmed or zero-padded to ensure a constant length of T samples. All waveforms are normalised and a short linear fade of samples is applied to the start and end of each waveform to ensure that they consistently begin and end at 0 amplitude.

4. RESULTS

A system for generative audio synthesis of drum sounds has been implemented as presented in Sections 2 and 3. We report on the system's capacity for generating coherent drum samples and provide an accompanying webpage[3] for examples of individual generated audio samples, interpolation experiments, and example usage within electronic music compositions. These examples allow for subjective evaluation of audio generation quality and waveform interpolation properties.

4.1. Generation Quality

Figure 3 presents examples of a kick drum, snare drum and cymbal generated through the system output. Informal listening tests were conducted to assess the generation quality of audio samples from each class. Conditioning the system with labels improves overall quality and omits overlap between classes. Generally, kick and snare drums can be more easily modelled by the system and are less prone to artifacts. As can be seen from the spectrograms in Figure 3, some checkerboard artifacts remain; however, this does not have a considerable effect on the overall perceived quality of the drum sounds and in most cases could be removed with simple post-processing (e.g., amplitude fading, equalisation). Inclusion of the progressive growing procedure results in both reduced training time and coherent audio generated at an earlier stage. Unfortunately, this results in an increase in artifacts present, degrading the perceived quality of the generations. Due to its fast training time, the progressive growing model could be used as a tool to preview drum samples from a large collection.

4.2. Latent Space Dimensionality

As the proposed system is intended to allow producers to interactively navigate a compact representation of audio sounds, experimentation was undertaken with a small latent dimensionality. The dimensionality of the latent space and its relationship to generation quality and diversity in GANs has yet to be thoroughly explored in literature.

For comparison, we provide a selection of randomly generated drum sounds from each domain using $d_z = 100$ and $d_z = 3$. Interestingly, the size of the latent space had little effect on output audio quality, following findings in other similar research [24]. Different values for d_z returned similar results, leading to our early conclusion that latent parameters up to rank three define the majority of parameter variance within the set of 100 dimensions; however, additional investigation is required to validate this.

4.3. Waveform Interpolation

The proposed system learns to map points in the latent space to the generated waveforms. The structure of the latent space can be explored by interpolating between two random points. Experiments with linear interpolation and spherical linear interpolation are provided on the accompanying webpage. The purpose of the spherical linear interpolation experiment is to ensure that the curving of the space is taken into account as linear interpolation assumes that the latent space is a uniformly distributed hypercube. When traversing the latent space, changes in audio quality are continuous

[3] https://jake-drysdale.github.io/blog/adversarial-drum-synthesis/

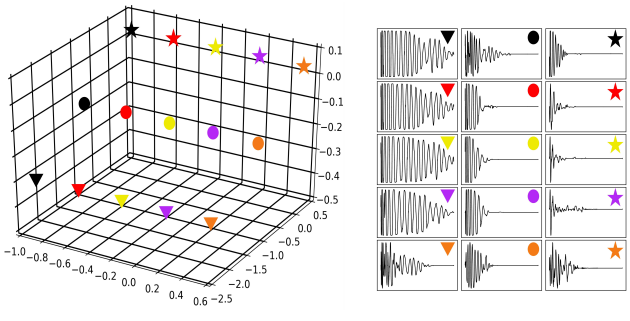

Figure 4: *Interpolation in the latent space for kick drum generation. Kick drums are generated for each point along linear paths through the latent space (left). Paths are colour coded and subsequent generated audio appears across rows (right).*

and without abrupt variation. Figure 4 demonstrates the transition between output kick drums when navigating linearly through the latent space. Timbral modifications can be made to a generation by making adjustments to latent variables Z. Larger steps in the latent space are perceptually equivalent to smoothly mixing amplitudes between distinct drum sounds whereas smaller adjustments result in subtle variations of timbral characteristics. Subtle variations in timbre could be a useful for humanizing programmed drum sequences to provide a more natural feel. While the effect each dimension in d has on the output can not be anticipated, many examples demonstrate consistent variations in pitch, envelope shape and the presence or omission of high and low frequencies. Spherical interpolation seem to result in a more abrupt change of timbral characteristics (e.g., alteration between different kick drum sounds) than linear interpolation.

5. CONCLUSIONS AND FUTURE WORK

A method for generative audio synthesis of drum sounds using a generative adversarial network has been presented. This system provides a music production tool that encourages creative sound experimentation. The results demonstrate the capacity of the conditional model to generate a wide variety of different class-specific drums sounds. High-level conditioning organises drum sounds into specific categories, while a compact latent space allows for intuitive synthesis control over output generations. The model is lightweight and can be trained using a reasonably small dataset to generate high-quality audio, further demonstrating the potential of GAN-based systems for creative audio generation. The experimental dataset could be replaced with an EM producers personal collection of samples and custom tags could be defined for conditioning. Future work will involve embedding the system into an audio plug-in that can be evaluated by EM producers in efforts to inform and improve the breadth of the design goals. The plug-in will be designed to have various parameters that enable navigation of the latent space.

6. REFERENCES

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial nets," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pp. 2672–2680, 2014.

[2] Diederik P. Kingma and Max Welling, "Auto-encoding variational bayes.," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[3] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "WaveNet: A generative model for raw audio," in *Proceedings of the ISCA Speech Synthesis Workshop*, 2016.

[4] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan, "Neural audio synthesis of musical notes with WaveNet autoencoders," in *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1068–1077, 2017.

[5] Maciek Tomczak, Jake Drysdale, and Jason Hockman, "Drum translation for timbral and rhythmic transformation," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2019.

[6] Philippe Esling, Axel Chemla-Romeu-Santos, and Adrien Bitton, "Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces," in *Proceedings of the International Society of Music Information Retrieval Conference (ISMIR)*, pp. 175–181, 2018.

[7] Adrien Bitton, Philippe Esling, Antoine Caillon, and Martin Fouilleul, "Assisted sound sample generation with musical conditioning in adversarial auto-encoders," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2019.

[8] Cyran Aouameur, Philippe Esling, and Gaëtan Hadjeres, "Neural drum machine: An interactive system for real-time synthesis of drum sounds," in *Proceedings of the International Conference on Computational Creativity (ICCC)*, 2019.

[9] Mehdi Mirza and Simon Osindero, "Conditional generative adversarial nets," *arXiv:1411.1784*, 2014.

[10] Chris Donahue, Julian J. McAuley, and Miller S. Puckette, "Adversarial audio synthesis," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[11] Alec Radford, Luke Metz, and Soumith Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[12] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts, "GANSynth: Adversarial neural audio synthesis," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[13] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[14] Martin Arjovsky, Soumith Chintala, and Léon Bottou, "Wasserstein gan," in *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 214–223, 2017.

[15] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C. Courville, "Improved training of wasserstein gans," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pp. 5767–5777, 2017.

[16] Jesse Engel, Lamtharn (Hanoi) Hantrakul, Chenjie Gu, and Adam Roberts, "Ddsp: Differentiable digital signal processing," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[17] António Ramires, Pritish Chandna, Xavier Favory, Emilia Gómez, and Xavier Serra, "Neural percussive synthesis parameterised by high-level timbral features," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 786–790, 2020.

[18] Daniel Stoller, Sebastian Ewert, and Simon Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pp. 334–340, 2018.

[19] Maciek Tomczak, Masataka Goto, and Jason Hockman, "Drum synthesis and rhythmic transformation with adversarial autoencoders," in *Proceedings of the ACM International Conference on Multimedia*, 2020.

[20] Vaishnavh Nagarajan and J. Zico Kolter, "Gradient descent gan optimization is locally stable," in *Proceedings of the Advances in neural information processing systems (NIPS)*, pp. 5585–5595, 2017.

[21] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus, "Deep generative image models using a Laplacian pyramid of adversarial networks," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pp. 1486–1494, 2015.

[22] Augustus Odena, Vincent Dumoulin, and Chris Olah, "Deconvolution and checkerboard artifacts," in *Distill*, 2016.

[23] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[24] Thomas Pinetz, Daniel Soukup, and Thomas Pock, "Impact of the latent space on the ability of GANs to fit the distribution," `https://openreview.net/forum?id=Hygy01StvH`, 2019.

STYLE-BASED DRUM SYNTHESIS WITH GAN INVERSION

Jake Drysdale, Maciek Tomczak, Jason Hockman
Sound and Music Analysis (SoMA) Group, Digital Media Technology Lab (DMT Lab)
School of Computing and Digital Technology
Birmingham City University
Birmingham, UK
jake.drysdale@bcu.ac.uk

ABSTRACT

Neural audio synthesizers exploit deep learning as an alternative to traditional synthesizers that generate audio from hand-designed components, such as oscillators and wavetables. For a neural audio synthesizer to be applicable to music creation, meaningful control over the output is essential. This paper provides an overview of an unsupervised approach to deriving useful feature controls learned by a generative model. A system for generation and transformation of drum samples using a style-based generative adversarial network (GAN) is proposed. The system provides functional control of audio style features, based on principal component analysis (PCA) applied to the intermediate latent space. Additionally, we propose the use of an encoder trained to invert input drums back to the latent space of the pre-trained GAN. We experiment with three modes of control and provide audio results on a supporting website.

1. INTRODUCTION

One of the chief skills harnessed by electronic music (EM) producers is the ability to select and arrange suitable drum sounds. The integration of drum sounds into an EM composition may be achieved either through the time-consuming task of browsing sound libraries for an appropriate drum recording or alternatively through the use of traditional drum synthesizers, which require mastery over a large number of parameters, and provide only limited control over sound generation. More recently, neural drum synthesis [1–3] has been proposed to allow EM producers to interactively generate and manipulate drum sounds based on personal sound collections.

As compared to traditional drum synthesis techniques (e.g., subtractive, FM), control parameters are learned through an unsupervised process. Neural audio synthesis enables intuitive exploration of a generation space through a compact latent representation. A crucial requirement of a well-trained latent representation is its ability to generate audio with musically-meaningful control over the output. In the case of neural drum synthesis, previous methods have been proposed to enable continuous semantic control over generations by supplying high-level conditional information based on a chosen set of audio features [2–4].

In this paper, an unsupervised approach for deriving useful synthesis parameters is used as an alternative to selecting and extracting a fixed set of audio features. The proposed system is based on [1], and is extended with a conditional style-based generator network [5] for drum style (i.e., timbre) transformations. A mapping between a distribution of labelled drum sounds and a low-dimensional latent space is learned, providing high-level control over generation. The system operates directly on waveforms and leads to an unsupervised separation of high-level features (e.g., pitch, envelope shape, loudness), and enables intuitive, layer-wise control of the synthesis. Additionally, we introduce a novel approach to audio reconstruction through GAN inversion—a set of techniques for inverting a given input back into the latent space of a pre-trained GAN [6]. Using the predicted latent code, an input can be reconstructed by the GAN and manipulated using directions found in its latent space.

2. SYSTEM OVERVIEW

Figure 1 provides an overview of the proposed system for neural drum synthesis, which is achieved using four networks: (1) mapping network, (2) generator, (3) discriminator, (4) encoder. In a style-based GAN formalisation [5, 7], latent space $\mathcal{Z}$ is transformed into an intermediate space $\mathcal{W}$ using mapping network $M : \mathcal{Z} \rightarrow \mathcal{W}$. To facilitate the functionality of the mapping network, G is modified to take a constant value as input, and an intermediate latent vector $\mathbf{w} \in \mathcal{W}$ is provided to each upsampling layer. The mapping network M is a conditional multilayer perceptron, that learns to create disentangled features that are integrated at each upsampling block of the generator network G through adaptive instance normalisation (AdaIN). The generator is trained to output audio waveforms given a constant input and latent vector $\mathbf{w}$ for each upsampling block with affine transform A. Gaussian noise is added to each upsampling block using per-layer scaling factors B to introduce stochastic variation at each layer. Discrimi-

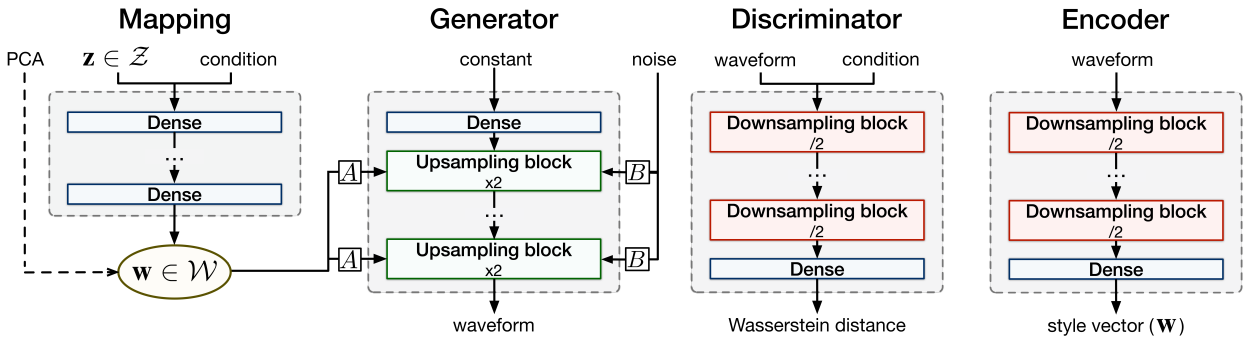

Figure 1. *Overview of proposed style-based drum synthesis system.*

nator D takes both a waveform and conditioning variable as an input and is trained to estimate the Wasserstein distance between the generated and observed distributions. All three networks are optimised simultaneously until G can produce waveforms that are indistinguishable from the observed training data.

The model was trained using a dataset of 9000 single drum hits selected from multiple commercially available sample libraries [1]. To increase the size of the dataset, individual drum sounds were augmented by pitch shifting between ± 3 semitones, resulting in a total of 63000 samples. The system uses WGAN-GP [8] training strategy to minimise the Wasserstein distance [9] between the training data distribution and generated data distribution. The model is trained using Adam optimiser [10] with a learning rate of 0.002 for the D and G networks and a minibatch size of 64 on an NVIDIA 2080ti GPU for 200k iterations.

2.1 Audio Inversion Network

Based on recent advances in the image domain [6], an encoder network E is trained separately to embed a given waveform into the intermediate latent space of the pre-trained generator. The predicted latent vectors are fed into the generator to synthesise drum sounds with similar characteristics to the input waveform. E replicates the architecture of the discriminator network D; however, the model is unconditional and its final dense layer has been modified to have has 128 output units to match the dimensionality of $\mathcal{W}$. Using a dataset of 10000 drum sounds generated with pre-trained network G, E is trained to minimise the MSE between the ground truth latent vectors and the predicted latent vectors.

2.2 Principal Feature Directions

PCA identifies patterns within the intermediate latent space $\mathcal{W}$, deriving a set of coordinates that emphasise variation in timbre. G feature controls are achieved by layer-wise perturbation along the principal directions. Following [11], principal axes of $p(\mathbf{w})$ are identified with PCA. N random vectors are sampled from Z and the corresponding $\mathbf{w}_l$ values are computed with M. At inference, PCA can be computed on $\mathbf{w}_l$ to obtain a basis V for $\mathcal{W}$. The principal components of $\mathbf{w}_l$ can then be used to control features of the generator by varying PCA coordinates scaled by control parameter g such that $\mathbf{w}' = \mathbf{w} + Vg$. Each entry g_i is initialised with zeros until modified by a user.

2.3 Synthesis Control Parameters

Style-based drum synthesis with GAN inversion allows the user to interact with the system parameters in three ways. In the first approach, drum synthesis can be controlled by sampling from the intermediate latent space and exploring the timbral characteristics with a preset number of style faders (i.e., PCA coordinates at each layer). In the second approach, the user can input a single drum sample to the encoder and modify its characteristics with style faders. In the third approach, the encoder can be used to reconstruct two arbitrary drum sounds, and various interpolation techniques may be incorporated for style transformation. Additionally, in each control method, Gaussian noise can be introduced into individual generator layers to modify the amount of stochastic variation within network layers. This is useful for shaping the high-frequency content of the generated drum sounds—especially for cymbals and snares.

3. EXPERIMENTAL RESULTS

For a demonstration of the generated drum sounds and synthesis parameters, we invite the reader to listen to the results and experiment with the code available on the accompanying website. [1] A python script is provided, which loads the networks pre-trained weights and enables utilisation of the synthesis control parameters described in Section 2.3. By traversing the latent space using the controls provided, a user can explore a space of different drum sounds. The audio examples demonstrate the systems capacity to generate a variety of different drum sounds, manipulate timbral characteristics and perform transformations between different inputs such as beatboxing and breakbeats. Although it is currently difficult to anticipate the exact effect that each principal direction has on the generated drum sounds, the directions correlate to changes in timbre, pitch and amplitude envelope.

[1] `https://jake-drysdale.github.io/blog/stylegan-drumsynth/`

4. REFERENCES

[1] J. Drysdale, M. Tomczak, and J. Hockman, "Adversarial synthesis of drum sounds," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2020.

[2] A. Ramires, P. Chandna, X. Favory, E. Gómez, and X. Serra, "Neural percussive synthesis parameterised by high-level timbral features," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 786–790, 2020.

[3] J. Nistal, S. Lattner, and G. Richard, "DrumGAN: Synthesis of drum sounds with timbral feature conditioning using generative adversarial networks," in *the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[4] M. Tomczak, M. Goto, and J. Hockman, "Drum synthesis and rhythmic transformation with adversarial autoencoders," in *Proceedings of the ACM International Conference on Multimedia*, pp. 2427–2435, 2020.

[5] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

[6] W. Xia, Y. Zhang, Y. Yang, J.-H. Xue, B. Zhou, and M.-H. Yang, "GAN inversion: A survey," *arXiv preprint arXiv:2101.05278*, 2021.

[7] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.8110–8119, 2020.

[8] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein GANs," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pp. 5767–5777, 2017.

[9] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," in *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 214–223, 2017.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[11] E. Härkönen, A. Hertzman, J. Lehtinen, and S. Paris, "GANspace: Discovering interpretable GAN controls," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pp. 9841–9850, 2020.

IMPROVED AUTOMATIC INSTRUMENTATION ROLE CLASSIFICATION AND LOOP ACTIVATION TRANSCRIPTION

Jake Drysdale * *and Jason Hockman*

Sound and Music Analysis Group, Digital Media Technology Lab
School of Computing and Digital Technology
Birmingham City University
Birmingham, UK
jake.drysdale@bcu.ac.uk

António Ramires * † *and Xavier Serra*

Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
antonio.ramires@upf.edu

ABSTRACT

Many electronic music (EM) genres are composed through the activation of short audio recordings of instruments designed for seamless repetition—or loops. In this work, loops of key structural groups such as bass, percussive or melodic elements are labelled by the role they occupy in a piece of music through the task of automatic instrumentation role classification (AIRC). Such labels assist EM producers in the identification of compatible loops in large unstructured audio databases. While human annotation is often laborious, automatic classification allows for fast and scalable generation of these labels. We experiment with several deep-learning architectures and propose a data augmentation method for improving multi-label representation to balance classes within the Freesound Loop Dataset. To improve the classification accuracy of the architectures, we also evaluate different pooling operations. Results indicate that in combination with the data augmentation and pooling strategies, the proposed system achieves state-of-the-art performance for AIRC. Additionally, we demonstrate how our proposed AIRC method is useful for analysing the structure of EM compositions through loop activation transcription.

1. INTRODUCTION

Affordable music production technologies (e.g., digital audio workstations) for incorporating and manipulating samples have democratised the EM creation process, allowing users with varying levels of musical knowledge to experiment in the creation of EM. A large majority of popular music is composed in this manner, inheriting some characteristics of EM, such as the use of samples, sequence-driven composition and a fixed tempo throughout the piece.

For sampled content, EM producers often rely on well-known sample libraries (e.g., Splice), which consist primarily of individual sounds and loops—short audio recordings (one, two or four bars in length) of instruments designed for seamless repetition [1]. Loops may be created by sequencing individual sounds or sampling a short musical phrase from a solo or polyphonic instrumental recording. These loops serve as the material from which music makers can generate EM compositions through various editing and combinatory processes (e.g., layering, splicing, rearranging). Figure 1 depicts a simplified representation of the EM creation process involving the layering and repeated activation of loops with different roles.

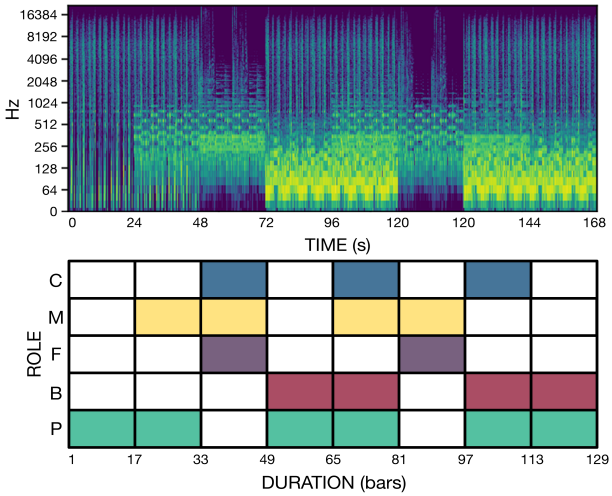

Figure 1: A simplified EM composition structure, built with five loop layers. Log-scaled STFT power spectrogram (*top*) and corresponding role activations: Chords (`C`), Melody (`M`), Sound Fx (`F`), Bass (`B`), and Percussion (`P`) at 4-bar intervals (*bottom*).

In this paper, we propose a system for automatic instrumentation role classification (AIRC) to label a loop by its specific function within a EM composition (e.g., drums, chords, melody, bass, sound Fx). System performance is measured through evaluation with state-of-the-art audio classification models and a data augmentation procedure that utilises common production techniques used in commercial music recordings. By estimating instrumentation roles for fixed-length segments of an EM composition, it is possible to retrieve an informative map of musical structure.

1.1. Related Work

In recent years, there has been an increased focus on research related to audio loops within the field of music information retrieval. There are several methods that exist for automated loop retrieval [2, 3, 4, 5, 6], and loop creation [7, 8, 9].

In addition, there are two recent methods proposed for loop activation transcription, a task that involves estimating the locations

* Both authors have contributed equally

† This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement Nº 765068, MIP-Frontiers.

in which loops occur throughout a piece of music. López-Serrano et al. [10] proposed a method for decomposing loop-based EM using non-negative matrix factorization deconvolution (NMFD) [11] to estimate spectral templates and rhythmic activations from magnitude spectrograms. Following this work, Smith et al. [12] propose an alternative method to discovering loop activations of EM using non-negative tensor factorization (NTF) [13]. While non-negative matrix factorization approaches allow for separation of mixed audio into the constituent loops of a music composition, they rely on non-varying repetitions of loops and do not optimise independence between learned loop representations.

As an alternative to the aforementioned approaches, which seek to identify the instrument within a loop and its associated activation, music auto-tagging is a multi-label classification task that may be used to denote presence of an attribute or instrument. Previous approaches have applied standard Convolutional Neural Networks (CNNs) to this task [14, 15], while recent work has focused on the application of musical knowledge for pitch and loudness invariance through musically-motivated filter shapes [16]. Won et al. [17] achieved state-of-the-art accuracy for auto-tagging by using data-driven Harmonic filters, a harmonically-stacked trainable representation to preserve time-frequency locality in convolution layers.

AIRC is a music auto-tagging task that estimates the presence of active instrumentation role groups (e.g., percussion, bass, melody, chords, sound Fx) within audio recordings. Research in AIRC has been facilitated by the development of the Freesound Loop Dataset (FSLD), a large public collection of loops and corresponding instrumentation role annotations from Freesound.[1] Ching et al. [18] benchmark AIRC performance of neural network and non-neural network models on the non-sequenced loops of FSLD, and achieve the current state-of-the-art performance using a Harmonic CNN [17].

1.2. Motivation

As EM production (i.e., creation, selection, manipulation) is guided heavily by aesthetic preferences, producers often select sounds based on their function within loops. With a wide range of traditional and synthesized timbres from which to select, instruments are often utilised outside traditional roles. We thus follow the AIRC problem formalisation as represented in Ching et al. [18], which associates instrumental roles with short loops. We expand on this approach by applying AIRC to full EM compositions, in which multiple instrumentation roles (e.g., percussion, melody, bass playing together) are often active. [18] observed that accuracy and bias were reduced by the overuse of single labels, due to limited coverage of multi-label annotations in the FSLD. To mitigate this imbalance, we introduce a novel data augmentation technique to balance classes and experiment with several deep-learning architectures and pooling operations, resulting in a state-of-the-art AIRC system.

We then demonstrate the usefulness of AIRC in EM structural analysis by comparing our system with previous approaches for loop activation estimation. Additionally, the proposed AIRC system is shown to derive key structural information from full-length EM compositions in the form of instrumentation role activation maps, which would be of use in tasks such automatic DJing [19], mashups [20], and loop creation [4]. Finally, we show that the system is capable of identifying percussion only passages and then compare it against the previous state-of-the-art for breakbeat identification. For reproducibility, we provide open-source code for the proposed data augmentation method and AIRC system.

The remainder of this paper is structured as follows: Section 2, presents the proposed method for AIRC and loop activation transcription. Evaluation methodology and the datasets used in this study are detailed in Section 3 and the results and discussion are provided Section 4. Conclusions and suggestions for future work are presented in Section 5.

2. METHODOLOGY

In this study, several CNN architectures are evaluated to identify the best system for AIRC. Each architecture utilises different configurations of front-end filter shapes to learn a representation from spectrograms and pooling operations that derive the final predictions by summarising the information learned by the network.

As the data employed in AIRC contains different types of musical audio, from tonal melodies to noise-like sound Fx, this motivates the experimentation of architectures aimed at different sound classification tasks. Three front-end filter shapes are used: general domain square filters, vertical filters [16] tailored towards classifying the timbre of melodic instruments, and previous state-of-the-art for AIRC—harmonic band-passfilters which capture harmonic characteristics.

To improve the AIRC predictions, two methods for summarizing the information learned in the final convolutional layers of a CNN are investigated. The standard approach to is to use global max-pooling (GMP); however, this infers strict assumptions about the label characteristics of the data. In the closely related field of sound event detection, auto-pooling has been proposed to automatically learn the best suited operation by interpolating between max-, mean-, or min-pooling during training. We implement both GMP and auto-pooling and compare their performance for the task of AIRC.

2.1. Implementation

Audio is input into the networks as a spectrogram representation, from which features are extracted through convolutional layers. Output predictions return values between $[0., 1.]$ depicting the presence of active instrumentation roles.

For each network, the input layer is a four-dimensional tensor $t \in \mathbb{R}^{b \times w \times h \times c}$, with batch size b, number of frames w, number of frequency bins h, and channels c. Following [16], each model uses L2-norm regularization of filter weights to encourage loudness invariance with the exception to the harmonic CNN-based models, which use a weight decay of $1e$–4 [17].

2.1.1. *Vertical filter network*

The vertical filter network (`VF-CNN`) is based on the *multi-layer* architecture in [16] for musical instrument recognition. Figure 2 provides an overview of the `VF-CNN` configuration. The input spectrogram is set to be of size 500×128 to accommodate for longer observations of audio loops (see Section 2.2). The front-end utilizes several vertical convolution filter sizes (black rectangles in Figure 2) to efficiently model timbral characteristics present in the spectrogram. Custom filter sizes are used to capture both wide (e.g., bass, chords) and shallow spectral shapes (e.g, percussion).

[1] `https://freesound.org/`

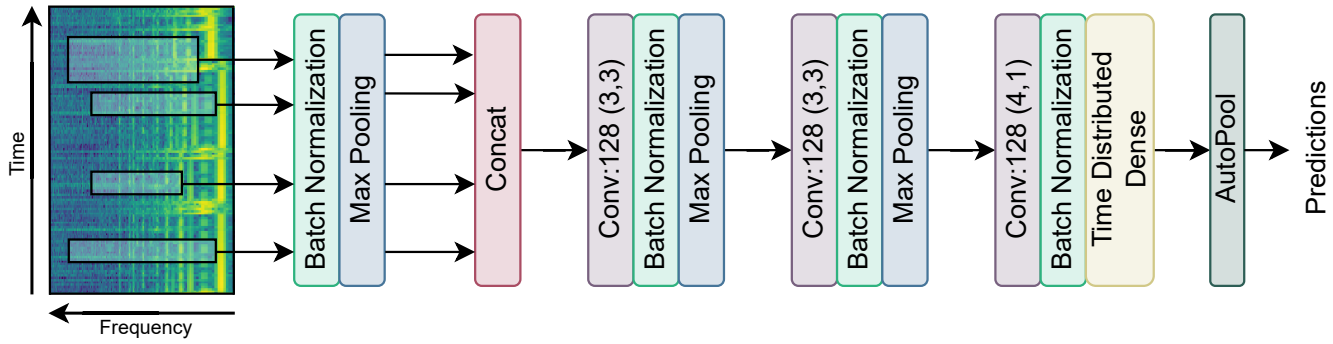

Figure 2: Block diagram showing the configuration of the vertical filter network with auto-pooling.

The numbers and sizes of filters used in the front-end are as follows: 128 filters of sizes 5×1 and 80×1; 64 filters of sizes 5×3 and 80×3; and 32 filters of sizes 5×5 and 80×5.

All convolutions in the front-end use *same* padding, and max-pooling is applied to obtain a 16×16 summary of each feature map. This is followed by two 2-D convolutional layers with batch normalisation [21] and exponential linear unit (ELU) [22] activation functions. The first 2-D convolutional layer is followed by strided $(2, 2)$ max-pooling. After the final 2-D convolutional layer, we experiment with two pooling operations to summarise the information learned by previous layers prior to predictions (see Subsection 2.1.4).

2.1.2. *Square filter network*

The square filter network (`SF-CNN`) contains four 2-D convolutional layers with 128 small-rectangular filters of size 3×3 and *same* padding. After each convolutional layer, batch normalization is applied with an ELU [22] activation function. Each convolutional layer is followed by strided $(2, 2)$ max-pooling, with the exception of the final convolutional layer, which also uses one of the two summarization pooling operations described in Section 2.1.4.

2.1.3. *Harmonic CNN*

In [18], AIRC was approached using a CNN with a data-driven harmonic filter-based front-end (`H-CNN`) [17]. We re-implement this architecture and use it as a baseline to test our proposed models. The input t is passed through a set of triangular band-pass filters to obtain a tensor representing it as six harmonics. Harmonic structure is captured by treating the harmonics as channels and processed by a 2-D CNN. The CNN consists of seven convolution layers and a fully connected layer. All but the final convolutional layer is followed by 2×2 max-pooling, batch normalization and a ReLU activation function. Global max-pooling is applied to the final convolutional layer. The output layer is a 5-way fully-connected layer with, a sigmoid activation function and a 50 % dropout.

2.1.4. *Summarization Pooling*

We consider two pooling operations for summarizing the information learned in the final convolutional layers: auto-pooling and standard global max-pooling.

Auto-pool is a trainable pooling operator capable of adapting to data characteristics by interpolating between min-, max-, or average-pooling [23]. For the configurations that use auto-pooling, the final convolutional layer uses as kernel size (4,1). This is followed by batch normalisation and a time-distributed dense layer with a sigmoid activation function and r output nodes, where r is equal to the number of classes. The output of the time-distributed dense layer is fed to the auto-pooling operation, which produces the final predictions.

For configurations that use GMP, the final convolutional layer is summarised with global max-pooling and then fed to a fully-connected output layer consisting of r output nodes, sigmoid activation functions and a 50% dropout.

2.1.5. *Loss function*

The loss function used for updating the parameters of each model is binary cross-entropy (BCE). BCE can be calculated as:

$$\mathrm{BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(\, p(y_i)) + (1-y_i) \cdot \log\,(1-p(y_i)), \quad (1)$$

where *N* represents the number of examples in the training set and $p(y_i)$ s the predicted probability of the i^{th} example.

2.2. Network Training

Input audio is pre-processed through resampling and conversion to a spectrogram representation. Audio loops are resampled to 16kHz and the short-time Fourier transform (STFT) of each loop is calculated using a window size of 512 samples and a hop size of 256 samples. For the `H-CNN`, magnitudes of STFT are provided as input to the model. For the `SF-CNN` and `VF-CNN`, the inputs are log-scaled Mel spectrograms with 128 Mel-frequency bands.

All models are trained using the Adam optimiser [24] with a learning rate $1e$–4, where each iteration takes a mini-batch of 8 examples. All weights are initialized using He's constant [25] to promote equalized learning. Early stopping was used to complete the training once the model performance ceases to improve over 15 epochs. The epoch that achieves the best accuracy on the validation set is used for testing.

2.3. Loop Activation Transcription

Loop activation transcription involves predicting the loop activations of instrumentation roles as they occur over time. Taking ad-

vantage of the grid-based structure and consequently fixed tempo of loop-based EM, we are able to use the proposed AIRC system to analyse the loop structure of a given composition. The AIRC system enforces separation between roles by design and does not rely on loops being an exact repetition of themselves, thus making it robust to variation such as automation and resequencing.

Instrumental role predictions for full EM compositions are obtained by passing an audio file into the AIRC system in 4-bar segments and assessing output activations. By segmenting a full-length EM composition into 4-bar loops, on which we then perform AIRC, instrumentation role activations may be derived for each loop, resulting in a form of EM transcription.

3. EVALUATION

The AIRC model presented in Section 2 is assessed through two evaluations to determine: 1) AIRC performance using the various configurations and augmented version of the FSLD, and 2) performance for loop activation transcription.

3.1. Automatic Instrumentation Role Classification

3.1.1. Evaluation Methodology

The architectures (i.e., `VF-CNN`, `SF-CNN`, and `H-CNN`) and pooling strategies (i.e., auto-pooling and GMP) presented in Section 2 are evaluated in order to determine the optimal configuration for AIRC. Following [18, 26], we use two sets of performance measurements: Area under receiver operating characteristic curve (ROC-AUC) and area under precision-recall curve (PR-AUC). The metrics were calculated on the test set for each of the models under evaluation.

In [18], the authors also calculate the F1 score; however, we omit this evaluation metric as it depends on a decision threshold applied to the per-class output scores; whereas, ROC-AUC and PR-AUC measure model performance globally, integrating all possible thresholds.

3.1.2. Augmented Freesound Loop Dataset

To train and evaluate the different models we use the Freesound Loop Dataset (FSLD) [27], comprising of various loops uploaded to Freesound [28] under Creative Commons licensing. Of the various annotations present within the FSLD, we use tempo, key and loop instrumentation roles. The most important of which is the instrumentation role—a multi-label annotation for which the possible roles are: percussion, bass, chords, melody, sound Fx and vocals.

The original FSLD contains 2936 loops, of which 1531 have only one instrumentation role and 1405 have more than one. As can be seen from the class distribution in Table 1, the classes in this dataset are heavily imbalanced.

Percussion	54.95	Fx	24.80
Bass	19.10	Melody	21.31
Chords	11.90	Vocal	2.29

Table 1: Distribution (%) of instrumentation roles in FSLD.

In order to adapt this dataset to our task, we apply modifications to the data. We first remove all vocal loops as they do not provide sufficient training and testing material. All remaining loops are time-stretched to 120 beats per minute (BPM). Longer loops are cropped to a length of 4 bars (i.e., 8 secs), while loops shorter than 4 bars are cropped to either 1 or 2 bars and repeated to a length of 4 bars. We separate loops which have multiple instrumentation roles from those which only have one, and randomly select 70% of each for training and 30% for validation and testing. From the latter split, 60% are used for testing and 40% for validation.

Besides using the previously described training set of the FSLD, we applied a data augmentation procedure to handle the main imbalance issues on the dataset. These are 1) the lesser presence of loops with more than one instrumentation role (i.e., multi-label) compared to the ones with just one role (i.e., single-label) and; 2) the number of loops for each instrumentation role class, shown in Table 2.

The data augmentation procedure utilises common production techniques that are used in commercial music recordings including key matching, tempo matching and the use of audio Fx such as distortion, reverb and chorus.

Percussion	929	Fx	222
Bass	92	Melody	174
Chords	102		

Table 2: Distribution of the loops with only one instrumentation role in FSLD.

To balance the number of loops per class, we use an augmentation methodology similar to the one proposed in Ramires et al. [29]. The loops are processed through several effects, including delay, bitcrusher, chorus, flanger, reverb, tube saturation and pitch-shifting, resulting in 1000 loops for each of the r classes under observation ($r = 5$), totalling 5000 loops.

We create multi-role data by overlapping loops from each augmented single label class such that all single and combined classes contain the same number of loops. We start by calculating the possible combinations $\binom{r}{k}$, where k is the number of instrumentation roles in the combination ($2 \leq k \leq 5$). To balance the dataset in both the number of loops per instrumentation role and in k, the number of augmented loops (5000) is divided by $\binom{r}{k}$ to obtain the number of loops required for each combination (e.g., for k=2, $5000/\binom{r}{k} = 500$ for each combination of roles). The final loops are then created by harmonically combining the single instrumental role loops. We combine only loops with compatible modes (e.g., Major with Major), and pitch shift the selected loops to their average key.

Discarding the original multi-label loops of the training set, this process results in a total of 25000 loops that can can used for training. In order to evaluate the effect of this augmentation procedure (Aug), we compare the accuracy of the models trained with those trained with the original dataset (FSLD) on the same test and validation data.

Percussion	27.59	Fx	23.17
Bass	20.33	Melody	18.15
Chords	10.77		

Table 3: Distribution (%) of instrumentation roles in the test set.

Model	Dataset	Pooling	Param.	PR-AUC	ROC-AUC	Bass	Fx	Perc.	Chords	Melody
H-CNN	Aug	GMP	3619986	59.18	77.34	40.30	57.05	94.60	47.92	56.01
H-CNN	Pure	GMP	3619986	61.83	80.39	53.65	42.21	94.10	60.30	58.89
VF-CNN	Aug	GMP	1098869	65.60	80.99	47.11	64.92	97.62	63.11	55.22
VF-CNN	Pure	Auto	1102394	66.98	82.52	57.59	66.43	95.75	50.37	64.77
VF-CNN	Aug	Auto	1102394	67.47	81.40	46.18	67.10	97.05	56.13	**70.89**
SF-CNN	Aug	Auto	**313674**	68.15	82.19	55.12	68.30	**98.03**	58.81	60.49
SF-CNN	Aug	GMP	445701	68.40	81.59	**62.21**	59.80	95.93	62.39	61.68
SF-CNN	Pure	GMP	445701	68.74	83.83	58.72	63.11	95.97	64.74	61.14
VF-CNN	Pure	GMP	1098869	70.62	**85.72**	53.83	**71.73**	97.84	64.90	64.78
SF-CNN	Pure	Auto	**313674**	**71.28**	85.12	57.76	59.18	95.98	**73.20**	70.30

Table 4: AIRC performance (%) and model size for each configuration, where bold indicates highest scores.

3.2. Loop Activation Transcription

3.2.1. Evaluation Methodology

To investigate the capacity of the AIRC system for transcribing loop activations in EM compositions, we compare all the AIRC configurations (Section 2). The best performing configurations are then compared with the results of the previous approach to loop activation transcription by Smith et al. [12].

As in [30, 12], we evaluate the loop activation predictions against a ground truth in terms of accuracy. As accuracy expects a binarised transcription, we use a repeated k-fold cross validation together with a grid search to identify the best threshold for binarising the predictions of each role. In order to investigate the generalization of the proposed models, we use 2-fold cross validation repeated 10 times, where one fold is used as a validation set to identify thresholds and the other is reserved for computing accuracy against the ground truth. Thresholds for each class are identified by performing a grid search over a range between 0.01 and 1 with a step size of 0.01, then selecting the thresholds which provide highest accuracy on the validation set.

In [12], approaches which require the downbeat tracking are considered *guided*. As our proposed approach requires BPM annotations for time-stretching, we only compare our models with the *guided* algorithms.

3.2.2. Dataset

We apply our proposed models to the dataset used in [30, 12]. The dataset consists of simplified EM compositions built by generating templates similar to the ones in Figure 1 with 4-bar loops. We refer to this as the *Artificial* dataset for the reason that the the loops are repeated without variation, which would usually be achieved in professional music through DAW techniques, such as automation and resequencing.

The automatic arrangement method provided in [12] is used to build 21 music compositions with seven genres and three templates–*composed*, *factorial* and *shuffled factorial*. For the *composed* template, loops are introduced and removed in an iterative manner. The *factorial* template contains all possible combinations of loops, arranged iteratively. The *shuffled factorial* template contains the same loop combinations, with shuffled ordering. *Factorial* and *shuffled factorial* datasets are useful for seeing how the models perform on all of the loop combination possibilities for the *Artificial* dataset, whereas *composed* layout is more representative of typical EM compositions in regards to the way that loops are iteratively introduced and removed throughout the composition.

Following the AIRC procedure, compositions are time-stretched from their annotated tempo to 120BPM and divided into 4-bar loops, which are provided as input to the AIRC systems.

4. RESULTS & DISCUSSION

4.1. Automatic Instrumentation Role Classification

The models are evaluated using use the marco-average (MA) of the PR-AUC and of the ROC-AUC as a global metric. For individual instrumentation roles, we only show the PR-AUC. Due to the imbalance of the FSLD, which also affects the test set (Table 3), MA is used to provide an average accuracy over each class.

Table 4 presents the results of our AIRC experiment for the models discussed in Section 2, in which each model is presented in ascending order of their average PR-AUC. The ROC-AUC performance measure is consistently higher than PR-AUC; however, this metric can lead to over-optimistic scores when the dataset is unbalanced [31].

The best performing models *w.r.t* PR-AUC, are the SF-CNN with auto-pooling (71.28%) followed by the VF-CNN with GMP (70.61%). Both models surpass the current state-of-the-art, H-CNN trained on FSLD (61.82%) by a substantial margin. The SF-CNN mostly performs better than its VF-CNN counterpart. Vertical filters have been demonstrated to produce comparatively better results with tonal musical audio [32]; however, the results of our evaluations suggest that square filters generalise better to the non-standard types of audio associated with EM.

The overall best performing model in terms of PR-AUC is the SF-CNN with auto-pooling trained on the Pure dataset. However, by closely inspecting the results achieved for individual instrumentation roles, it can be seen that it surpasses by almost 10% the PR-AUC achieved by other models in the *Chords* class, while not achieving such a high result in *Bass*, *Fx* and *Percussion*.

The highest performing instrumentation role for all models is *Percussion*, which was expected due to this role having the largest number of examples in the FSLD dataset. The roles that generally perform worst are *Bass* and *Chords*, which have the smallest number of examples in the FSLD. The performance of *Bass* has a considerable increase when using a combination of the SF-CNN with GMP and augmented data. Additionally, *Chords* performs significantly better when using the SF-CNN and auto-pooling configuration trained with the Pure dataset.

The best three performing models in terms of PR-AUC are trained on the Pure dataset, followed by the Augmented one. However, it can be seen that the *Bass*, *Percussion* and *Melody* roles tend

to benefit from training with the Augmented dataset. As the configurations perform better for different classes, it is possible to use a combination of the models for classifying individual instrumentation roles. This combination would lead to an average PR-AUC of 75,213%, substantially surpassing each model.

4.2. Loop Activation Transcription

Table 5 presents the loop activation transcription results using the AIRC configurations (Section 2) to transcribe the compositions in the *Artifical* dataset. Each model is presented in ascending order of their mean classification accuracy over the instrumentation roles. Additionally, Table 5 provides the classification accuracy for each individual role (*Bass*, *Drums*, *Fx* and *Melody*).

Model	Data	Pooling	Mean	Bass	Drums	Fx	Melody
H-CNN	Pure	GMP	75.1	71.8	96.1	55.6	76.7
H-CNN	Aug	GMP	79.5	53.1	95.8	81.6	87.6
VF-CNN	Pure	Auto	80.2	63.7	98.6	63.1	**95.3**
VF-CNN	Pure	GMP	80.9	69.0	99.3	65.8	89.4
SF-CNN	Pure	Auto	81.0	66.9	97.3	71.6	88.4
SF-CNN	Pure	GMP	81.8	69.2	**100.0**	63.4	94.6
VF-CNN	Aug	Auto	82.5	**74.2**	99.7	79.7	76.6
VF-CNN	Aug	GMP	84.7	71.7	**100.0**	75.7	91.4
SF-CNN	Aug	GMP	86.2	71.7	**100.0**	79.6	93.2
SF-CNN	Aug	Auto	**86.9**	68.3	**100.0**	**85.7**	93.4

Table 5: Loop activation transcription accuracy (%) results for AIRC configurations, where bold indicates highest scores.

The overall best performing model uses the SF-CNN with auto-pooling configuration trained using the augmented dataset (86.9%) followed by the SF-CNN with GMP (86.2%). For this task, models trained with the augmented dataset generally appear to outperform those trained with Pure dataset, which could be due to the fact that the augmentation process ensures there is a balanced distribution of all possible role combinations and it is common in the compositions for several roles to be active in a single loop. Drums are classified most accurately for all model configurations with four models achieving 100% accuracy. This is expected as *percussion* has the largest number of samples in the FSLD dataset, and is usually the most prominent element in EM compositions. In some cases, the VF-CNN configuration seems to improve performance of *Melody* and *Bass* roles, which could suggest that the classification of roles containing melodic instruments benefit from using vertical filters at the front end of the system.

Figure 3 presents loop activation transcription results for the three template variations using our two best performing AIRC configurations (i.e., SF-CNN-AUTO and SF-CNN-GMP) compared with the NTF [12] and NMFD [10] methods previously proposed for this task.

On a glance, we can see our architectures out perform the previous methods in regards to accuracy for the *composed* layout, with SF-CNN-GMP (red) achieving the highest score. NTF (blue) achieves the best performance for the *factorial* layouts closely followed by our SF-CNN-AUTO architecture. Furthermore, the AIRC system has a considerably faster runtime than NTF (∼30 secs per composition) and NMFD (∼10 mins per composition). Predictions for a full EM composition are calculated in under a second using AIRC, which could be beneficial when analysing large collections of music in DJ software. As mentioned in [12], an additional shortcoming of the NTF and NMFD approaches is that the algorithms depend on loop roles not co-occurring throughout the composition. The proposed AIRC approach enforces independence between the different roles, thus making it more suitable for transcribing loop activations of real-world EM compositions, in which loops often vary through automation and resequencing.

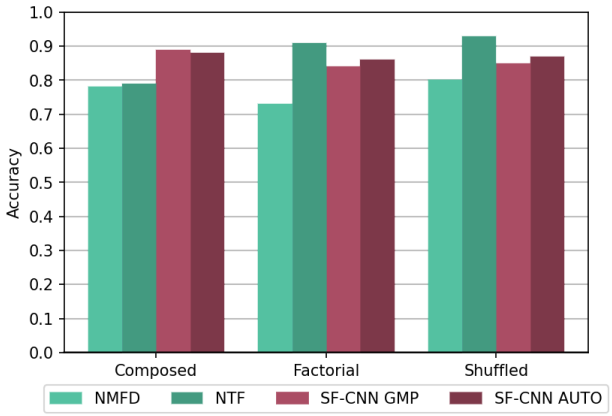

Figure 3: Loop activation transcription accuracy scores.

4.3. Real-world Scenario

Our approach to loop activation transcription with AIRC can be applied to full-length, professionally produced EM, which has not been explored in previous literature.

An *instrumentation role activation map* (IRAM) of the EM composition *Joyspark* (2020) by Om Unit[2] using the proposed method for loop-based EM structure analysis (Section 3.2) is presented in Figure 4. For visualisation and comparison, we show a log-scaled STFT power spectrum of the EM composition above the IRAM. The IRAM allows us to visualise activations for each role over the duration of the EM composition, where each square is a measurement of four bars. Furthermore, we can see how each role develops throughout the EM composition. For example, the melody role activations progressively increase between bars 1–41, which corresponds with a synthesizer arpeggio that is gradually introduced by automating the cut-off frequency of a low-pass filter. Additionally, the chord role activations increase between bars 1–49 in correlation with the chords in this section that gradually increase in volume. Activations for the percussion role also correlate well with the composition as can be seen between bars 49–81 and 97–129—the only sections that contain percussion. Finally, the key structural sections of the composition are easily identifiable. For example, the introduction to the composition (bars 1–49) begins relatively sparse in the composition and IRAM; whereas, bars 49–81 and 97–129 are quite clearly the *core* of the piece—that is, the most energetic sections of the composition typically established by the *drop* [33].

Additionally, the transcription enabled by our system could help EM producers identify sections of music that contain specific roles. For example, this would be useful for finding breakbeats (i.e., percussion-only passages) in digital music recordings [3].

[2] https://omunit.bandcamp.com

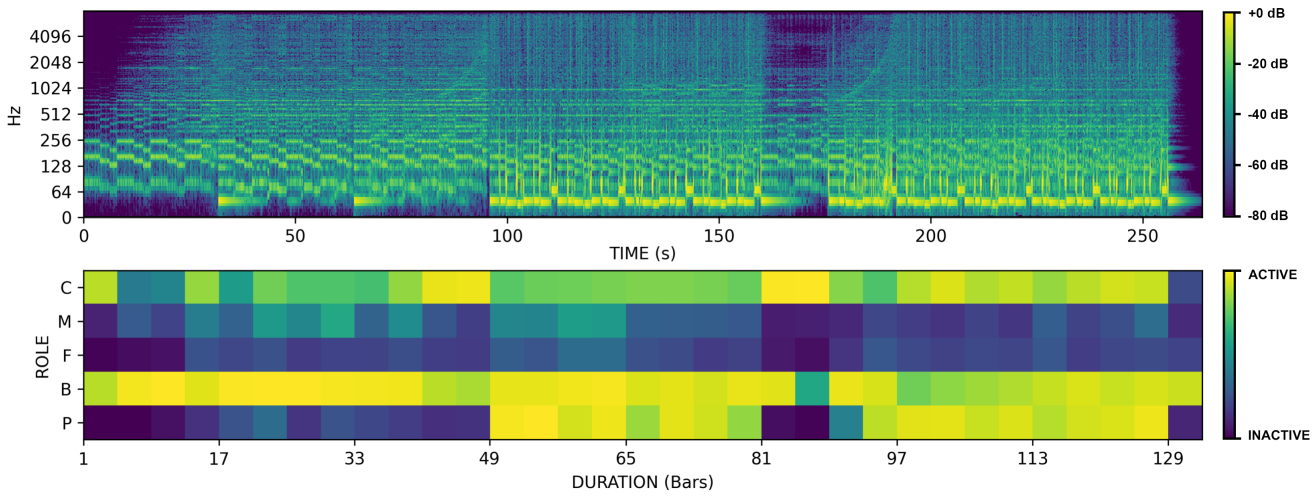

Figure 4: Estimated loop activation structure of *Joyspark* (2020) by Om Unit using our proposed model. Log-scaled STFT power spectrogram of the EM composition (*top*) and estimated templates corresponding to the loop activations showing predictions for each class: Chords (`C`), Melody (`M`), Sound Fx (`F`), Bass (`B`), and Percussion (`P`) at 4-bar intervals (*bottom*).

5. CONCLUSIONS

In this study, we have introduced a system for automatic instrument role classification of loops that utilises a novel data augmentation method and CNN-based architecture with auto-pooling. The evaluation results show that we outperform previous state-of-the-art performance in AIRC, allowing for a more reliable transcription of loops in unstructured collections of audio. Furthermore, we have introduced a deep learning approach for estimating the structure of loop-based electronic music and compared it with previous loop activation detection methods. Our approach achieves comparable results while achieving a considerably faster computation time.

The IRAM derived from our system has many potential use cases in music production and performance. MIR tasks that rely on structural information could benefit from this transcription (e.g., automatic DJing [19], music mashups [20]). The IRAM could be used as a visual aid for DJs to anticipate upcoming sounds (e.g., drums, bass) or to help to identify key structural events in EM [33].

A possible direction for future research in this area would be to train the system using a smaller timescale (e.g., 1-bar measures) to achieve higher resolution transcription of instrumentation role activations. Additionally, as no annotations for ground-truth instrumentation roles exist for real-world EM compositions, future work could involve annotating a corpus of these recordings for the evaluation of this task.

6. ACKNOWLEDGMENTS

The authors would like to kindly thank Patricio López-Serrano and Jordan B. L. Smith for the fruitful discussions and access to the loop activation transcription datasets and Eduardo Foncesca for the guidance on the implementation of auto-pooling.

7. REFERENCES

[1] Glenn Stillar, "Loops as genre resources," *Folia Linguistica*, vol. 39, no. 1-2, pp. 197 – 212, 2005.

[2] Olivier Gillet and Gaël Richard, "Drum loops retrieval from spoken queries," *Journal of Intelligent Information Systems*, vol. 24, no. 2, pp. 159–177, 2005.

[3] Patricio López-Serrano, Christian Dittmar, and Meinard Müller, "Finding drum breaks in digital music recordings," in *Proceedings of the International Symposium on Computer Music Multidisciplinary Research*, 2017, pp. 111–122.

[4] Zhengshan Shi and Gautham J Mysore, "Loopmaker: Automatic creation of music loops from pre-recorded music," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–6.

[5] Jordan BL Smith, Yuta Kawasaki, and Masataka Goto, "Unmixer: An interface for extracting and remixing loops.," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, 2019, pp. 824–831.

[6] Bo-Yu Chen, Jordan BL Smith, and Yi-Hsuan Yang, "Neural loop combiner: Neural network models for assessing the compatibility of loops," in *Proceedings of the 21th International Society for Music Information Retrieval Conference*, 2020.

[7] Diogo Cocharro, George Sioros, Marcelo F. Caetano, and Matthew E. P. Davies, "Real-time manipulation of syncopation in audio loops," in *Music Technology meets Philosophy - From Digital Echos to Virtual Ethos: Joint Proceedings of the 40th International Computer Music Conference and the 11th Sound and Music Computing Conference*, 2014.

[8] Guillaume Alain, Maxime Chevalier-Boisvert, Frederic Osterrath, and Remi Piche-Taillefer, "Deepdrummer : Generating drum loops using deep learning and a human in the loop," in *Proceedings of The 2020 Joint Conference on AI Music Creativity*, 2020, pp. 81–91.

[9] Pritish Chandna, Antonio Ramires, Xavier Serra, and Emilia Gómez, "Loopnet: Musical loop synthesis conditioned on intuitive musical parameters," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing Proceedings*, 2021.

[10] Patricio López-Serrano, Christian Dittmar, Jonathan Driedger, and Meinard Müller, "Towards modeling and decomposing loop-based electronic music.," in *Proceedings of the 17th International Society for Music Information Retrieval Conference*, 2016, pp. 502–508.

[11] Paris Smaragdis, "Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs," in *Proceedings of the International Conference on Independent Component Analysis and Signal Separation*, 2004, pp. 494–499.

[12] Jordan BL Smith and Masataka Goto, "Nonnegative tensor factorization for source separation of loops in audio," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing Proceedings*, 2018, pp. 171–175.

[13] D. FitzGerald, M. Cranitch, and E. Coyle, "Sound source separation using shifted non-negative tensor factorisation," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, 2006.

[14] Keunwoo Choi, George Fazekas, and Mark Sandler, "Automatic tagging using deep convolutional neural networks," in *Proceedings of the 17th International Society for Music Information Retrieval Conference*, 2016.

[15] Sander Dieleman and Benjamin Schrauwen, "End-to-end learning for music audio," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing Proceedings*, 2014, pp. 6964–6968.

[16] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra, "Timbre analysis of music audio signals with convolutional neural networks," in *Proceedings of the 25th European Signal Processing Conference*. IEEE, 2017, pp. 2744–2748.

[17] Minz Won, Sanghyuk Chun, Oriol Nieto, and X. Serra, "Data-driven harmonic filters for audio representation learning," *2020 IEEE International Conference on Acoustics, Speech and Signal Processing Proceedings*, pp. 536–540, 2020.

[18] Joann Ching, António Ramires, and Y. Yang, "Instrument role classification: Auto-tagging for loop based music," in *Proceedings of The 2020 Joint Conference on AI Music Creativity*, 2020, pp. 196–202.

[19] Len Vande Veire and Tijl De Bie, "From raw audio to a seamless mix: creating an automated dj system for drum and bass," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2018, no. 1, pp. 1–21, 2018.

[20] Matthew EP Davies, Philippe Hamel, Kazuyoshi Yoshii, and Masataka Goto, "Automashupper: An automatic multi-song mashup system.," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2017.

[21] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015, pp. 448–456.

[22] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *4th International Conference on Learning Representations*, 2016.

[23] Brian McFee, Justin Salamon, and Juan Pablo Bello, "Adaptive pooling operators for weakly labeled sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 2180–2193, 2018.

[24] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations*, 2015.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[26] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik Schmidt, Andreas Ehmann, and Xavier Serra, "End-to-end learning for music audio tagging at scale," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2017.

[27] Antonio Ramires, Frederic Font, Dmitry Bogdanov, Jordan B. L. Smith, Yi-Hsuan Yang, Joann Ching, Bo-Yu Chen, Yueh-Kao Wu, Hsu Wei-Han, and Xavier Serra, "The freesound loop dataset and annotation tool," in *Proceedings of the 21st International Society for Music Information Retrieval*, 2020.

[28] Frederic Font, Gerard Roma, and Xavier Serra, "Freesound technical demo," in *ACM International Conference on Multimedia*, 2013, pp. 411–412.

[29] António Ramires and Xavier Serra, "Data augmentation for instrument classification robust to audio effects," in *Proceedings of the International Conference on Digital Audio Effects*, 2019.

[30] Patricio López-Serrano, Christian Dittmar, Jonathan Driedger, and Meinard Müller, "Towards modeling and decomposing loop-based electronic music," in *Proceedings of the 17th International Society for Music Information Retrieval Conference*, 2016.

[31] Jesse Davis and Mark Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 233–240.

[32] Jordi Pons, *Deep neural networks for music and audio tagging*, Ph.D. thesis, Universitat Pompeu Fabra, 2019.

[33] Karthik Yadati, Martha A Larson, Cynthia CS Liem, and Alan Hanjalic, "Detecting drops in electronic dance music: Content based approaches to a socially significant music event.," in *Proceedings of the 15th International Society for Music Information Retrieval Conference*, 2014, pp. 143–148.