Towards controllable neural audio synthesis: An exploration of sound effects creation with generative models

YUNYI LIU

Doctor of Philosophy

Supervisor: Dr. Craig Jin
Associate Supervisor: Densil Cabrera

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy

School of Electrical and Information Engineering
Faculty of Engineering
The University of Sydney
Australia

30 August 2025

Abstract

Sound effects are crucial components in enhancing the auditory experience in media such as film, television, and video games. Traditionally, these effects are created using two primary methods: Foley recording, where sound is physically performed to match onscreen actions, and digital sound processing (DSP), which manipulates audio signals with algorithms to achieve desired sounds. While effective, these methods come with limitations in terms of scalability and the ability to rapidly prototype sounds.

Recent advancements in deep learning have introduced a new paradigm for sound effects generation through neural audio synthesis. This approach utilizes generative models to produce sound effects from learned audio features automatically. However, a significant drawback of this technology is the lack of fine-grained control over the sound output, as generative models typically provide fewer parameter adjustments compared to traditional DSP methods. This limitation poses challenges in achieving specific auditory outcomes necessary for creative sound design.

This thesis explores neural audio synthesis in sound effects generation. The data scarcity in the domain of sound effects has been a well-known issue, which makes it challenging for data-driven approaches towards learning and synthesizing. This thesis aims to study controllable and diverse sound effects generation under the setting of a limited audio dataset without requiring huge computation resources. To this end, I focus on three aspects towards controllable neural audio synthesis:

- This thesis starts by exploring the quality and diversity trade-off in the synthesis of audio using generative models. Via a proposed regularizing technique, it aims to improve the diversity of sound generation while maintaining the overall quality of synthesis.
- This thesis then studied implicit timbre control via a compressed time-varying latent representation in a DDSP+VAE architecture. This proposed approach is capable of

varying timbre attributes in the time domain, and a case study of voice-controlled sound effects synthesis is presented.

- Finally, two conditioning approaches were proposed to help control the timbre (or style) of sound effects by leveraging pre-trained audio presentation models while relying only on discrete class information. This research demonstrates the possibility of mixing/interpolating sound timbres across different categories.

Acknowledgements

This thesis was supported by the Electrical and Information Engineering School of the University of Sydney. I'm grateful for the hardware and software support provided by the University of Sydney. Part of the research was done during my internship at Dolby Laboratory in Sydney. I am very fortunate to join Dolby Advanced Technology Group (ATG) as an intern and be given the opportunity to complete a research project within the team.

I wish to thank especially to my supervisor, Dr Craig Jin, for granting me the opportunity to be his PhD student. He has been extremely responsible in providing guidance and mentorship on how to conduct academic research as a standalone researcher. I'm also grateful to our lab members for supporting me in participating in listening experiments and providing feedback on my research.

I would also like to thank my partner, Sia, who has always been with me during my PhD. I am extremely grateful for the accompaniment of my partner when I was facing the challenges during my candidature.

I would also like to express my gratitude to Adrián Barahona-Ríos, who has been a senior PhD candidate working in a similar area with me at the University of York. We have been having regular meetings about our research during my candidature, and his advice and support have been extremely helpful for my research.

I am grateful for everyone who has participated in the human listening tests as required for the analysis of my PhD thesis. Your participation and suggestions are very helpful for my research.

I would like to express my deepest gratitude to my parents, whose unconditional support and belief in my abilities have been my cornerstone. From the earliest days of my education to this very moment, you have sacrificed immensely, providing not just financial and emotional support, but also the freedom to pursue my passions. Your endless patience and understanding

during the challenging periods of this PhD journey have been a source of immense comfort and strength.

Finally, to everyone who has contributed to my academic and personal journey, directly or indirectly, I am sincerely thankful. Your support has been essential to the completion of this work.

Authorship Attribution Statement

The following papers are published or submitted as first-author papers where I have made substantial work in ideation, research, experiments, analysis and writing. All of the publications are the author's original work and attested by the thesis supervisor.

- **Chapter 4** has been published as a conference paper "Conditional Sound Effects Generation with Regularized WGAN" at 20th Sound and Music Computing Conference 2023, July. In this work I explored a specific training framework for neural audio synthesis and studied the performance of this approach. I did the experiment, analysis and writing of the paper. I presented this work remotely in 2023 due to COVID-19.
- **Chapter 5:** has been published as a journal article, "Impact on quality and diversity from integrating a reconstruction loss into neural audio synthesis" in the Journal of Acoustic Society of America 2023, December. In this work I conducted all the experiments for the three variants of the model and evaluated the performance via objective metrics. I finished the writing of the manuscript.
- **Chapter 6:** has been published as a conference paper "DDSP-SFX: Acoustically-guided sound effects generation with differentiable digital signal processing" at Digital Audio Effects (DAFx) conference 2024, September. This work was doen during a research internship at Dolby Laboratories in North Sydney from 6 March 2023 to 6 July 2023. I did all the experiments, subjective studies, analysis and writing the draft. The paper was further refined by the other two authors.
- **Chapter 7:** has been published as a conference paper "ICGAN: An implicit conditioning method for interpretable feature control of neural audio synthesis" at Digital Audio Effects (DAFx) conference 2024, September. In this work I presented a new conditioning method and finished the experiments. The evaluation method was introduced by the last author. I then completed the evaluation and finished the manuscript.
- **Chapter 8:** has been submitted as a journal article "Simi-SFX: A similarity-based conditioning method for controllable sound effect synthesis" at Journal of Audio

Engineering Society 2024, December. In this work I introduced a novel conditioning method and completed all the experiments, analysis and evaluations and writing of the manuscript.

Statement of Originality and Use of Generative AI

This is to certify that the content of this thesis is my own work. This thesis has not been submitted for any other degree or purpose. During the preparation of this thesis, Grammarly was used for the purpose of grammar check. Grammarly was also used for text enhancement, especially for sentence paraphrasing in Chapter 2 and Chapter 3. Where any text was modified by generative AI, the author then reviewed the resulting content for any errors, inaccuracies or biases, and modified it as required. The author takes full responsibility for the submitted thesis and ensures the work is their own and has used generative AI within the parameters of use, see the University of Sydney generative AI guide for researchers [1]. No generative AI was used for the production of contents such as text, image, or figures for this thesis.

I certify that the intellectual content of this thesis is the product of my own work and that all assistance received in preparing this thesis and all sources have been acknowledged.

[1] https://sydneyuni.service-now.com/sm?id=kb_article_view&sysparm_article=KB0031813

Contents

List of Figures

Nomenclature

ADAM Adaptive Moment Estimation

ADSR Attack Decay Sustain Release

AE Autoencoder

AI Artificial Intelligence

ANN Artificial Neural Network

BFA Bayes Factor Analysis

CNN Convolutional Neural Network

DAW Digital Audio Workstation

DCGAN Deep Convolutional GAN

DCT Discrete Cosine Transform

DCT Discrete Fourier Transform

DDSP Differentiable Digital Signal Processing

DL Deep Learning

DSP Digital Signal Processing

FAD Fréchet Audio Distance

FFT Fast Fourier Transform

FIR Finite Impulse Response

Fs Sampling Rate

GAN Generative Adversarial Network

GRU Gated Recurrent Unit

IFFT Inverse Fast Fourier Transform

IIR Infinite Impulse Response

ISTFT Inverse Short-Time Fourier Transform

KD Knowledge Distillation

LLM Large Language Model

LReLU Leakly Rectified Linear Activation

LSTM Long-Short Term Memory

MD Mahalanobis Distance

ML Machine Learning

MOS Mean Opinion Score

MSD Maximum Separation Distance

MUSHRA Multiple Stimuli with Hidden Reference and Anchor

NAS Neural Audio Synthesis

NDB Number of Statistically Different Bins

NMD Normalized Mahalanobis Distance

PANNs Large-Scale Pretrained Audio Neural Networks

PGHI Phase Gradient Heap Integration

REC Range of Effective Control

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

SFX Sound Effects

Simi-SFX Similarity-based Sound Effects Generative Model

STFT Short-Time Fourier Transform

STGAN Spectro-Temporal GAN

STGAN-AL STGAN with only Adversarial Loss

STGAN-GEN-RL STGAN Generator with Reconstruction Loss

TFR Time-Frequency Representations

TTA Text-to-Audio

V2A Visual-to-Audio

VAE Variational Autoencoder

VGGish Rectified Linear Unit

WGAN Wasserstein GAN with Gradient Penalty

WGAN-GP Wasserstein GAN

CHAPTER 1

Introduction

Sound effects refer to natural or synthetic sounds different from speech or music [1]. Sound effects are essential elements of auditory storytelling that enhance the realism and emotional depth of audio-visual experiences. Although they play an important part in digital media, such as film and games, general sound effects have received much less attention in research and development compared to speech or music. Even nowadays, recording sound objects on location or syncing sounds to a video via Foley recording has still been the mainstream approach for sound effects creation. As sound effects vary greatly among different categories, careful annotation of each sound recording is required, which makes it much more difficult to obtain a well-constructed and descriptive audio dataset compared to speech or music.

Additionally, unlike speech or music, which has high-level and interpretable labels such as texts and scores, a lot of the sound effects are difficult to describe with human-understandable languages or labels (e.g. the variations of a wind "whoosh" sound may in nature be hard to be described with words). Furthermore, many sound effects contain multiple acoustic events; it requires meticulous analysis and segmentation of the sound events to be able to model the physical activities (e.g. a recording of rain might be treated as an audio texture [2] whereas a single raindrop might be considered as an impact sound). Although many synthesis methods have been introduced (which is covered in Chapter 2 and Chapter 3), the modelling of sound effects remains a challenging yet rewarding task. In this thesis, research is conducted to analyze and categorize different sound effects, review the pre-existing synthesis methods, and explore novel and new techniques for generative sound modelling with a particular emphasis on controllability.

1.1 Motivation

In the field of speech synthesis, generative models are commonly used as neural vocoders[3] for converting texts into spectral representation and then into speech waveforms. The neural vocoders that utilize autoregressive or non-autoregressive neural networks serve to upsample the mel-spectrograms that contain linguistic information and acoustic information in a lower dimensional space back into a sequential audio waveform in a higher dimension. In other words, such neural networks learn a mapping from the temporal-spectral domain to the time domain by inputting precise mel-spectrograms. Additional conditioning methods have also been used to control the generation prosody [4, 5], phoneme [6, 7], syllable [8, 9] and speaker styles [10, 11], etc.

In the domain of music generation, deep generative models have been used to synthesize novel musical timbres with favourable guided user controls that are difficult or impossible to achieve using acoustic instruments [12, 13]. Under this task, deep generative models have been used extensively in modelling musical audio signals by learning the underlying structures of instrument sounds and summarizing them into what's known as latent spaces. Once the latent space is learned and structured, it could be used to offer parametric controls over the generated timbre by interpolating within the space [14].

With the demonstrated success of generative models in speech synthesis and music generation, an interesting but worthwhile question arises: **How can existing neural audio synthesis methods be applied to the creation of sound effects?** The methodology for creating sound effects varies depending on the context of the media. For instance, in linear media such as films and animations, the focus is primarily on synchronizing sound effects with visual actions, ensuring the appropriateness of sounds for specific scenes, and leveraging sound expressiveness to evoke emotions or capture audience attention. In contrast, interactive media like video games prioritize the timeliness of sound triggers, the spatial accuracy of sound origin, and the overall sound quality to enhance user interaction and realism. To this end, procedural audio [15, 16] is usually applied for games that require real-time interaction between players' actions and the delivered sounds. Across different applications,

the overarching goal remains the same: **To provide a wide variety of sound expressions to achieve sophisticated auditory landscapes.** Therefore, the development of tools that allow precise control over sound variations becomes crucial in the production and application of sound effects.

1.2 Research Questions

This thesis mainly focuses on the data-driven approach for sound effects modelling. Particularly, this thesis studies how to achieve controllable sound effects generation under the setting of a limited audio dataset without requiring huge computation resources. Given the current challenges in sound effects modelling and control that are introduced in 3.8, this thesis focuses on two steps to address the existing issues.

- How to generate wider range of sounds?
- If it's possible to generate a wide range of sounds, how to provide meaningful controls for neural audio synthesis?

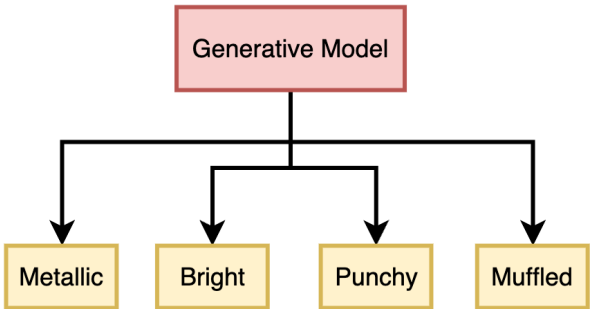


FIGURE 1.1. How to generate a wider range of sound effects with generative models?

To answer these questions, in Chapter 4, 5, I studied the trade-off between generation sample quality and diversity, and propose methods to improve not only the audio quality of frequently-used generative models such as GAN but also to preserve the generative diversity.

For the challenge of audio control, I focused on temporal control of the SFX in Chapter 5 and Chapter 6 by studying SFX modelling conditioned on temporal features. Then in Chapter 7 and

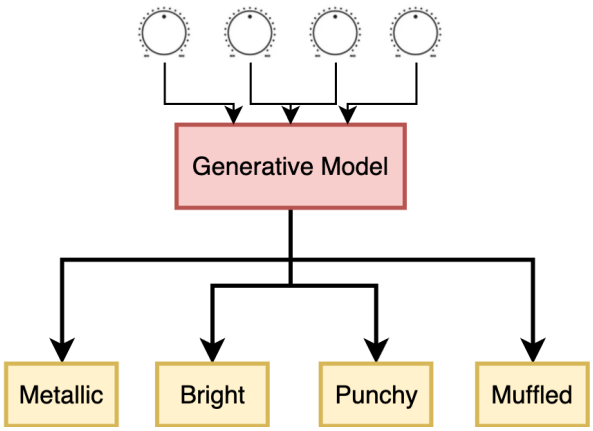


FIGURE 1.2. How to provide more meaningful controls for neural audio synthesis?

Chapter 8, I explored timbre control of SFX given small audio datasets with only categorical labels.

Given the current research gap in evaluation techniques that measure the effectiveness of timbre control, in Chapter 6, I have conducted listening tests to study the perceptual differences of various timbres generated from our model. In Chapter 7 and Chapter 8, I proposed several objective metrics to evaluate the effectiveness of timbre encoding from the learned latent spaces.

1.3 Outline of thesis

As shown in Figure 1.3, this thesis is separated into 9 chapters and is structured as follows:

- **Chapter 1:** Introduction of the thesis. This chapter introduces the definition, importance, and background of sound effects. It briefly summarizes the current stage and trends for sound effects modelling and points out the limitations and challenges of existing methods for sound effects creation. Then it introduces the three research objectives and scopes.
- **Chapter 2:** Background of sound effect synthesis. This section reviews the different types of sound effects, how they are created and used in different circumstances,

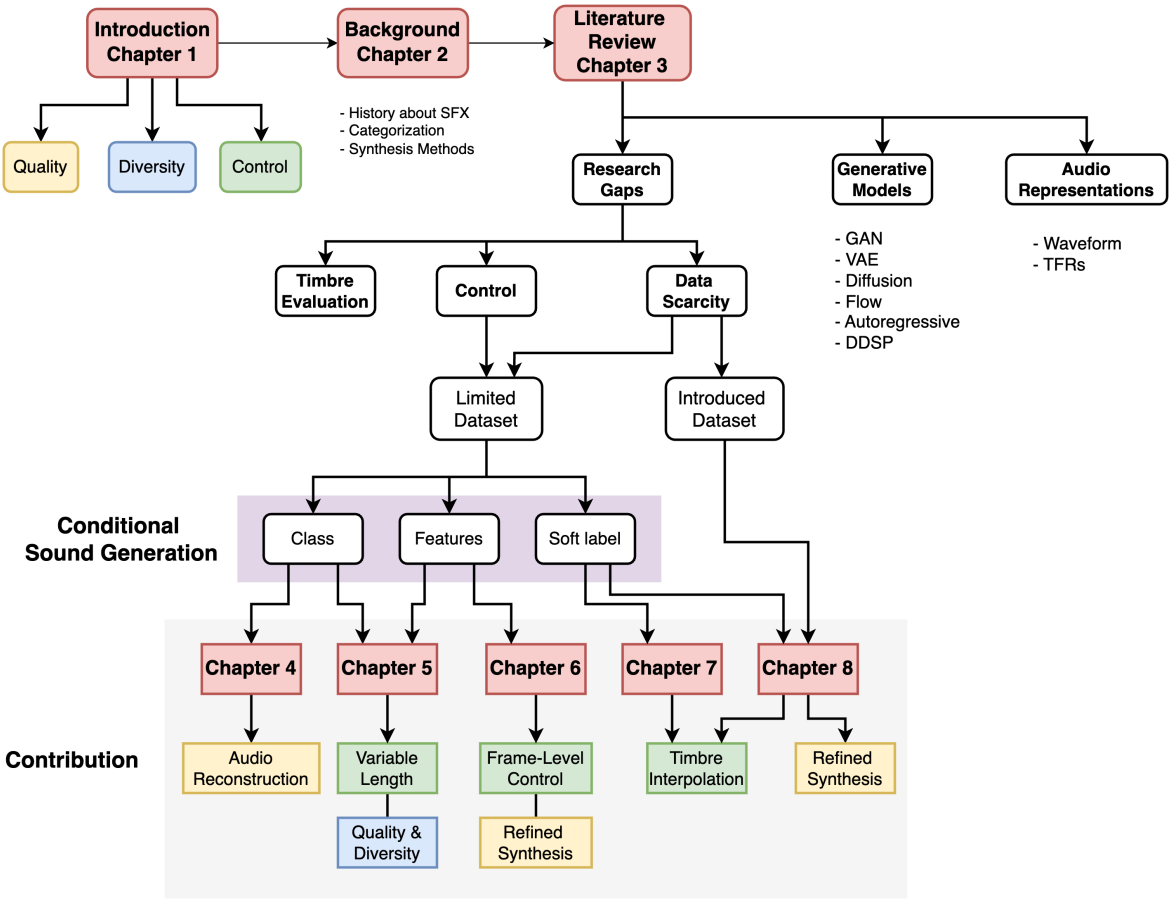


FIGURE 1.3. An overview of the thesis structure.

the various sound effect synthesis methods, and a knowledge review of machine learning.

- **Chapter 3:** Literature review of neural audio synthesis for sound effects generation. It also reviews audio representations, training techniques, and evaluation metrics.
- **Chapter 4, 5** studies the first research question: *How to generate wider range of sounds?* Specifically, in **Chapter 4**, I proposed a regularization technique to improve the generated audio quality by adding a regularizing loss into the generator to penalize the reconstruction errors caused by phase estimation algorithms [17]. In **Chapter 5**, I studied the impact on quality and diversity from integrating a reconstruction loss into neural audio synthesis. Additionally, I also explored arbitrary-length audio generation considering the problem of upsampling artifacts from transposed convolutional neural networks.

- In **Chapter 6, 7, 8**, I aim to answer the second research question: *How to provide meaningful controls for neural audio synthesis?* Specifically, in **Chapter 6**, I studied a hybrid modelling approach by leveraging differentiable digital signal processing (DDSP) [13] for sound effects synthesis. It proposes an architecture that achieves frame-level timbre control by directly manipulating the compressed latent space. In **Chapter 7**, I explored a novel conditioning method for neural audio synthesis that achieves timbre interpolation between discrete audio classes by relying on an implicit sampling process similar to variational inferencing [18]. In **Chapter 8**, I have further refined the conditioning method by proposing a similarity-based conditioning approach. Based on the findings and method introduced in Chapter 8, this chapter further extends this approach by leveraging large pre-trained audio representation models. It proposes a similarity-based conditioning method that enables users to interact with a fine-grained control parameter to interpolate audio timbres across different types of sounds.
- **Chapter 9:** Conclusion of the thesis. This chapter reviews how the work done during the candidature fits into the literature and the current status of neural sound synthesis. It also discusses directions for future research.

1.4 Statement of ethics

This thesis involves experiments, including listening studies, that require the participation of human subjects. All participants were fully informed about the nature and purpose of the studies prior to participation, and written consent was obtained in compliance with ethical research standards. The studies were conducted under the University of Sydney Human Ethics protocol, approved by the relevant ethics committee under Project Identifier: 2023/HE000562. All procedures adhered to the University of Sydney's Human Research Ethics Guidelines, ensuring respect for participant rights, privacy, and welfare. Details of the consent forms and demographic questions used during the listening studies are provided in the appendices.

Additionally, this thesis acknowledges the broader ethical considerations related to the field of generative sound synthesis. While the algorithms and methods explored aim to advance automation in creative processes, the intent is to assist and augment human creativity, not to replace individuals working in creative roles. Care has been taken to approach this research with sensitivity to its potential societal implications, aiming to contribute positively to the fields of sound design and generative audio technology.

CHAPTER 2

Background

2.1 History of Sound Effects

Sound effects [15], distinct from music and dialogue, are crafted sounds deployed in settings such as movies, television, video games, or virtual reality [19]. The primary goal of creating sound effects is to provide auditory cues for actions or occurrences within a story's universe to immerse the audience. Sound effects were first introduced in the theatre, where artists used simple props to simulate sounds like wind, rain, and thunder. Thunder, for example, could be simulated by shaking a large sheet of thin metal, and wind effects were often created by turning a crank on a device containing silk or canvas that brushed against wooden slats. Common tools included shoes for footsteps, doors and hinges for creaks and slams, and cornstarch in leather pouches to mimic the sound of walking on snow.

In the digital era, sound effects are often recorded through Foley techniques or field recording. Foley [20], named after sound effects pioneer Jack Foley in the 1920s, refers to the art of performing sound effects live to enhance the audio in film, television, and other media. Foley recording involves using everyday objects to create realistic auditory illusions that match the actions on screen. For instance, the sounds of gunshots and explosions in films are often a blend of several loud, distinct noises to make them more thrilling. Over the decades, Foley has evolved into a professional field, with artists skilled in synchronizing precise sounds to live or recorded visuals.

Field recording, on the other hand, is conducted in natural or urban environments with portable recording devices, aiming to capture everything from wildlife sounds and natural phenomena

to bustling cityscapes and cultural events. In film and video games [21], many scenes require an on-site recording of the atmosphere or sonic activities to provide immersive auditory experiences.

Foley and field recordings are important traditional techniques for creating sound effects. Sound libraries, which store SFX recordings, are crucial for the development of sound design, where sound designers typically use them as a starting point and layer them with other sounds or with audio effects.

However, from the sound design perspective, even though recorded audio libraries provide a large number of audio samples for recreation purposes, in real-world applications such as film and games, it is often unrealistic to find an ideal and suitable audio piece that exactly fits the context and scene of the video or game. Sound designers would usually need to carefully record sounds and manipulate the audio effects to reshape the sounds to better adapt to the scene.

Additionally, recording sounds over and over again for different clips of videos can be extremely time-consuming and labour-intensive. Creating Foley sounds that are perfectly synchronized with the visual action requires not only a large number of equipment and materials but also an undisturbed studio environment. This setup can be costly to maintain, especially for smaller production companies or independent filmmakers.

Another limitation is the inherent challenge of reproducing certain sounds. While Foley is excellent for enhancing physical actions like footsteps, cloth movement, or door creaks, it can struggle with more complex sounds like intricate machinery or unique natural phenomena, which might still require original field recordings or sophisticated sound design techniques.

Field recording typically requires a team of skilled individuals to handle specialized equipment, manage logistics, and ensure the quality of recordings. Additionally, achieving the necessary silence in the background to ensure clean recordings can be a huge challenge, particularly in naturally noisy or bustling environments. These factors make Foley and field recordings a resource-intensive endeavour that demands significant time, effort, and financial investment.

Therefore, having a synthesis algorithm capable of generating sound effects that are appropriate for the scenes in a controllable way has been an indispensable task. In the early ages of films and games, DSP-based algorithms were mainly used for synthesizing sound effects. Yet, recently, neural audio synthesis approaches have brought a lot of attention to the field of game audio and media industry, owing to its unprecedented power of high-quality sound generation. In the following sections, I summarize DSP-based synthesis methods for sound effects and briefly describe the generative modelling approach for sound synthesis.

2.2 Sound Synthesis Methods

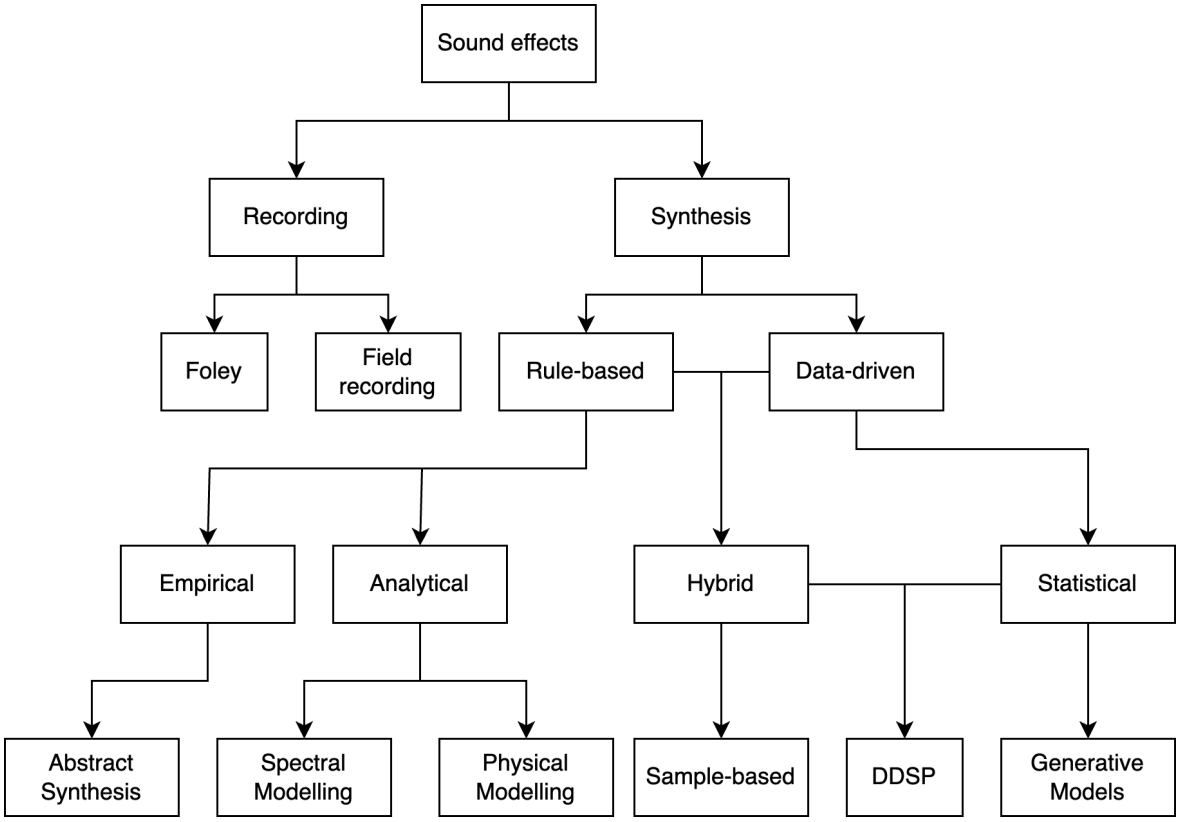


FIGURE 2.1. An overview of different synthesis methods.

Sound effects synthesis refers to the process of artificially creating or manipulating audio signals to produce desired sound effects. As shown in Figure 2.1, I broadly categorize sound synthesis methods as **Rule-based** and **Data-driven** methods. Much like deductive vs inductive reasoning, rule-based or knowledge-based methods aim to explicitly define laws or general principles that describe a sound generation process, while data-driven approaches rely on large amounts of audio data to implicitly infer the hidden statistics to generate new waveforms via sampling. In the modern world with digital technologies, digital signal processing (DSP) has become the cornerstone for the development of audio technologies, including sound synthesis. Therefore, most rule-based methods take use of DSP techniques to model the sound generation behavior. More specifically, it could be categorized into **Empirical Approach** and **Analytical Approach**.

The empirical approach is commonly used to synthesize simple and abstract sounds directly with oscillators [22]. The analytical approach, which includes spectral modelling and physical modelling, usually involves analyzing the contents of a sound waveform or studying the physical behaviour of a system before synthesis.

A data-driven approach, on the other hand, instead of studying the underlying law of the sound system, models waveforms by directly studying the digitized audio samples. Generative models, for example, fall into the category of a data-driven approach, which focuses only on the statistics of the audio dataset and is tasked to learn the distribution of the target audio.

Additionally, apart from rule-based and data-driven approaches, hybrid methods extend the data-driven approach by modifying existing audio samples via explicit DSP Algorithms. In recent years, with the advancement of machine learning, a new combination between DSP methods and generative models has come into place, which integrates differentiable digital signal processing (DDSP) [13] operators into a neural network architecture.

In the following sections, I first review the backgrounds of DSP and DSP techniques, followed by a detailed discussion of each synthesis method as well as its advantages and suitable sounds for modelling.

2.2.1 Background about DSP

Sound, in its nature, exists as an analog waveform-a continuous signal representing variations in air pressure over time. In the digital era, these continuous waveforms are sampled in discrete intervals, which are a series of data points that approximate the original signal. This process, governed by the Nyquist-Shannon sampling theorem [23], ensures that the waveform can be faithfully reconstructed if the sampling rate is at least twice the highest frequency of the signal. Each of these discrete data points, or samples, is quantified and stored as digital data, enabling a range of mathematical operations to be applied. This transition from continuous to discrete signals forms the foundation of DSP, opening up possibilities for precise control and manipulation of audio signals.

In the domain of DSP, techniques for audio signal processing [24] are broadly covered as filtering, equalization, compression, reverberation, and more. Such techniques expose precise control parameters over the timbre, dynamics, and spatial characteristics of sound. These techniques are equally pivotal in sound synthesis, where digital algorithms generate or shape sounds in real-time. For sound synthesis, common DSP techniques commonly include:

- **Waveform Synthesis**: By directly generating audio waveforms using oscillators such as sine, square, or sawtooth waves, DSP enables the creation of basic building blocks for synthetic sounds. For instance, in retro video games [21], simple square waves were often used to produce iconic chiptune sound effects like laser blasts or beeps. In modern sound design, waveform synthesis is combined with additive synthesis to create complex textures, such as simulating the hum of machinery or the shimmering sound of futuristic engines.
- **Filtering and Modulation**: Filters and modulation techniques are the cornerstones of DSP [25]. Digital filters—such as Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters—allow for precise control over the spectral content of a sound by attenuating or emphasizing specific frequency bands [26]. This capability is essential in crafting iconic effects like the muffled tone of a voice on a telephone (band-pass filtering), or the eerie, low-pass filtered rumble of a distant explosion. Beyond static filtering, modulation techniques such as amplitude modulation (AM) [27] and frequency modulation (FM) [28] introduce dynamic movement into sound by varying parameters over time. These methods are commonly used to create rich, evolving textures—from the bell-like tones popularized in science fiction soundscapes to the siren-like pitch fluctuations of emergency vehicle alarms.
- **Time-Domain Processing**: Algorithms such as granular synthesis [29] or time-stretching [30] manipulate the temporal structure of sounds, enabling the creation of effects like echoes, reverb, or pitch shifting. Granular synthesis, for example, can break down a recorded explosion into small sound grains and rearrange them to create a slow-motion explosion sound for cinematic sequences [31]. Time-stretching is often used in horror films to elongate screeches or growls, adding a sense of dread and tension.

- **Dynamics processing**: Dynamics processing [32] refers to a class of DSP techniques that manipulate the amplitude envelope of audio signals, shaping how sounds evolve over time in terms of loudness and impact. This includes tools such as compression, limiting, expansion, and gating. In sound effect synthesis, these tools are crucial for emphasizing or controlling transient events—such as enhancing the punch of an explosion, tightening the attack of a gunshot, or muting background noise between footsteps. Compression reduces the dynamic range by attenuating loud peaks and amplifying quieter parts, often used to create more intense and controlled effects.

2.2.2 Empirical Approach

An empirical approach towards sound synthesis relies heavily on the experience or perception of the sound designer, who uses multiple signal processing techniques to create and shape the timbre of the sound. The most straightforward technique would be trial-and-error, which entails building up the complexity of a sound via digital oscillators and filters while listening to the output until it becomes the favoured sound [22]. However, this approach depends highly on the experience of the sound designer and can be extremely time-consuming in most circumstances.

Many empirical synthesis algorithms entail modulation techniques, which is manipulating one or more parameters of a carrier signal using a modulator signal to generate complex and dynamic sounds. Common modulation techniques include Amplitude Modulation (AM) and Frequency Modulation (FM).

Amplitude Modulation [27] involves varying the amplitude of a carrier signal according to a modulator signal. The resulting waveform contains the original carrier frequency along with sidebands at the sum and difference of the carrier and modulator frequencies.

$$s(t) = [A_c + A_m \cdot \sin(2\pi f_m t)] \cdot \sin(2\pi f_c t)$$

where:

- A_c: Amplitude of the carrier signal
- A_m: Amplitude of the modulator signal
- f_c: Frequency of the carrier signal
- f_m: Frequency of the modulator signal

AM is commonly used to create tremolo effects in sound synthesis, where the perceived loudness of a sound fluctuates over time. It is also used to generate metallic and bell-like tones by choosing appropriate carrier and modulator frequencies. Compared to other synthesis methods, AM synthesis is simpler to implement but unable to produce a wide range of timbres and variations. Therefore its usage has been limited to abstract sounds required for specific Sci-fi or futuristic scenes in films or games [21].

Frequency Modulation (FM) [28] involves varying the instantaneous frequency of a carrier signal based on a modulator signal. The technique produces a rich spectrum of harmonics and sidebands, determined by the modulation index I and the ratio between the carrier and modulator frequencies.

$$s(t) = A_c \cdot \sin\left(2\pi f_c t + I \cdot \sin(2\pi f_m t)\right)$$

where I is the modulation index, determining the intensity of frequency deviation. FM synthesis is widely used for creating complex and dynamic timbres. Owing to this advantage, many retro games [33] employ FM synthesis for sound design. However, as the relationship between the output sound timbre the instantaneous frequency behaves uninterpretable, it if often difficult to shift sound towards the favored timbre, and it relies heavily on listening to the synthesized output frequently.

Another empirical approach is the digital waveshaping [34] technique, which is a sound synthesis technique that transforms a simple input waveform, such as a sine wave, into a more complex waveform by applying a nonlinear mathematical function [35]. The essence of waveshaping lies in using a transfer function $f(x)$ to map the amplitude of the input signal $x(t)$ to a new amplitude $y(t)$, resulting in unique and expressive sound characteristics. However, similar to FM synthesis, waveshaping is also a nonlinear transformation that may

introduce high-frequency components, leading to aliasing in digital systems if not handled carefully. Therefore, it can be challenging to design and tune the transfer function to achieve a specific timbre.

2.2.3 Analytical Approach

2.2.3.1 Spectral Modelling

Spectral modelling is a two-stage method, which includes the analysis of the spectral and dynamic contents of the audio waveform, and the synthesis based on the analyzed features. In the analysis stage, interpretable audio features (pitch, spectral centroid, loudness, etc) are usually extracted from the input sound and used as controllable parameters. During the synthesis stage, waveform is created by inputting such parameters into standalone synthesizers, which are built specifically for certain types of sounds. One of the best-known approaches is the Spectral Modelling Synthesis (SMS) [36], which entails the separation of audio signals into harmonic contents to be synthesized by sinusoidal models and noisy components to be synthesized by subtractive noise synthesis [37]. The process works as follows:

1. Analysis: The audio signal $x(t)$ is analyzed in the frequency domain using a transform such as the *Short-Time Fourier Transform (STFT)*. The STFT breaks the signal into overlapping time frames, producing a spectrogram:

$$X(t, f) = \int_{-\infty}^{\infty} x(t) \cdot w(t - \tau) e^{-j2\pi f \tau} \, d\tau \tag{2.1}$$

where:

- $X(t, f)$: Time-frequency representation (magnitude and phase).
- $w(t)$: Window function applied to each frame.
- f: Frequency.

2. Decomposition: The spectrogram is decomposed into two main components:

- **Sinusoidal Components:** Represented by partials (harmonic or inharmonic):

$$x_{\text{sin}}(t) = \sum_{k=1}^{K} A_k(t) \cos(2\pi f_k t + \phi_k(t)) \tag{2.2}$$

  where $A_k(t)$, $f_k(t)$, and $\phi_k(t)$ are the amplitude, frequency, and phase of the k-th component.

- **Residual Noise:** The part of the signal that cannot be explained by sinusoidal components, typically modeled as stochastic noise:

$$x_{\text{res}}(t) = x(t) - x_{\text{sin}}(t) \tag{2.3}$$

3. Synthesis: The sound is resynthesized by reconstructing the sinusoidal and residual components, either exactly or after modifying their parameters. For instance, the amplitudes or frequencies can be adjusted for pitch shifting or timbre modification.
Spectral modelling allows precise control over timbre, pitch, and amplitude by separating deterministic (sinusoidal) and stochastic (noise) components. It has been shown to be useful in creating realistic sounds, such as musical instrument [38, 39, 40], singing voice [37] or environmental noises [41]. However, for a lot of rigid-body sound effects such as footsteps or gunshots, this method falls short of synthesizing the transient peaks and is, therefore, limited in its application. To this end, this method has been extended [42] by splitting audio into three components, namely, harmonic, transient, and residual parts. The sound is then recreated using harmonic and noise synthesis, plus a transient synthesizer. For many real-world sound effects, this approach has been shown to be really, given its simplicity and interpretability. However, one critical drawback of this approach is that the synthesis performance depends heavily on the algorithm of the decomposition of the original signal.

2.2.3.2 Physical Modelling

Physical modelling refers to the synthesis technique that simulates the physical mechanisms and properties of real-world sound sources. Different from spectral modelling, which analyzes the recorded audio contents directly, physical models aim to build mathematical equations and algorithms to replicate the acoustic behaviour of vibrating objects, resonant bodies,

and interaction forces to generate sounds. Physical modelling is particularly effective for synthesizing sound effects that originate from well-understood physical systems, such as rigid-body collisions like footsteps [43, 44, 45], impacts [46, 43]. However, since sound systems vary in categories, it is typically required to construct a specific physical model for each of the various acoustic behaviors. Some common physical models for sound effects synthesis include modal synthesis [47], latent force models (LFMs) [48], and digital waveguides [49].

Modal Synthesis

Modal synthesis [47] is a physical modelling technique that uses the eigenmodes of a physical system to synthesize sound, particularly effective for simulating the vibrations of solid objects such as musical sounds [50, 51] or percussive sounds [47, 52, 53]. Modal synthesis represents a system's response as a sum of its modal responses, where each mode is modelled as a damped harmonic oscillator:

$$s(t) = \sum_{n=1}^{N} A_n e^{-d_n t} \cos(2\pi f_n t + \phi_n)$$

where $s(t)$ is the sound signal, A_n, d_n, f_n, and ϕ_n are the amplitude, damping factor, frequency, and phase of the n-th mode, respectively. Modal synthesis is well suited for sounds with a clear resonant frequency, such as musical instruments or object impacts like weapon sounds [54], allowing detailed and high-precision control over sound characteristics.

2.2.3.3 Mass Spring System

The mass-spring system is a core model in physical modelling for sound synthesis, offering a versatile method to simulate the dynamics of physical systems. It employs interconnected masses and springs to mimic mechanical vibrations, where the masses represent points of inertia, and the springs model the restoring forces. Dampers are included to simulate the dissipation of energy, reflecting real-world damping effects [55]. The behaviour of this system is governed by the principles of mechanics, which ensure that the movement of each mass and spring corresponds to forces applied, whether internal (from the spring tension and damping) or external. This framework is particularly effective for generating realistic audio effects for

virtual environments and interactive applications, such as simulating collision sounds [56] or other interactions of objects. It provides a computationally efficient solution for real-time sound generation. However, its simplicity may sometimes limit the ability to fully capture the non-linear behaviours of more complex systems, thereby affecting the realism of the synthesized sounds.

Digital Waveguides

Digital waveguide synthesis [49] is a powerful technique for simulating the vibration and propagation of waves through various physical media, and it has become particularly influential in the field of sound synthesis, especially for musical instruments [57]. This method models wave propagation as it occurs in real-world physical systems, using discrete time delay lines that represent the wave medium. Digital waveguides are primarily used to model systems where wave-like behaviour is prominent, such as in the strings of a guitar or the air column of a wind instrument [58]. The technique involves splitting the wave into two components—travelling in opposite directions along the waveguide—and utilizing algorithms to simulate their reflections and interactions within the medium. This approach allows for highly realistic and dynamic sound production, making it ideal for virtual instruments and effects that require nuanced audio representations. In the context of sound effects, digital waveguides can be adapted to simulate sounds like swinging swords [59], moving vehicles [60], or any object that interacts with the air in a way that produces acoustic waves. The ability to precisely control wave reflections and interactions makes digital waveguides particularly suitable for creating complex, evolving soundscapes that react to user inputs.

2.2.4 Hybrid Approach

Apart from the above-mentioned methods, which synthesize sounds from scratch, the hybrid approach utilizes audio samples as a starting point and then applies signal processing techniques to shape or control the sound. One common method is sampled-based synthesis. Sample-based synthesis refers to a set of techniques that use recorded audio samples as the basis for generating sound. This category includes granular synthesis, concatenative synthesis,

and wavetable synthesis, each employing distinct approaches to manipulate and utilize these samples.

Granular Synthesis. Granular synthesis [29] is a method of sound synthesis that operates on the microsound time scale, where sounds are constructed from basic units called grains. These grains are small (typically between 1 to 50 milliseconds long) and are extracted from a longer sample. This technique manipulates these sound grains by changing their duration, pitch, and spatial position, or by layering them to form new sounds. The power of granular synthesis lies in its ability to create highly complex sounds that can evolve in unpredictable and rich ways, which makes it highly useful in sound design for films and games [61]. It allows designers to explore textures and timbres that are difficult to achieve with other synthesis techniques. Applications include creating ambient soundscapes, unique sound effects, and morphing sounds in ways that blur the lines between recognizable and abstract sound forms [62].

Wavetable Synthesis. Wavetable synthesis [63] is a technique that uses pre-recorded waveform data organized into tables, hence the name. Synthesizers that employ this method read these tables sequentially or non-sequentially to produce sound. Unlike granular synthesis, which slices sound into tiny grains, wavetable synthesis allows for the creation of waveforms that can be smoothly interpolated between different shapes, enabling dynamic timbral control. The ability to morph between different waveforms seamlessly without generating audible artifacts is one of the major advantages of wavetable synthesis. Although less common than other synthesis methods, attempts have been made to adapt wavetable synthesis for the creation of complex sound effects for multimedia [64].

Concatenative Synthesis: Concatenative synthesis [65] is a technique that synthesizes sounds by concatenating short segments of recorded sound according to a set of sound descriptors. This method is driven by a target sound's acoustic properties that the synthesis system tries to replicate by selecting and assembling segments from a large database of source sounds. The segments are chosen based on their spectral, temporal, and amplitude characteristics to match the target sound's properties as closely as possible. Concatenative

synthesis has been explored in the interactive and real-time context in [66], where users are given real-time control over the sound attributes with gestural inputs.

2.2.5 Summary about DSP-based synthesis methods

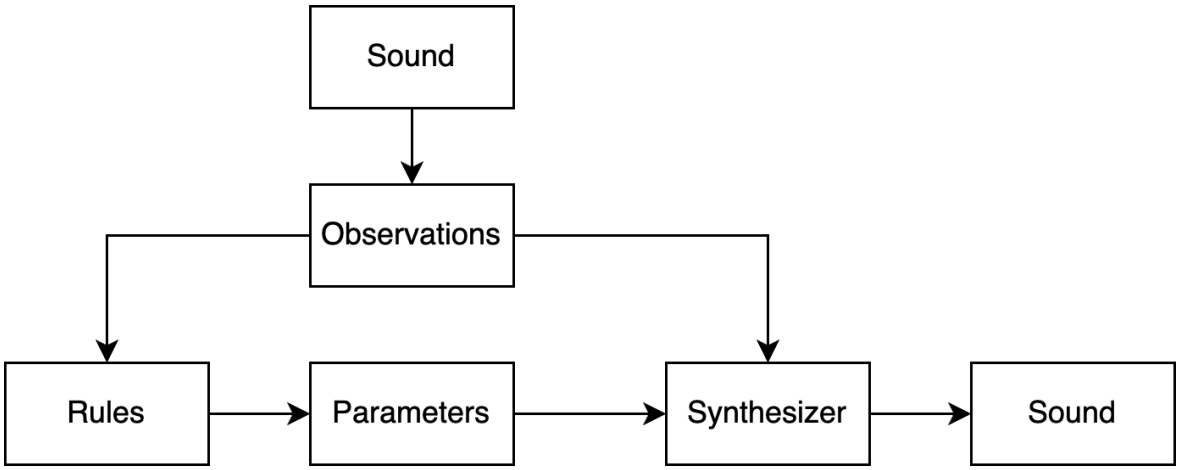


FIGURE 2.2. Rule-based sound synthesis approach.

As detailed previously, DSP-based sound synthesis, categorized under rule-based methods, utilizes predetermined rules and mathematical models to fabricate sounds. The primary advantage of rule-based DSP lies in its predictability and low latency. Such attributes are indispensable for real-time audio applications such as musical instruments [67] and interactive game audio [16]. However, this methodology often demands considerable time and a deep understanding of the sound system, making it generally less efficient compared to data-driven approaches. Moreover, the complexity of many real-world sounds poses a challenge to the synthesis quality and fidelity achievable with DSP-enabled algorithms, which often do not match the performance of data-driven methods. Consequently, the subsequent section delves into machine learning models for sound generation, with a particular focus on neural audio synthesis approaches, offering a more sophisticated framework for understanding and generating complex audio waveforms.

2.3 Chapter Summary

In this chapter, I reviewed DSP-based synthesis methods and the fundamentals of machine learning. Although DSP-based synthesis algorithms provide interpretable control parameters, to achieve a good synthesis quality, they usually require a thorough understanding of the physical activities of the sound generation process. Generative models, on the other hand, have many advantages in sound generation, such as the great performance in sound quality and fidelity [68] given abundant dataset, the capability of automatic generation given another modality [69], and the flexibility of sound generation without requiring extensive knowledge about the physics of the sound. In the following chapter, I will review state-of-the-art neural audio synthesis using generative models. For readers without background in machine learning, a brief summary about machine learning was included in Appendix 1.

CHAPTER 3

Literature Review

3.1 Generative Models

Generative models, a subset of machine learning, excel at generating new data samples that resemble the training data, capturing complex input distributions effectively. Unlike discriminative models that model the conditional probability $P(y|x)$, where y is the target and x is the input feature, generative models aim to learn the joint probability $P(x, y)$, encompassing both features and labels. Their primary goal is to simulate the dataset's distribution $p(x)$, allowing them to generate new instances x from this learned distribution.

A typical generative model can be expressed with the equation:

$$x = g(z; \theta) \tag{3.1}$$

where z is a latent variable drawn from a prior distribution $p(z)$, g is the generative function parameterized by θ, and x is the generated data. This setup enables the model to transform latent representations into data space, effectively modeling the conditional probability $p(x|z)$.

During training, these models often employ Maximum Likelihood Estimation (MLE) [70] to optimize parameters θ by maximizing the data's likelihood under the model. This usually involves minimizing the negative log-likelihood:

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{N} \log p(x^{(i)}|z^{(i)}; \theta) \tag{3.2}$$

Here, $x^{(i)}$ represents observed data instances, and $z^{(i)}$ the corresponding latent variables. By learning to map these latent codes to the data space, the model captures and reproduces the underlying distribution effectively, allowing it not only to mimic the training set but also to explore and generate novel data variations within the learned distribution.

Generative models initially focused on simple algorithms, such as mixture models [71] and hidden Markov models [72], which could model the distribution of data but were limited in their capacity to handle the high dimensionality and complexity of real-world data. As deep learning techniques developed, they were integrated into generative modeling, leading to the creation of deep generative models. Deep generative models leverage the power of deep neural networks to learn detailed and complex data distributions. They are characterized by their ability to generate high-fidelity and diverse outputs that closely mimic the input data.

There are various types of deep generative models, but a good classification of them can be found in [73], where generative models are categorized based on the strategy they use for modeling the data distribution: through explicit or implicit density estimation.

Explicit density models directly define and manipulate the probability distribution of the data. These models explicitly parameterize the distribution and are capable of calculating exact probabilities for given data instances. Explicit density models include Autoencoders (AE) [74], Flow-based Models [75, 76], and Autoregressive Models [77, 78]. These models are particularly advantageous for tasks where probability estimation, sampling, or inference about the data distribution are crucial.

Implicit density models, in contrast, focus on learning to generate data by sampling from a distribution, without necessarily learning or specifying the exact form of that distribution. These models are known for their ability to generate complex and high-quality samples. Examples of implicit density models are Generative Adversarial Networks (GAN) [79] and Diffusion Models [80, 81, 82]. Implicit models are highly effective for generating data where the primary goal is realism and diversity of the outputs rather than explicit probability estimation. They are especially prominent in fields where the detailed modeling of data distributions is less critical or where exact likelihoods are impractical to compute.

Generative models offer many advantages for sound synthesis. Paired with extensive data, they can usually achieve a high degree of synthesis quality and realism, which can be extremely challenging for traditional rule-based methods which require extensive analysis and study of the sounds beforehand. With proper instructions [83] and conditioning [84, 85], generative models can generate new audio instances that do not exist in the dataset. This enables sound designers to creatively explore and create new sound effects [86]. Additionally, generative models are able to learn intricate features and representations of audio signals, facilitating the discovery of underlying patterns and structures in sound data [87, 88, 89]. This capability allows them to synthesize complex audio sequences that would be challenging to produce using traditional synthesis techniques. Finally, generative models are usually source-agnostic, meaning that they can be adapted to different types of audio synthesis tasks without needing to be retrained from scratch. They can also be scaled to accommodate larger datasets or more complex sound environments [90, 91] as needed, making them versatile tools for a range of audio applications.

3.2 Generative Sound Effects Modelling

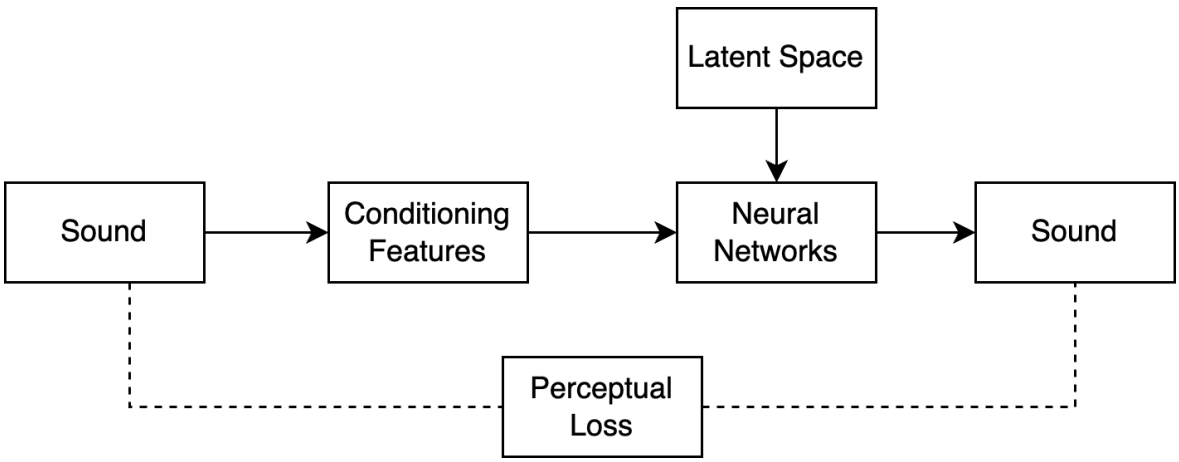


FIGURE 3.1. Statistics-based sound modelling approach.

Generative sound modelling, or often called neural audio synthesis (NAS) [92], is the process of generating digital audio waveforms using neural networks and generative models. Different from rule-based methods, NAS is data-driven, which utilizes a group of audio samples as

dataset, and generates new samples from learned statistics. Similar to many of the other sound generation tasks, generative modelling of sound effects does not concern the sound propagation and diffusion process itself but focuses only on the audio waveform that has been recorded. The reconstruction of wave fields [93] and propagation [94] is a different topic separate from this audio source generation. In the era of deep learning, NAS usually involve mapping a latent distribution to the true audio data distribution during training and sampling from the latent distribution to generate new audio data during inference. These models are now being explored for their potential to generate high-quality sound effects, offering new possibilities for creating realistic, dynamic, and novel audio content in applications such as film, gaming, and virtual reality [16].

Given the success of speech synthesis, especially with the maturity of neural vocoders [95, 96] in the speech synthesis domain, nowadays, many NAS research start to focus on the generation variation aspect. In the field of sound effects modelling, a critical task includes learning the patterns of similar audio waveforms in order to generate new ones that fall into the learned categories with some human-interpretable controls on certain sonic characteristics [97, 98]. For example, when generating footstep sounds, it is usually favored to synthesize them with slightly different acoustic features, as if they have been recorded with different types of shoes stepping on different surface materials [99, 100]. Given a set of sounds in the same category such as gunshots, generative models could capture and learn the nuanced waveforms every recorded gunshot sound effect produces and generate slightly new ones that sound similar to the inputs. With the help of conditioning techniques, such models could also learn the sound effects across various categories such as rifle shots and pistol shots [101].

Neural audio synthesis, despite its advanced capabilities in generating sound effects, is marked by several challenges, particularly in terms of controllability when compared to traditional DSP-based methods. DSP-based synthesis offers a high degree of controllability through numerous adjustable parameters [102, 103], allowing sound designers to explicitly shape the audio output according to precise specifications. This level of direct control is crucial in scenarios where specific sound characteristics [81] are needed, such as in video game development or film production. In contrast, neural audio synthesis generally involves training

models on large datasets to learn the distribution of sound data, which inherently limits the ability to manipulate specific sound attributes independently. While neural models excel at capturing complex audio textures and producing naturalistic sounds, they often lack the intuitive control that DSP methods offer. For instance, adjusting the timbre, pitch, or decay of a sound can be straightforward with DSP techniques through set parameters, but achieving the same level of detailed control in a neural model may require intricate manipulation of the model's inputs or retraining the model with augmented data to steer the outputs.

Moreover, neural audio synthesis can struggle with the generation of highly stylized or abstract sounds that do not closely align with the training data's distributions. This is particularly challenging in creative fields where unique and unconventional sounds are desirable. The synthesis of long, continuous sounds also poses problems in maintaining consistency and coherence over time, which DSP-based methods can handle more effectively due to their deterministic nature. These challenges are compounded by the resource-intensive nature of neural models, which require significant computational power and lengthy training periods. This demand for high-end hardware like GPUs or TPUs can be a barrier for smaller teams or projects with limited resources, further affecting the practicality of implementing neural audio synthesis in various settings.

In the subsequent sections, we will delve deeper into the definitions and functionalities of various generative models, highlighting their unique advantages and applications in the domain of sound synthesis. Each section will present detailed examples and case studies that illustrate how these models have been effectively utilized to empower audio creation. This exploration will not only clarify the operational mechanisms of each model type but will also discuss the latest advancements and emerging trends in the field, offering insights into the future directions of audio generative technology.

3.2.1 Autoregressive Models

Autoregressive models [104] are a type of generative models that generate data sequentially. It models sequential data with the help of chain rules and conditional probability. It assumes that

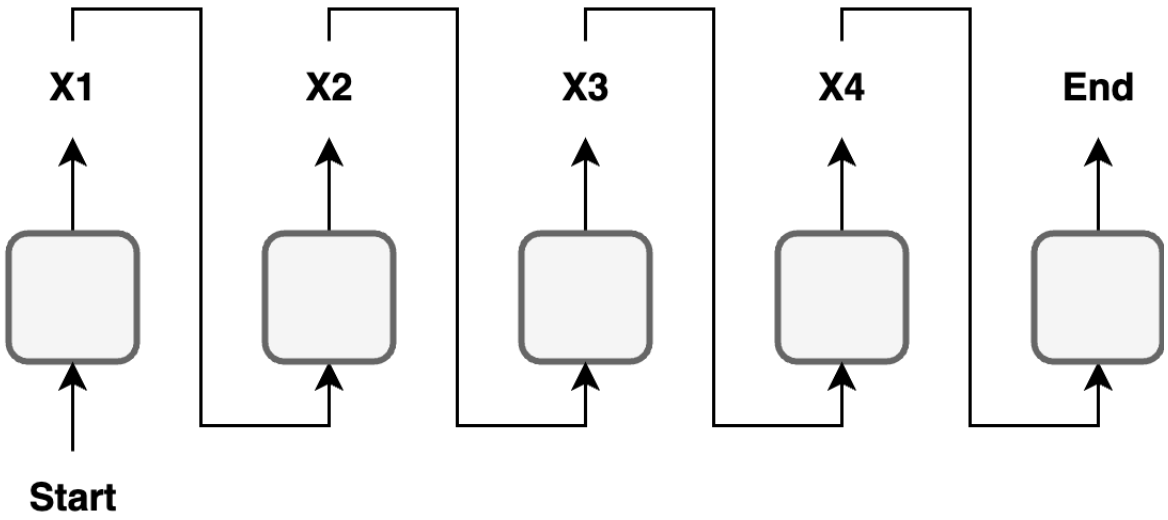


FIGURE 3.2. A simple visualization of auto-regressive models.

the generated output is only dependent on its previous outputs, and is modeled by multiplying the conditional probabilities of its previous samples. The likelihood of an i-dimensional data distribution $P(x)$ is modeled as the product of the conditional probabilities of the observed sequence:

$$P(x) = \prod_i P(x_i|x_1, x_2, ...x_{i-1}), \tag{3.3}$$

where x_i is the current sample and $x_1, ...x_{i-1}$ refer to the previous samples.

Because of this conditional probability on sequences flowing only one direction from left to right, it is very natural and appropriate to apply to time-series data and audio signals. The benefit of autoregressive generative models is that they are easier to train than other generative models, such as GAN, and capable of generating high-fidelity data, although they usually suffer from slow training time. In fact, it has shown great value in speech modelling with the introduction of Wavenet[77], a deep autoregressive model that is able to generate high-quality audio by modelling audio signals one sample at a time. To ease the computational needs and increase its capability for understanding the relationship between audio samples, Wavenet uses dilation convolution neural networks, which expands its receptive fields exponentially. Xubo et al[105] used PixelSNAIL[106], an improved autoregressive model that enables

self-attention, to train on the learned discrete time-frequency representations(DTFR) from VQ-VAE.[107] In the inference stage, it enables sampling from the variables trained in PixelSNAIL to generate new DTFRs.

3.2.1.1 Applications

As autoregressive models don't learn a feature representation as density estimation models do in an unsupervised way, they are usually used as a decoder paired with a separate encoder which allows for interpolating in the latent space. Nsynth[14] for example, uses an autoregressive decoder that is conditioned on the learned temporal encoder representation. By pairing with the temporal encoder, the autoregressive decoder could be used to generate novel and good-quality instrument timbres by interpolating in the latent space. We can also see autoregressive decoders being used in text-to-sound generation task. Liu et al.[105] uses a conditional vector-quantized VAE model that encodes spectrogram information of the sounds into sound latent codebooks which are then decoded discretely into reconstructed spectrograms with convolutional filters. Yang et al.[83] proposes two approaches for decoding the learned discrete representations from text encoders, one autoregressive approach and another non-autoregressive approach. They further found that a non-autoregressive approach with diffusion models[82] outperforms the autoregressive decoders, because it is difficult for autoregressive models to model long-term time dependent global structures due to the unidirectional modelling nature. In a more recent example, Felix et al.[108] uses an autoregressive transformer decoder to generate audio samples conditioned both on textual inputs and also discrete representations of the audio trained in the first stage.

3.2.2 Autoencoders (AE)

Autoencoders (AEs) [109] are neural networks that aim to learn compressed, efficient representations of the input data through an unsupervised learning process that involves data compression and decompression. The architecture of an AE consists of two main parts: the encoder and the decoder. The encoder compresses the input data into a latent-space representation, and the decoder reconstructs the input data from this compressed form.

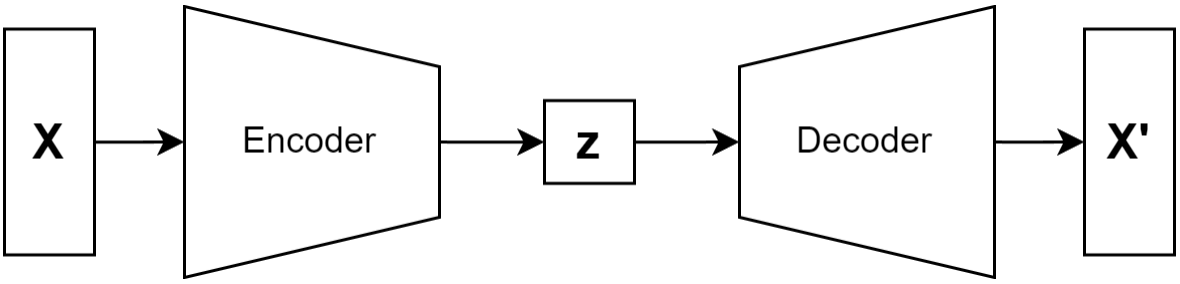


FIGURE 3.3. A simple visualization of autoencoders.

The encoder function can be defined as:

$$z = f(x; \theta), \tag{3.4}$$

where x is the input, z is the latent representation, and θ represents the encoder parameters. The decoder function is defined as:

$$\hat{x} = g(z; \phi), \tag{3.5}$$

where $\hat{x}$ is the reconstructed input and ϕ represents the decoder parameters. The training objective is to minimize the reconstruction error, typically measured by the mean squared error:

$$L(x, \hat{x}) = \|x - \hat{x}\|^2. \tag{3.6}$$

3.2.3 Variational Autoencoders (VAE)

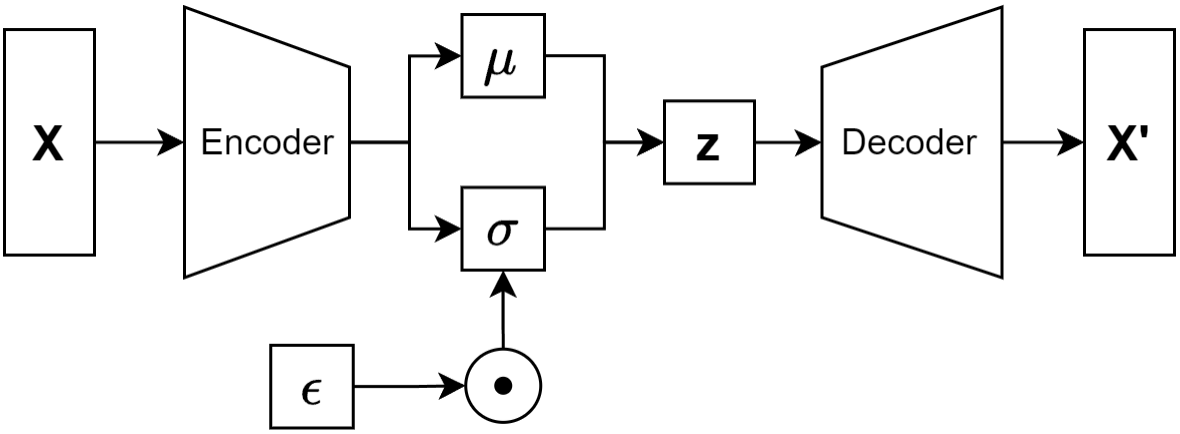


FIGURE 3.4. A simple framework for variational autoencoders.

Variational Autoencoders [18] are a special type of generative model that introduces a probabilistic twist to the traditional autoencoder. They are designed to learn the parameters of the probability distribution modeling the data. The encoder in a VAE produces not direct values of the latent variables but parameters defining a probability distribution over the latent variables.

The encoder outputs means and variances given by:

$$\mu, \sigma = h(x; \theta), \tag{3.7}$$

where h is a neural network representing the encoder. The latent variable z is then sampled from this distribution using the reparameterization trick:

$$z = \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1), \tag{3.8}$$

which allows the backpropagation of gradients through random operations ensuring differentiability.

The VAE loss function, also known as the Evidence Lower Bound (ELBO) [18], combines the reconstruction loss with the Kullback-Leibler divergence, encouraging the latent variables to approximate a prior distribution, typically a standard normal distribution:

$$\text{ELBO} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x)\|p(z)), \tag{3.9}$$

where $p(x|z)$ is the likelihood of the data given the latent variables, modeled by the decoder, and $q(z|x)$ is the encoder's approximation of the posterior.

3.2.3.1 Applications

Both AEs and VAEs have found extensive applications in sound synthesis. They are capable of learning to encode and decode audio data efficiently, thereby supporting the synthesis of new sounds based on learned audio features. Owing to their capability of learning a compressed latent space of the original audio, they are commonly used to extract complex information while reducing dimensionality of data. Nsynth [14] and DDSP [13] are based on

autoencoders that synthesizes raw audio waveforms conditioned on the latent representations. In RAVE [110], VAE was combined with adversarial networks to output various audio samples. Nowadays, many text-to-audio synthesis models (TTA) [83, 69, 111] utilize VAE for representation learning and spectrogram generation. The challenge in designing these models for audio applications lies in balancing the fidelity of sound reproduction with the richness and diversity of generated sounds, addressing overfitting, and managing the computational complexity of real-time audio processing.

3.2.4 Generative Adversarial Networks

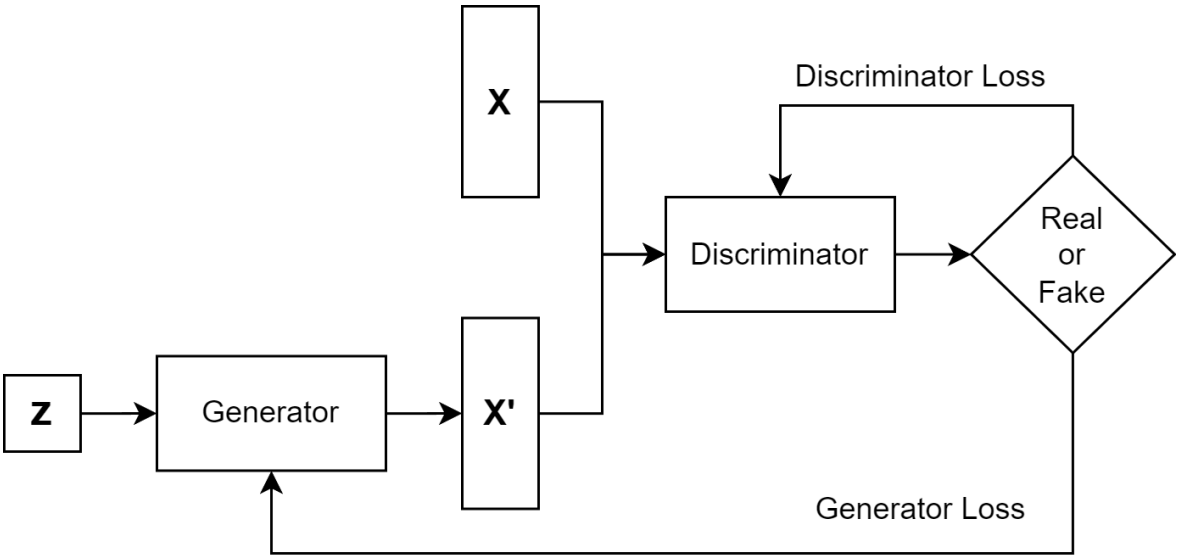


FIGURE 3.5. A simple framework for generative adversarial networks.

Generative Adversarial Networks (GANs) were introduced by Ian Goodfellow et al. [79] in 2014 as a novel approach to generative modeling. GANs consist of two neural networks, the generator G and the discriminator D, which play a minimax game. The generator aims to produce data indistinguishable from real data, while the discriminator strives to distinguish between real and generated samples.

The original GAN framework defines the loss function as a minimax game:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{3.10}$$

where x are samples from the real data distribution, and z are samples from a noise distribution used by the generator.

Although GANs have been popular in the area of image generation, they are known to be hard to train. Mode collapse [112] has become a serious issue that hinders the applications of GAN, as the generator quickly learns to generate only certain modes of the original data distribution. To address training instabilities and mode collapse issues in the original GAN, the Wasserstein GAN (WGAN) [113] uses the Earth-Mover's (Wasserstein) distance to improve the training convergence. The WGAN loss function is:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[D(x)] - \mathbb{E}_{z\sim p_z(z)}[D(G(z))] \tag{3.11}$$

WGAN-GP [114] further refines WGAN by adding a gradient penalty to enforce the Lipschitz constraint, enhancing training stability:

$$\lambda \mathbb{E}_{\hat{x}\sim p_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2] \tag{3.12}$$

where $\hat{x}$ are interpolated samples between real and generated data, and λ is a penalty coefficient.

3.2.4.1 Applications

In the domain of sound effects, WaveGAN[115] presents the first attempt at utilizing generative models. It draws reference from DCGAN[116] by changing the 2-D convolutional filters into 1-D directly to let the neural networks learn the waveform patterns from start to end. Sounds were sampled at 16000Hz but then padded into 16384 samples per second for coherence with convolutional layers because the filter sizes are to the power of 2. They also include a SpecGAN model by directly applying the DCGAN architectures to the spectrograms of the audio waveforms. By shaping the spectrograms into 256*256 resolution a square image of the spectrograms could be obtained for easier training with convolutional networks. Following this approach, Barahona et al [100] and Communita et al [99] also experimented with GAN-based generative networks for conditionally generating impulsive sounds with different sonic characteristics. It has been demonstrated that such approaches show great

capability in generating precise waveforms similar to the training set, but they usually suffer from generating a diverse range of sounds.

3.2.5 Flow-Based Models for Generative Modeling

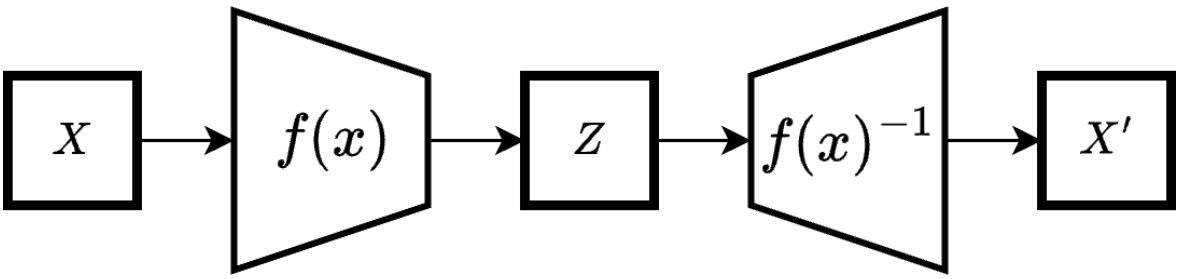


FIGURE 3.6. A simple visualization of flow-based models.

Flow-based models, or normalizing flows, utilize invertible neural networks to model the distribution of data by transforming a simple base distribution, typically Gaussian, into a complex target distribution. This transformation process is governed by a sequence of invertible and differentiable functions, denoted as f, and their inverses, denoted as f^{-1}, as shown in 3.6. The relationship between the densities of the original and transformed data is captured by the change of variables formula in probability:

$$p_X(x) = p_Z(f^{-1}(x)) \left| \det \left(\frac{\partial f^{-1}}{\partial x} \right) \right| \tag{3.13}$$

where $x = f(z)$ maps a latent variable z from the base distribution $p_Z(z)$ to the data space, and $\left| \det \left(\frac{\partial f^{-1}}{\partial x} \right) \right|$ represents the Jacobian determinant of the transformation, adjusting for the change in volume in the probability space.

The training objective is to maximize the log-likelihood of the observed data under the model, which involves optimizing:

$$\log p_X(x) = \log p_Z(f^{-1}(x)) + \log \left| \det \left(\frac{\partial f^{-1}}{\partial x} \right) \right| \tag{3.14}$$

This objective ensures that the model parameters are adjusted to increase the probability of the observed data, effectively learning the transformation function f. The optimization is performed using backpropagation to compute gradients and update the model parameters, thus enabling the flow-based model to capture the underlying data distribution accurately. Unlike GANs, which may suffer from mode collapse, and diffusion models, which often require longer sampling times, flow-based models can provide faster and more stable training processes. However, flow-based models require all transformations to be invertible and often volume-preserving or easily computable, which restricts architectural flexibility.

3.2.5.1 Applications

In the domain of sound synthesis, flow-based models have shown potential for generating high-fidelity and diverse sounds. For instance, WaveGlow [117] operates as a flow-based neural vocoder, synthesizing speech by converting mel spectrograms to audio waveforms efficiently and with high fidelity. This capability makes it an invaluable tool for applications requiring real-time speech generation. FlowWaveNet [118], on the other hand, introduces an innovative approach by using flow-based technology to directly model the waveform generation process, enhancing the clarity and realism of the synthesized speech. It offers advantages in terms of stability and consistency in audio generation compared to traditional vocoders [17], which can suffer from artifacts and distortion under certain conditions. FlowWaveNet's [118] ability to generate continuous, high-resolution audio samples illustrates the potential of flow-based models in producing natural-sounding speech and complex audio sequences, thereby enriching the capabilities of generative audio synthesis models. In the field of sound effects specifically, Sergi and Monica Villanueva [119] explored WaveFlow [120], a generative model originally effective in speech synthesis, adapted here for audio effects. The model employs a lower-dimensional mel spectrogram as a conditioner, enhancing user control and diversity in output. This method enables style transfer and was evaluated for its ability to produce high-quality, variable sounds as judged by audio experts in gaming.

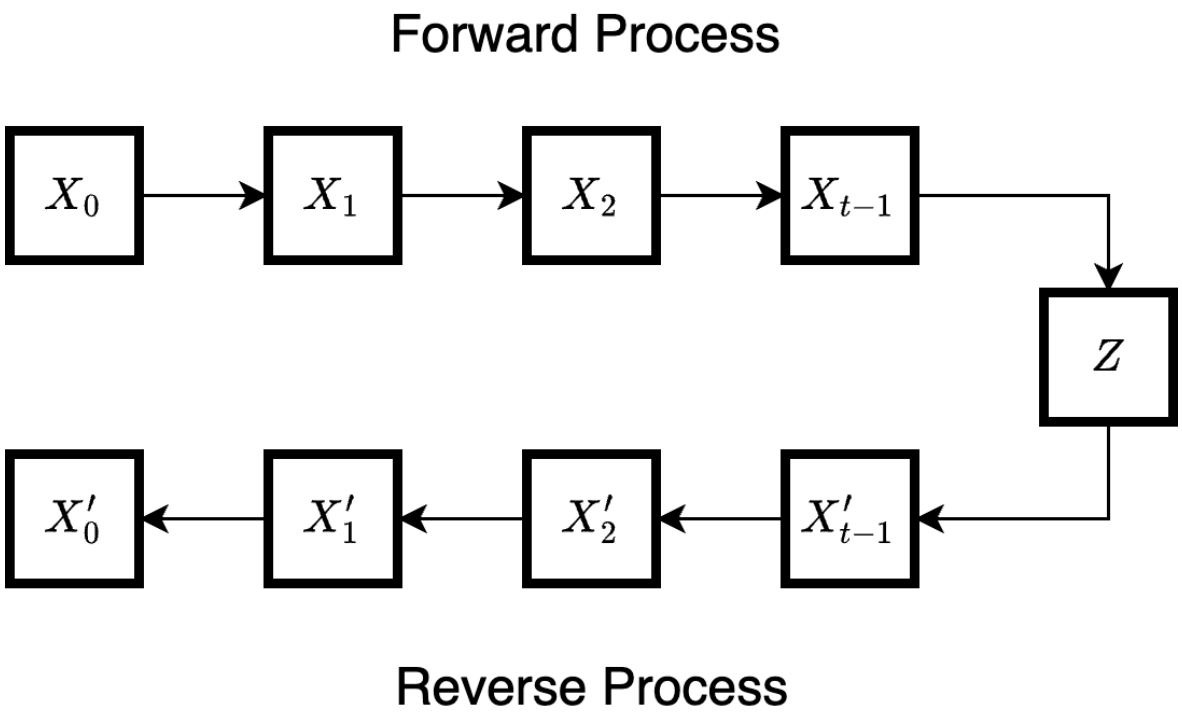


FIGURE 3.7. A simple visualization of diffusion models.

3.2.6 Diffusion Models

Diffusion models [82, 81] are a class of probabilistic generative models that iteratively transform simple noise distributions into complex data distributions. Initially introduced in the context of denoising and thermodynamic processes [121], these models have gained extensive popularity in recent years due to their ability to produce high-fidelity images. Broadly speaking, diffusion models can be separated into two processes: a forward diffusion process and a reverse diffusion process. The forward diffusion process gradually adds Gaussian noise to the data, effectively destroying information until it becomes pure noise. This process can be described as a Markov chain:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}), \tag{3.15}$$

where x_0 represents the original data, x_t is the noisy data at timestep t (x_t is usually referred as the latent space z and is illustrated as z in Figure 3.7.) β_t is a variance schedule, and $\mathcal{N}$ denotes a Gaussian distribution. The forward process is thus explicit, as we have defined the

introduced noise progressions deterministically. The link between the original data x_0 and the noisy data at any step t can be expressed through the marginal distribution:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)\mathbf{I}), \tag{3.16}$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$.

The reverse process, on the contrary, seeks to reconstruct the data by iterative denoising the noisy sample. The reverse diffusion is also modelled as a Markov chain:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)). \tag{3.17}$$

Here, μ_θ and Σ_θ are parameterized by a neural network trained to approximate the true posterior distribution $q(x_{t-1}|x_t, x_0)$. The reverse process can also be seen as the 'generation' process, as it takes the latent vector z and aims to predict the introduced noise distribution in t steps.

Diffusion models are typically trained by minimizing a variational bound on the negative log-likelihood:

$$L = \mathbb{E}_q \left[\sum_{t=1}^{T} \mathrm{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)) - \log p_\theta(x_0|x_1) \right]. \tag{3.18}$$

By simplifying the training objective, one can directly predict the noise ϵ added to the data at each step:

$$L_{\text{simple}} = \mathbb{E}_{x_0,\epsilon,t} \left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2 \right], \tag{3.19}$$

where ϵ_θ is a neural network predicting the noise component.

Unlike Generative Adversarial Networks (GANs), which can suffer from training instability and issues like mode collapse, diffusion models offer a more stable training process. The gradual denoising procedure provides a clear learning path from noise to data, which generally results in smoother convergence. Diffusion models can be easily conditioned on various types of auxiliary information, allowing for controlled generation processes. This is highly beneficial for tasks such as conditional image generation or targeted audio synthesis, where

outputs need to adhere to specific attributes. Additionally, unlike models that directly generate samples from a latent space, diffusion models produce outputs by continuously transforming noise into data. This process inherently allows the exploration of a more diverse range of intermediate states, which contributes to the variety of the final outputs. Owing to these advantages, diffusion models have excelled in the task of data generation in recent years, especially in image synthesis such as Stable Diffusion [122], DALL-E 2 [123] and Mid Journey [124].

3.2.6.1 Applications

In the field of audio synthesis, diffusion models are also making significant strides, with applications spanning neural vocoding, text-to-audio generation, and sound effects synthesis. Models like DiffWave [125] and WaveGrad [126] have been instrumental in high-fidelity speech synthesis, transforming mel-spectrograms into realistic waveforms with high efficiency. In sound effects synthesis, models such as DiffSound [83] leverage the strengths of diffusion processes to create diverse and contextually rich audio effects from textual descriptions. Additionally, latent diffusion-based approaches like AudioLDM [69] and Make-An-Audio [111] have demonstrated their capability in text-to-audio generation, given large amounts of training audio datasets.

3.3 Hybrid Models

3.3.1 Neural Hybrid Architectures

As mentioned above, each generative model has its advantages and drawbacks in certain areas. To this end, many researchers have tried to incorporate more than one generative model into one architecture. Specifically, VAEs are frequently used as the *first stage* in hybrid architectures, learning a concise latent representation of data by encoding inputs into a lower-dimensional distribution. This latent space captures global structure such as overall pitch, timbre, or semantic content, while remaining tractable to manipulate or sample from.

Subsequently, other generative methods are usually employed to transform the latent codes back into high-fidelity outputs.

For example, RAVE [110] employs a VAE+GAN [127] approach by first training a variational autoencoder architecture to learn the latent representations of the audio and the decoder is then fine-tuned using a discriminator. This approach not only ensures capturing the detailed acoustic features of the audio but also allows the decoder to output diverse audio timbres. This architecture has proven to work very well with many types of sounds and has inspired numerous following studies [128, 129, 130]. The Flow Synthesizer [131], illustrates how VAEs and Flow-based models could be incorporated together to connect between the auditory space and the parameters space. Under this setup, their approach achieves macro-level parameter control as well as automatic parameter estimation with a predefined synthesizer.

Additionally, in the field of text-to-audio synthesis, a great many works have employed a hybrid approach [83, 69, 132] to audio modelling. First training a VAE to learn the latent representation of similar audio. Then, train another neural vocoder that is capable of converting Mel spectrograms into audio waveforms. Finally, another generative model such as diffusion model [82, 81] could be used to learn and generate the latent representations. On the other hand, WaveFlow [120] tries to merge autoregressive conditioning with normalizing flows to achieve high-quality raw audio generation. It uses a two-dimensional structure where one dimension applies an invertible flow transformation to audio segments, and the other handles a partial autoregressive factorization to capture fine temporal dependencies. This combination yields a parallel, likelihood-driven model capable of producing realistic speech at high fidelity without the full sequential cost of purely autoregressive approaches.

3.3.2 DSP+Neural Hybrid Models

So far the density estimation models we've mentioned are the most common deep generative models being used across various domains. In the sound generation task, they are typically used to directly learn and generate either audio waveforms or time-frequency representations. Another type of synthesis model combined with deep learning has gained much popularity

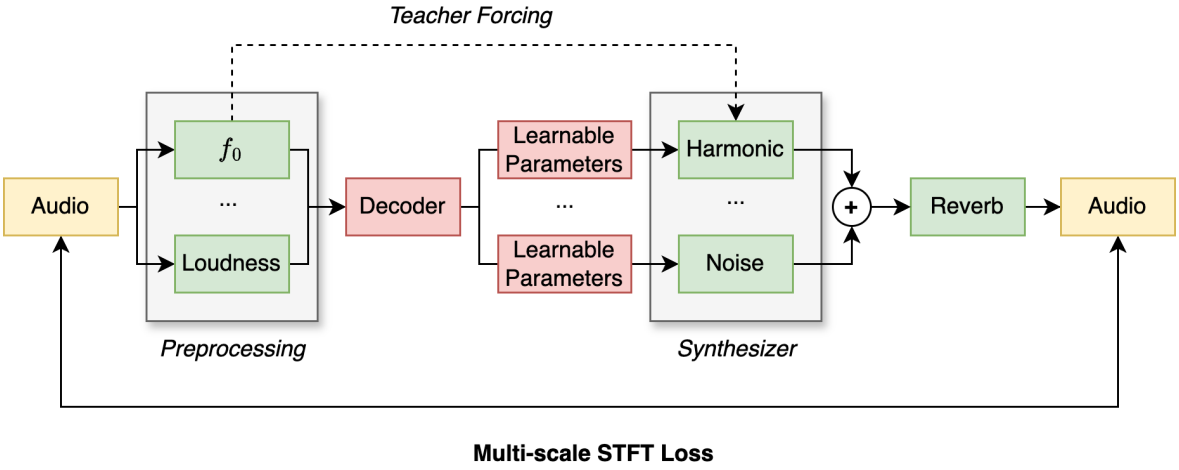


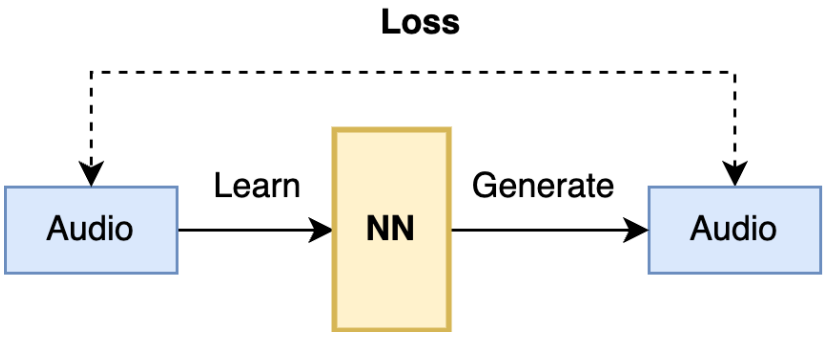
FIGURE 3.8. A simple visualization of DDSP models.

in the music domain by learning to estimate parameters of some synthesis algorithms given a sound dataset. For example, DDSP[13] utilizes an autoencoder structure by extracting pitch and loudness and other acoustic information from the sounds with an encoder, followed by a decoder that learns to resynthesize the audio with a harmonic plus noise synthesis algorithm.[36]. This architecture gives us direct access to the deterministic synthesizer parts after being trained so that we can utilize them to generate new sounds with controls over pitch and loudness.

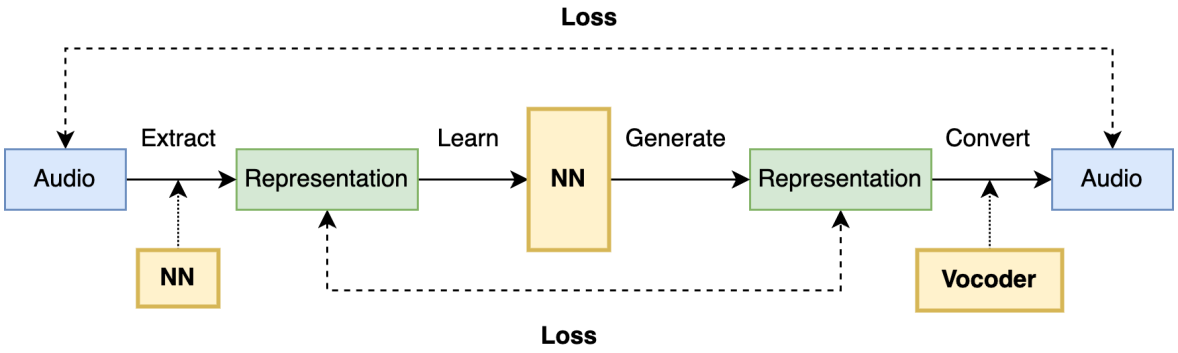
Furthermore, Anton Lundberg[133] explored using this approach to generate procedural engine sounds by extracting the engine cycle's phase as input. Furthermore, the author also paired it with a differentiable transient synthesizer[134] to model the initial transient sounds portion of car engines. However, owing to the complexity and diversity of general sound effects, we have yet to see any mature research to model source-agnostic sounds using this approach. Nevertheless, this modular and interpretable approach to modelling audio signals utilizing traditional DSP components is very rewarding, especially given its lightweight architecture design, few computation requirement, and flexibility of adapting to different sounds using different synthesis algorithms such as waveshaping [135] and granular synthesis [136].

3.4 Audio Representations

3.4.1 Raw Audio Waveforms



(A) Raw audio waveform modelling with neural networks.



(B) Time-frequency representations modelling with neural networks.

FIGURE 3.9. A visualization of raw audio modelling vs audio representations modelling.

In the age of digital technologies, audio waveforms are usually recorded and synthesized with fixed sampling rate (typically 44.1kHz for mainstream audio devices, 8/16kHz for phone calls, 48/96kHz for high-quality music recordings) and bit rate (it varies depending on the media, 128–160 Kbps for streaming services, 96–320 Kbps for MP3s, and 1411 Kbps for audio CDs). It is thus natural to model the raw audio waveforms directly with generative models, as such digital samples provide the complete description of the sounds. In the early age of neural audio synthesis, autoregressive models such as Wavenet [77] and SampleRNN [78] have demonstrated success in raw audio modelling of generating sounds sequentially. New audio samples are conditioned on previous samples, resulting in a high demand for memory buffers

and computational needs. However, it was notoriously slow to generate audio in the inference stage with autoregressive models, as parallel processing of audio samples is not enabled. To this end, WaveGAN [115] was proposed to generate sounds that achieve sound generation in parallel. However, there is a well-known issue called 'checkerboard artifact' when generating data with convolutional neural networks, where artifacts occur in the adjacent boundaries between processed kernels of data. In raw audio generation, this results in unwanted audible tones. To this end, phase shuffle was proposed to remove the pitched noise artifacts.

However, different from images, which could be used directly for training, audio signals have a high temporal resolution, which makes it inherently difficult to model long-term global structures with generative models. Generating audio in the raw waveform domain is usually computationally expensive [77]. Meanwhile, as raw waveforms don't reflect spectral contents, which are intuitive indicators of the sonic properties, the generated outputs may not correspond to human hearing. Additionally, capturing long-range correlations in raw audio is notoriously difficult. Many autoregressive approaches [77, 137, 138] rely on large receptive fields (achieved via large dilation or hierarchical approaches) to address this, but it adds complexity and training challenges. Furthermore, raw audio waveforms are less interpretable than symbolic or time-frequency representations (e.g., MIDI or spectrograms). Gaining high-level control (like pitch manipulation, timbre, or timing) is more challenging when working directly in the time domain without explicit conditioning mechanisms.

3.4.2 Time-frequency Representations

Another common technique for neural audio synthesis is to model its time-frequency representations (TFRs). For example, short-time-Fourier-transform (STFT) is commonly used in audio deep learning [139]. The Short-Time Fourier Transform (STFT) is a mathematical technique used to analyze the frequency content of a signal over time and can be converted back to audio waveforms with Inverse STFT (ISTFT). Unlike the standard Fourier Transform, which provides a frequency representation of the entire signal, the STFT divides the signal into short, overlapping time segments and applies the Fourier Transform to each segment. This process yields a time-frequency representation, where both temporal and spectral information

are captured. The STFT of a discrete signal can be defined as:

$$X(m,k) = \sum_{n=-\infty}^{\infty} x[n]w[n-mR]e^{-j\frac{2\pi kn}{N}}, \tag{3.20}$$

where $X(m,k)$ is the STFT at time frame m and frequency bin k, $x[n]$ is the input signal, $w[n]$ is the window function (e.g., Hamming, Hanning) of length N, R is the hop size (step size between successive frames), and $e^{-j\frac{2\pi kn}{N}}$ represents the Fourier basis function. The computed STFT is a two-dimensional complex-valued signal, which makes directly modelling STFT [140] extremely challenging and inconvenient with machine learning models, as common activation functions and optimization techniques are built for real-valued data. Therefore, STFTs are usually separated into two parts:

$$X(n,k) = |X(n,k)| \cdot e^{j\phi(n,k)} \tag{3.21}$$

where $|X(n,k)|$ is the **magnitude** (or amplitude) of the STFT, and $\phi(n,k)$ is the **phase** of the STFT. In generative sound modelling, magnitude spectrograms are usually used as they capture perceptually relevant information about the energy distribution across frequencies, which is crucial for timbre and loudness perception. Phase information, on the other hand, is often ignored due to its complexity, noise sensitivity, and low impact on perceived audio quality for many tasks. Therefore, existing research has mainly focused on modelling the 2-D magnitude spectrograms with generative models. More specifically, SpecGAN [115] was the first attempt at learning to generate the magnitude spectrograms extracted from small audio clips discarding the phases. The generated spectrogram images can then be converted back to audio with some phase reconstruction algorithms such as the Griffin Lim algorithm [17] as shown in Figure 3.9b. Furthermore, Marafioti et al. [141] explored another phase reconstruction algorithm using Phase-gradient heap integration(PGHI) [142] which has shown better audio reconstruction quality compared to SpecGAN using Griffin Lim. It was also shown that log-magnitude spectrograms outperform linear scale spectrograms for sound modelling tasks. Owing to the development of speech synthesis, in recent years, neural vocoders [96, 95, 125, 126, 143] has become a mature technology that could convert spectrograms of almost any audio source back into waveforms. Therefore, many research have started focusing on generating the spectrogram representations alone [105, 69, 144].

3.4.3 Other Time-Frequency Representations

Although often omitted for simplicity, phase information can sometimes be important, such as in scenarios requiring high temporal precision, such as transient sound reproduction or high-fidelity audio synthesis, where accurate phase reconstruction is essential to avoid artifacts [145, 146]. Additionally, as STFTs are obtained by windowing audio frames sequentially, they suffer from the time-frequency resolution trade-off, where the choice of window size directly impacts the balance between temporal and spectral precision [147]. A shorter window provides better time resolution, allowing for accurate localization of transient events (e.g., drum hits or speech plosives) but results in poorer frequency resolution, which makes it difficult to distinguish closely spaced frequency components. Conversely, a longer window improves frequency resolution, enabling finer discrimination of spectral details, but at the cost of reduced time resolution, which can smear transient events and obscure their exact timing. To this end, Engel et al [12] explored "Instantaneous Frequencies" [148], the precise frequency content of an audio signal at any given instant, unlike traditional Fourier transforms which provide frequency information averaged over a window of time. This capability makes IF particularly valuable for synthesizing audio with high temporal resolution. This is essential for capturing the transient and subtle nuances of sound that occur in very short time frames, which are often lost in other spectral representation methods.

Another commonly used TFR to address the TF trade-off issue is Wavelet transforms [149]. Unlike STFT, which uses a fixed window size, wavelet transforms use variable-sized windows to provide multi-resolution analysis. This allows them to achieve high time resolution for high-frequency components (e.g., transients) and high-frequency resolution for low-frequency components (e.g., sustained tones). In audio synthesis, wavelet-based generative models have been shown to be very effective. For instance, the synthesis of normal heart sounds using GANs combined with empirical wavelet transform [150] demonstrates the utility of wavelets in capturing the nuanced features of biomedical signals. The work in [151] also employs wavelet trees to synthesize sound textures by learning and resampling wavelet coefficients, enabling detailed reconstruction of audio characteristics. However, in sound synthesis, wavelets also suffer from drawbacks in modelling harmonic structures, computational complexity [152],

phase reconstruction issues [153], and sensitivity to parameter choices [147]. Additionally, they struggle with high sampling rates and long-range dependencies and can introduce artifacts during reconstruction, making them less ideal for high-fidelity audio synthesis compared to other methods like STFT or neural vocoders.

3.4.4 Neural Audio Representations

Neural audio embeddings, also referred to as neural audio representations [154] or latent representations, have emerged as a crucial component in modern audio synthesis research. These embeddings provide compact, abstract characterizations of audio signals, enabling models to capture salient perceptual and acoustic features more effectively than raw, high-dimensional waveforms. By projecting audio data into a lower-dimensional latent space, generative systems can operate more efficiently and accurately, particularly when modelling complex signals such as speech and music.

Let $\mathbf{x} \in \mathbb{R}^T$ denote a raw audio waveform of length T. A neural encoder E_θ, parameterized by θ, maps $\mathbf{x}$ to a latent embedding $\mathbf{z} \in \mathbb{R}^d$ (or, in some cases, $\mathbf{z}$ belongs to a discrete codebook $\mathcal{V}$):

$$\mathbf{z} = E_\theta(\mathbf{x}). \tag{3.22}$$

A decoder D_ϕ, parameterized by ϕ, reconstructs the original audio from this latent representation:

$$\hat{\mathbf{x}} = D_\phi(\mathbf{z}). \tag{3.23}$$

When trained jointly, (E_θ, D_ϕ) act as an autoencoder, minimizing an objective function $\mathcal{L}$ that encourages $\hat{\mathbf{x}}$ to be perceptually or quantitatively similar to $\mathbf{x}$. A common choice might include a time-domain error (such as $\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2$) plus a frequency-domain loss (e.g., multi-resolution STFT). A great many representation models were trained in this way, such

as Nsynth [14] which uses a Wavenet-style encoder [77], SoundStream [155] that learns a residual vector quantization of audio, and Audio Spectrogram Transformer (AST) [156] which is a transformer-based model operating on 2D time-frequency representations. The learned embeddings capture both local (e.g., dynamics-level or pitch-level) and global (timbral features) attributes. This abstraction can improve model generalization by focusing on high-level audio features rather than noise or irrelevant details.

Although neural embeddings provide a great way to interpolate and fine-tune the generated sounds, they still have to be learned from either waveform or time-frequency representations obtained from the waveforms. On the other hand, extracted acoustic features such as pitch and loudness are very intuitive properties of sounds that could be used directly for spectral modelling [13]. However, many low-level acoustic features, including temporal features like zero-crossing rates and spectral features such as spectral centroid and spectral envelope, are mainly used for analysis [157] as most of them don't offer an invertible transform. On the other hand, for the task of sound effects generation, some high-level acoustic descriptors made by humans could be very beneficial for conditioning, such as the materials of contact surfaces for footstep sounds demonstrated by Communita, Phan, and Reiss [99].

3.4.5 Summary

Audio representation is a pivotal component in machine learning-based sound synthesis, as it determines how raw audio signals are processed, interpreted, and reconstructed by models. In the context of sound effects synthesis, where the diversity of sounds ranges from environmental noises to Foley sounds, choosing the appropriate representation is essential to balance fidelity, computational efficiency, and perceptual relevance. Unlike music and speech, sound effects often lack temporal regularity and exhibit wide spectral variability, posing unique challenges for their representation.

3.5 Training of Generative Sound Models

3.5.1 Loss Functions

When modelling audio waveforms, there are a number of loss functions that could be utilized. A good resource that collects the audio-focused loss functions is Auraloss [1]. Specifically, they could be categorized into time-domain loss, frequency-domain loss, and perceptual loss. In the sections below, I briefly discuss the usage of these loss functions in neural audio synthesis.

3.5.1.1 Time-domain Loss

Time-domain losses typically take the form of L_1 or L_2 distance between predicted and target waveforms. This class of objectives enforces direct alignment of amplitude and phase at each time step, thereby serving as a straightforward means of guiding the network. However, in cases where the audio signal exhibits rich harmonic or transient structure, time-domain losses alone may not capture subtle spectral details. Consequently, relying solely on sample-level distance has the potential to produce outputs that are amplitude-correct yet lacking in high-frequency nuances, especially for longer, more complex signals.

3.5.1.2 Frequency-domain Loss

To mitigate these limitations, many recent approaches employ frequency-domain losses that compare predicted and ground-truth signals in a transform domain. A frequent choice is the Short-Time Fourier Transform (STFT), wherein models minimize the discrepancy in spectrogram magnitude (and sometimes phase) across different time-frequency bins. One particularly effective variant is the multi-resolution STFT (MR-STFT) loss, which calculates spectrogram differences over multiple window sizes and hop lengths. This multi-scale formulation captures both transient details (short windows) and broader harmonic structures (longer windows), thereby improving fidelity in diverse audio contexts. In neural vocoders, such as HiFi-GAN [95], DDSP [13], and UnivNet [158], MR-STFT losses have played a

[1] https://github.com/csteinmetz1/auraloss

substantial role in producing natural-sounding speech, outperforming purely time-domain approaches on both objective and subjective measures. For many spectrogram generation tasks, loss functions could either be directly applied to the TFRs [115, 141], or be applied to the reconstructed audio waveforms after signal reconstruction [159]. A common choice is the Mean Square Error (MSE) loss for evaluating the difference between reference and generated spectrograms [160]:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{3.24}$$

where y_i is the target spectrogram, and $\hat{y}_i$ is the generated spectrogram. This loss is widely used for regression tasks and ensures smooth gradients for training [160]. Another commonly used loss function is Log-Spectral Distance (LSD), which compares the log-scale magnitude of spectrograms:

$$\mathcal{L}_{\text{LSD}} = \frac{1}{N}\sum_{i=1}^{N}\left(\log|y_i| - \log|\hat{y}_i|\right)^2. \tag{3.25}$$

This loss is particularly effective in speech enhancement and synthesis models [144]. Additionally, adversarial loss has become widely used in modern NAS systems [110], as it introduces higher sample diversities in generated outputs [161]. Adversarial Loss employs a discriminator to distinguish between real and generated spectrograms and can be written as:

$$\mathcal{L}_{\text{Adv}} = \mathbb{E}[\log D(y)] + \mathbb{E}[\log(1 - D(\hat{y}))]. \tag{3.26}$$

3.5.1.3 Perceptual Losses

While frequency-domain losses address issues of timbral accuracy and clarity, perceptual losses more explicitly incorporate psychoacoustic factors. Many systems adopt Mel-frequency-based metrics, reflecting the non-linear frequency sensitivity of human hearing. By penalizing errors according to the Mel scale, these losses better align with how listeners perceive differences in pitch and timbre. Other studies employ deep feature losses, in which a pre-trained network (such as an audio recognition model [88] or classifier [162]) computes

embeddings for the target and predicted signals; the network is then trained to minimize the distance between these embedding vectors. Such embeddings capture higher-level perceptual attributes that may not be fully represented by amplitude or spectrogram-based metrics, leading to more convincing timbral and stylistic reproduction.

3.6 Current Trends for Neural Audio Synthesis

In the field of sound synthesis, generative models are usually used in several tasks: sample generation/augmentation, audio inpainting, stylized generation, audio recording enhancement, cross-modal sound synthesis, and parameter-controlled synthesis. Broadly speaking, the focus of such tasks could be classified as synthesis-oriented, which aims to achieve better sound quality or fidelity of the output, and control-oriented, which aims to achieve better nuanced controls over the sound characteristics. In the sound effects domain

As introduced in Section 3.1, unconditional generative models try to learn a mapping from latent space to the true data distribution. However, the latent space is usually uninterpretable [163, 164] and difficult to directly use for new sample generation. In machine learning, conditional probability is usually used to associate specific data attributes or dependencies to existing frameworks to achieve controllable generation. Conditional generative models in neural audio synthesis utilize conditioning variables to guide the generation process, ensuring that the output adheres to specific attributes or features. This conditioning is mathematically achieved by modifying the generation function to depend explicitly on these conditioning variables. In any generative model, a mapping is usually learned from both noise (or latent variables) and conditioning variables to the output space, which can be represented as:

$$x = g(z, c; \theta) \tag{3.27}$$

Here, x is the generated data (such as an audio waveform), z is a latent variable or noise vector, c represents the conditioning variables (such as the category of the sound recording), and θ denotes the model parameters. Similar to unconditional generation as covered in

Section 3.1, the training objective typically involves maximizing the likelihood of the data given the conditioning variables or, equivalently, minimizing a loss function that measures the discrepancy between the generated data and real data, conditioned on c. This is often formulated as:

$$\min_{\theta} \mathcal{L}(x, g(z, c; \theta)) \tag{3.28}$$

where $\mathcal{L}$ is a loss function (e.g., mean squared error, cross-entropy). The conditioning information c effectively guides the generative process, allowing the model to generate outputs that are not just realistic but also aligned with specific desired attributes.

In the domain of sound effect synthesis, the nature of the conditioning information can vary significantly depending on the synthesis objective. This information might include discrete class labels that categorize the sounds, feature vectors that encapsulate the acoustic properties derived from audio recordings, or supplementary data from other modalities to enhance the generative process. This section introduces the methodology behind conditional sound generation, outlining the four predominant types of conditioning approaches utilized in contemporary sound synthesis.

3.6.1 Class-conditional Sound Generation

Class-conditional sound generation [105, 100] leverages neural networks to produce diverse types of sound effects based on discrete class labels. This approach typically utilizes one-hot encoded vectors to represent distinct sound categories, such as footsteps, raindrops, or explosions. By conditioning the generative process on these categorical labels, the model is trained to output audio samples that correspond to the characteristics of the specified class. Class conditioning in generative models often utilizes one-hot encoded vectors to represent discrete class labels. A one-hot vector is defined in the context of machine learning for representing categorical variables where each class is mutually exclusive. For a dataset with N number of classes and C represents a certain class, the one-hot vector e_i could be represented as:

$$e(i) = \begin{cases} 1 & \text{if } i \in C, \\ 0 & \text{if } i \notin C. \end{cases} \tag{3.29}$$

These one-hot vectors typically undergo a transformation through an embedding layer to obtain a dense representation suitable for integration with the model's architecture. The embedding effectively converts sparse categorical information into a form that enhances the model's ability to generate contextually relevant outputs. For instance, in an adapted conditional WaveGAN [99], the transformed embedding from the one-hot vectors are concatenated with the processed latent representations to be formed as the input for the generator to output synthetic audio. Similarly, the embeddings could also be concatenated with the audio waveforms that the discriminator uses to distinguish between real and generated (fake) data samples as well as to assess whether the generated samples are consistent with specific conditioned categories represented by the one-hot vectors.

This technique can be further refined through two distinct conditioning strategies. The first method conditions on broad sound categories, enabling the synthesis of general types of sounds within predefined groups, like Foley sound effects [165]. The second, more granular method conditions on specific attributes or types within these categories, such as the type of shoe for footsteps [99], the types of audio textures [166], or the emotions of knocking sounds [100], allowing for more detailed and context-specific audio generation.

Class-conditional sound generation is advantageous in scenarios where detailed labeling of sounds is absent as it offers a straightforward means to direct the types of sounds being generated. However, the discrete nature of class labels poses challenges for interpolation and nuanced control over the timbre of newly generated sounds, due to the categorical nature of such conditioning information.

3.6.2 Feature-conditioned Sound Generation

Feature-conditioned sound generation uses acoustic features extracted from audio recordings as conditioning information to guide the sound synthesis. Different from class-conditional

sound generation, this approach usually employs continuous feature vectors rather than discrete labels as the conditioning. For example, pitch vectors predicted using prediction algorithms like CREPE [167] or YIN [168] are usually used as conditioning information to model musical instrument sounds [14, 12]. Amplitude envelopes [13] is also very effective in guiding sound synthesis for many sounds that require precise temporal dynamics such as impacts or hits [169], while spectral centroid was used to control the brightness or sharpness of the sounds [170]. Apart from explicitly extracted audio features, complex temporal representations that describes the timbre of audio such as "warmth" or "depth" [171] could also be employed as conditioning features to generate audio with interpretable timbre descriptions. Furthermore, soft labels [84] that are implicitly derived from the audio waveforms using large audio representation models can also be used to achieve attribute control while maintaining good audio quality.

Nowadays in the field of neural audio synthesis, feature-conditioned sound generation has become indispensable for generative sound modelling. Extracting these features has become increasingly accessible due to advances in audio processing techniques. However, as the dimensionality or complexity of the feature set increases, the computational burden for neural networks to effectively learn and model these characteristics also rises. This can lead to increased training times and higher resource demands, potentially limiting the scalability of this approach in resource-constrained environments. Additionally, these continuous features demand meticulous tuning and integration into the neural network architecture, particularly when synthesizing diverse sound categories. This is because different types of sounds may necessitate distinct feature extraction configurations to be effective, especially for the sounds of Foley, where the acoustic properties can vary widely. Thus, finding a single feature extraction approach that universally describes such diverse sounds can be challenging, requiring adaptive strategies to ensure effective influence on the synthesis process without compromising the model's ability to generalize.

3.6.3 Descriptor-based Control for NAS

In the field of music generation, latent space has been explored to match perceptual sonic characteristics to achieve controlled sound synthesis. For example, [172] studied the mapping of latent space representations onto a pre-defined perceptual rating matrix. This approach enforces the latent vectors to associate with the topology of the 'timbre' space of multiple instruments. However, adapting this approach to the domain of sound effects requires extensive work on establishing the unanimous perceptual rating matrix.

[129] on the other hand, leverages Fader network [173] to learn a compressed pseudo latent space that contains audio features invariant to the chosen salient musical features such as RMS and spectral centroid. The features for control are then reintroduced after the model has been trained by the users. This approach has enabled expressive timbre control while providing a new way of sound generation.

3.6.4 ControlNet-approach towards NAS

Beyond selecting what to condition on, diffusion-based NAS also benefits from how conditioning is injected. Inspired by ControlNet in image synthesis [174], a practical pattern is to keep the pre-trained denoiser frozen and add a parallel control branch that encodes a time-resolved control map (e.g., ADSR/amplitude envelopes, onset/transient masks, rhythmic activations, F0 or spectral-tilt trajectories, band-energy curves, or time–frequency masks). The branch injects multi-scale residuals into the denoiser during each denoising step (often via zero-initialized 1×1 convolutions for safe start-up), enforcing local temporal structure while preserving global controls as shown in Stable-V2A [175]. Crucially, this mechanism is complementary to class labels, continuous features, and descriptor sliders: labels select content, features and descriptors bias timbre or dynamics, and the control map pins down when and where structure is expressed.

3.6.5 Multi-Modal Sound Generation

Multi-modal sound generation utilizes inputs from non-audio modalities such as text, images, and videos to synthesize sounds. While this approach encompasses various types of input, image-to-audio generation [176] has seen limited engagement due to the fundamental disparity between the static nature of images and the dynamic temporal characteristics of sound. The upcoming sections will delve into the latest advancements in text-to-audio (TTA) and video-to-audio (V2A) generation, highlighting how these modalities are leveraged to enrich sound synthesis.

3.6.5.1 Text-To-Audio Generation

Text-to-Audio (TTA) generation is an emerging field within audio synthesis that focuses on creating sound based on textual descriptions. Originally used for speech synthesis [177, 178, 179], text inherently carries semantic meanings, making it a powerful modality for capturing the subtle nuances [180] in audio synthesis. In the field of sound effects synthesis, TTA has received much attention since the introduction of text-conditioned sound synthesis such as Diffsound [83], which uses compressed textual information as input and leverages diffusion models [82] to generate the tokens of Mel spectrograms. Owing to the development of self-supervised learning [181] and contrastive learning [182], text and audio cross-modal representations were proposed and quickly gained popularity in the field of audio synthesis. For example, CLAP [89] has been employed as joint audio-text embedding space as conditioning to drive sound synthesis in many research, such as AudioLDM [69] and AudioLDM-2 [132], which offers the state-of-the-art synthesis quality in TTA task. Similarly, AudioGEN [108] explores continuous autoregressive modelling of audio waveforms conditioned on texts, while AudioLM [183] proposes a tokenization technique to model audio waveforms discretely by treating audio as a language modelling task. Nowadays, owing to the efficiency of modelling audio representations in smaller forms, many TTA systems employ latent diffusion models [122] such as Make-an-Audio [111] and AudioLDM [69].

TTA offers many benefits in sound effects synthesis. There is a vast availability of datasets featuring text-audio pairs which facilitates the training of models. Platforms like Vggsound [184], Audioset [185], and Clotho [186] provide a rich source of aligned audio-text data. Meanwhile, advancements in audio representation learning like CLAP [89] provide robust frameworks for aligning audio with textual descriptions, which makes it intuitive to generate sound based on text. This cross-modal alignment leverages the semantic richness of text to guide the sound synthesis process effectively. However, While text can provide a general outline or mood, it lacks the ability to specify detailed temporal dynamics that are crucial for realistic sound synthesis. This can result in a synthesis that might not fully correspond to the subtle temporal variations found in natural audio. Additionally, text operates in a sequential format without inherent temporal dynamics, which contrasts sharply with the high temporal resolution of audio. This fundamental difference can make it challenging to achieve seamless control over time-domain sound synthesis directly from text. Finally, textual descriptions may not always capture the full spectrum of sonic nuances, particularly for abstract or complex sounds like whooshes or environmental ambience. This semantic gap can lead to discrepancies between the intended and generated sounds, which makes it difficult to synthesize audio that faithfully reflects nuanced textual descriptions.

3.6.5.2 Visual-To-Audio Generation

Visual-to-audio generation entails the creation of audio that is not only aligned with silent video content but is also coherent in terms of timing and meaning. Traditionally, this alignment was achieved manually by Foley artists [20] who would watch scenes and perform the corresponding sound effects using different objectives in real-time, which results in a highly labour-intensive and skilled task. With advancements in cross-modal machine learning [187], synthesizing audio that accurately accompanies video content has become more feasible. Unlike text-to-audio synthesis, which primarily deals with translating semantic information from text to sound, visual-to-audio generation is a more complex challenge as it also requires precise synchronization of audio with the visual progressions in the video. In the early age of V2A research, the modalities of video and audio are usually treated differently. The video embeddings are usually extracted using CNNs to distill key elements from videos [169, 188,

189, 190], which are then used as conditioning input to synthesize audio modelled by RNNs networks [169, 191, 192]. With the advancement of large foundation models [193], joint representations [188, 194, 195] have become a popular approach in sound modelling. This can usually be done by concatenating the visual embeddings and audio embeddings to form a joint embedding representation. With the significant progress made in large language models, many studies now leverage text as a medium to bridge video and audio representations. For example, V2A Mapper [176] projects the visual-text embeddings from CLIP [196] onto the text-audio embeddings from CLAP [89] by incorporating a mapping operator learned by neural networks. Very recently, video-audio joint representations such as CAVP [197] were proposed to use directly synthesised sounds guided by video inputs.

In addition to the mapping of semantics between video and audio, the synchronization of video and audio modalities is a well-known difficult task. This is due to the large discrepancy of temporal resolutions between video (which is usually 30-60 frames per second) and audio (which is usually 44.1-96 samples per second). To this end, early works explored video frame interpolation and audio frame downsampling [198, 101]. Recently, Stable-V2A [175] proposed an effective temporal alignment approach by training a separate RMS mapper network that derives RMS (Root Mean Squared) envelopes from videos as conditioning information to guide the audio synthesis.

3.7 Evaluation Techniques for NAS

Evaluating generative audio systems presents unique challenges, partly due to the inherently subjective nature of audio perception [199]. Unlike tasks like audio classification where a firmly correct or wrong outcome can be defined, the acceptable outputs for generative audio can vary widely, thus making standard benchmarks difficult to establish. The complexity of audio evaluation arises from a few key areas: the lack of a precise ground truth [200], the failure of many objective metrics to capture the full perceptual qualities of sound [201], and the inherently subjective nature of personal preferences over the generated sounds [202]. For this reason, a synthesis algorithm could be evaluated based on the following metrics:

- **Sound Quality**: refers to the overall auditory experience of a sound effect. It involves subjective evaluations from listeners to assess the realness and authenticity of the sound—whether it convincingly represents what it intends to, such as footsteps or other sound effects. This quality is often assessed through listener surveys or testing panels where individuals rate the sound based on their perceptions.
- **Sound Fidelity**: This encompasses several aspects of a sound's technical and perceptual quality. For example, it can refer to the distinctness and sharpness of a sound, free from any distortions or muddiness. The generated output should not contain much unwanted noise or sound robotic, as if they were recorded rather than synthesized by machines. As a generative modelling task, the generated output should not deviate perceptually and statistically too far from any of the samples from the dataset we're training them on.
- **Sound Diversity**: For synthesis algorithms, sample diversity refers to the ability of the system to generate a wide range of different yet plausible outputs. A high diversity in generated samples ensures that the algorithm can produce varied sounds across different runs, avoiding repetition and increasing the richness of the audio output. This is particularly important in generative models to prevent overfitting to specific sound profiles and to enhance the model's robustness to various inputs.

To evaluate NAS models, many works have been established by comparing the different subjective and objective metrics [203, 204, 205, 206, 207]. The evaluation metrics can be broadly split into three categories: **sample-wise sound comparison**, **distribution distance measurement**, and **subjective listening tests**.

3.7.1 Sample-wise reconstruction error

Sample-wise reconstruction error refers to comparing the pair-wise difference between the generated sound and the reference sound. However, directly comparing waveforms in the time domain can be problematic due to phase differences. Even if two sounds are perceptually identical, slight shifts in phase can lead to large differences in a time-domain loss calculation. Since spectral loss typically focuses on the magnitude of the Fourier transform, it is inherently

more robust to such phase variations. Therefore in the audio domain, it is common to apply the mean absolute error (MAE) or mean square error (MSE) on the time-frequency representations such as spectrograms or Mel spectrograms [208]. As frequency domain contents align more closely with human hearing and spectral reconstruction error is better suited for analyzing complex sounds that consist of multiple overlapping frequencies [209], most sample-wise evaluation techniques are based on spectral differences. In earlier research, log-spectral distance (LSD) is usually used to measure the difference in spectral contents, owing to its correlation with human hearing. A more refined version, as introduced in Section 3.4, multi-scale STFT difference becomes a common perceptual metric that has been used to train [13] and evaluate [170] NAS systems. Root Mean Square Error (RMSE), on the other hand, is a quantitative measure used to assess the difference between values predicted by a model or system and the actual values observed. Specifically in audio processing, RMSE is utilized to evaluate the accuracy of audio signals reconstructed or synthesized by audio systems or models. The RMSE for audio signals can be calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2} \tag{3.30}$$

where N is the total number of samples in the audio signal, x_i represents the actual value of the audio sample at index i, $\hat{x}_i$ is the predicted or synthesized value at index i. Root mean square error (RMSE) provides a straightforward way to measure the magnitude of error between the synthesized or processed audio and the original audio, giving an overall indication of the sound quality or the effectiveness of audio processing algorithms.

However, as Vinay & Lerch [203] pointed out, spectral-domain difference calculation is usually used both as a loss function and as an evaluation metric [170]. Additionally, the spectral difference may not correlate well with human perceptions (e.g. a high spectral difference does not always mean the two sounds are not similar). Therefore, sample-wise reconstruction error is usually used when precise sound reconstruction is required.

3.7.2 Distribution distance measurement

As introduced in Section 3.1, generative models aim to simulate the data distribution based on samples drawn from the ground-truth distribution. Distribution distance measurement (DDM) usually involves using pre-trained machine learning models, typically classifiers, to evaluate the quality and diversity of sounds generated by a synthesis system. The core of this approach is to assess how well the generated sounds conform to the characteristics of sounds in the original dataset. By categorizing or classifying the generated sounds into predefined categories, the metric measures the closeness of the generated distribution to the original data distribution.

3.7.2.1 Frechet audio distance(FAD)

The *Fréchet Audio Distance* (FAD) [210] is an evaluation metric adapted from the concept used in image generation, known as the Fréchet Inception Distance (FID) [211]. FAD quantifies the quality of synthesized audio by comparing the statistical distribution of features extracted from real and generated sounds. It is particularly adept at capturing the perceptual essence of audio, making it ideal for assessing complex audio outputs such as those in sound effect synthesis.

The calculation of FAD involves the following steps. First, deep audio embeddings were extracted from both the real and synthesized audio samples using a pre-trained neural network such as VGGish [162], Encodec [212], and PANN [88]. Then, these embeddings are usually assumed to follow multivariate Gaussian distributions, and their means and covariances are then computed. Finally, the FAD score is calculated using the Fréchet distance between these two Gaussian distributions:

$$FAD = \sqrt{(\mu_r - \mu_g)^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})} \tag{3.31}$$

where μ_r and μ_g represent the mean vectors of the real and generated embeddings, respectively, and Σ_r and Σ_g are their covariance matrices.

FAD is distinctive from other audio quality metrics like peak signal-to-noise ratio (PSNR) or signal-to-distortion ratio (SDR) as it evaluates the perceptual similarity rather than mere signal fidelity, which is highly relevant for applications where the audio texture and timbre are crucial. This makes it an excellent tool for the synthesis of sound effects, where maintaining the naturalness and authenticity of sounds is paramount. FAD nowadays has become a benchmark metric commonly used for audio synthesis tasks and has been used in a lot of NAS research [84, 100, 99, 125, 170].

3.7.2.2 Maximum Mean Discrepancy (MMD)

The Maximum Mean Discrepancy (MMD) is utilized to compare the statistical properties of two distributions, often between generated and real datasets in audio synthesis. It is defined as:

$$\text{MMD}^2(\mathcal{X}, \mathcal{Y}) = \left\| \frac{1}{m} \sum_{i=1}^{m} \phi(x_i) - \frac{1}{n} \sum_{j=1}^{n} \phi(y_j) \right\|_{\mathcal{H}}^2, \tag{3.32}$$

where $\mathcal{X} = \{x_1, \ldots, x_m\}$ and $\mathcal{Y} = \{y_1, \ldots, y_n\}$ are samples from the two distributions being compared, ϕ represents a feature map to a reproducing kernel Hilbert space $\mathcal{H}$, and $\| \cdot \|_{\mathcal{H}}$ denotes the norm in this space. This metric effectively quantifies the distance between the mean embeddings of the two distributions in $\mathcal{H}$, providing a measure of how similar or different the underlying distributions of the datasets are.

The Maximum Mean Discrepancy (MMD) metric has been utilized in various audio evaluation [99, 213, 84] contexts to measure the similarity between distributions of audio features. In the study [214], MMD is adopted to address limitations of the Fréchet Audio Distance (FAD) by capturing differences in higher-order moments between real and generated audio embeddings, providing a more comprehensive comparison of audio sample distributions. Additionally, [215] employs a multi-kernel variant of MMD as a regularization term in the domain adversarial loss. This approach aims to minimize the discrepancy between source and

target domain distributions, which enhances the performance of speech enhancement models across different acoustic environments.

3.7.2.3 Number of statistically different bins (NDB)

The Number of statistically different bins (NDB) [216] is a metric utilized to assess the diversity of generated samples by examining the coverage of the generated data distribution in comparison to the actual data distribution. NDB is computed by partitioning the data space into a predefined number of bins and counting the number of real and generated samples falling into each bin. The diversity score is derived from the ratio of bins that contain a significantly different number of generated samples compared to real samples.

The Number of Statistically Different Bins (NDB) score is calculated by first partitioning the real dataset into k bins using k-means clustering, where each bin B_i represents a cluster centroid. Generated samples are then allocated to these bins based on the nearest cluster centroid. The discrepancy between generated and real data distributions within each bin is assessed using a Chi-squared test, computed as $\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$, where O_i and E_i denote the observed and expected frequencies of samples in bin i, respectively. The NDB score is the proportion of bins for which the null hypothesis of equal distributions is rejected at a chosen significance level, formally given by NDB Score $= \frac{\text{Number of bins where } H_0 \text{ is rejected}}{k}$. This measure provides insight into the model's ability to replicate the real data distribution across different segments of the feature space, and has been used in checking whether GANs have fall into mode collapse issues.

In neural audio synthesis, NDB has been used to assess the diversity of generated audio and detect mode collapse. In [203, 105], NDB measures how well GANs replicate real audio distributions, with higher scores indicating mode collapse. [217] applies NDB to evaluate hierarchical and cycle-regularized architectures, showing that lower NDB scores correspond to improved sample diversity. These studies highlight NDB as a key metric for quantifying the distributional fidelity of generative audio models.

3.7.3 Subjective Tests

Subjective evaluation methods are designed to assess the perceptual quality of sound synthesis by soliciting human judgments through structured listening tests. These methods are essential for determining how synthesized sounds are perceived by listeners, reflecting subjective experiences and preferences.

One commonly employed technique is the Mean Opinion Score (MOS) [218], where listeners rate the quality of audio samples on a Likert scale from 1 (poor) to 5 (excellent). It has been extensively used in many NAS research [115, 100, 125, 111, 132, 176], where participants provide direct feedback on the sound quality without comparing it to a reference. MOS provides insights into the overall perceptual quality of audio, evaluating aspects such as clarity, fidelity, and naturalness. However, depending on the task, listening tests could be scheduled differently, although usually asked to give an absolute score about the sound quality without direct comparison with a reference track. To this end, MOS is a highly subjective metric, which could lead to a high bias when the participants are sampled from a small distribution.

The MUltiple Stimuli with Hidden Reference and Anchor (MUSHRA) [219] is another method recommended by the ITU for more granular evaluations, particularly useful for assessing audio codecs. Unlike MOS, MUSHRA allows listeners to compare multiple audio samples directly against a reference and an anchor of lower-quality sound. The reference tracks are usually original, unprocessed, and high-quality ground-truth ones. It serves as the benchmark for comparing the other test samples. It is usually expected that participants would rate the reference tracks the highest. The anchor tracks, on the other hand, are usually degraded intentionally to provide lower bounds of quality. Typically, there are two levels of anchors: one that is mildly degraded and another that is heavily degraded. These help in calibrating the listener's perception of quality across a spectrum.

The Perceptual Evaluation of Audio Quality (PEAQ) [220] provides an objective measure that predicts subjective ratings. PEAQ uses mathematical models to approximate how typical listeners would rate the sound quality of audio codecs based on various acoustic properties

of the audio signal. This approach attempts to bridge objective audio measurements with subjective human perception, enabling consistent and replicable evaluations.

While these subjective testing methods are invaluable for gauging listener perception and satisfaction, they are not without challenges. They require meticulous setup and are prone to issues like listener fatigue and the variability of individual listener preferences. Furthermore, designing these tests to yield reliable and valid results demands a significant investment of resources.

3.8 Challenges of NAS for SFX

There are many challenges that are present nowadays for the modelling of sound effects using generative models. In the following sections, I will review the current research gaps in neural audio synthesis for sound design applications.

3.8.1 Limitation of SFX Dataset

In the task of text-to-speech synthesis, phonemes [221] are the smallest discernable elements in a language that contain acoustic information associated with the words. These discrete units are very useful because waveforms could be correlated to texts in a meaningful and self-contained way. In the task of music generation, symbolic representations [222] such as the note of a score or MIDI are commonly used for sound modelling.

For sound effects, unfortunately, there has not been an established symbolic representation of the sounds that could be readily used for sound modelling. Most commercial libraries such as Boom Library [223], BBC sounds [224], and Epidemic Sounds [225], are constructed with categorical labels about the acoustic activities as well as basic metadata such as the format, bit rate, and sampling rate of the sounds. From a modelling perspective, it is usually insufficient to generate new audio samples with varied versions supported merely by categorical labels. [105], for example, generates different categories of urban sounds [226] but lacks a fine-grained control of the variations of sounds.

Furthermore, the recording of SFX itself may include a series of actions, such as the loading and unloading of bullets for a gunshot sound, the bullets falling off to the ground followed by actions of bouncing, etc. Therefore, in order to generate sounds with meaningful controls, one-shot sounds are usually used for convenience. For example, the DCASE Foley dataset [227] and FSDnoisy [228] separate sounds into one shot with less than four seconds each. This ensures a clean and clear snapshot of the sound representation.

However, many SFX datasets such as Audioset[185], Clotho [229], or LAION-Audio-630K [89], contain multiple acoustic events of an object or different sources of sounds unless we are recording Foley sounds in a studio professionally. This can be suitable for the task of text-to-audio(TTA) [69] generation, where descriptive texts [89] are available for each sound recording. However, to be able to synthesize a wide range of timbres, it requires the dataset to be diverse and capable of capturing different aspects of the sounds (such as different recording devices, distance from the sound, intensity of the sound, duration of the sound, physical interaction of the sound, etc).

3.8.2 Lack of Expressive Control

As shown in Section 3.6, class labels are usually the most available information for SFX datasets, but they don't contain enough temporal or semantic information that describes the waveforms. Acoustic features can be extracted from audio deterministically but usually require fine-tuning. Although TTA [69] and V2A [191] enable audio control by changing the semantics of another modality, texts are limited by their temporal alignments with audio in providing fine-grained control in the time domain, while videos are limited by their expressivity in controlling the semantics/timbres of sounds.

Many modern NAS models focus primarily on the quality of generated sounds, given explicit conditioning, especially for text-to-audio synthesis. In many circumstances such as Foley and game audio, expressive temporal and timbre control over the SFX are crucial. Therefore, there is currently a research gap in providing expressive audio controls in the domain of generative sound effects modelling.

3.8.3 Lack of Appropriate Evaluation Techniques

A good many references mentioned in Section 7.4 are based on statistical measurement of similarity between generated samples versus reference sounds. Such metrics are under the assumption that better similarity to the original training dataset represents the higher performance of the generative model. However, being able to synthesize similar audio samples to the training dataset does not always mean the model captures all the variances in the timbres of the sounds. A critical aspect of SFX modelling, the degree of variation in timbre which describes the unique sonic characteristics of certain sounds, is usually not captured in such metrics.

As timbre itself is a very subjective description of sound, it can be challenging to quantify the subtle differences. For example, the sound of a footstep on gravel is very different from a footstep on concrete floors. Such timbre differences are not easily describable with acoustic features but are highly important auditory cues that help us understand the actions or materials of the sound. Therefore, being able to differentiate the timbre differences among different categories of sounds is beneficial for the task of sound modelling. More research is required on how to evaluate the timbre variations output by generative models.

3.9 Chapter Summary

In this chapter, I reviewed the fundamentals of machine learning, deep learning, and generative models. I have also reviewed key techniques used for generative sound modelling, including how different audio representations are used in the research of neural audio synthesis, how different loss functions could impact the result of training NAS models, as well as how different evaluation techniques are employed in NAS research. With a detailed description about each generative model, we have seen how they are applied in neural audio synthesis tasks.

Nowadays, most related research puts an emphasis on novel conditioning techniques, aiming to control the sound generation process more intuitively and interpretably. However, there are

several reasons why the neural network approaches have not become a prevalent synthesis method for sound effects as those in natural speech synthesis, such as the lack of properly constructed and publicly available dataset, the lack of meaningful and universally acknowledged symbolic guidance factors such as texts for speech and musical notations for music generation, the lack of and the immense categories of unique sound effects that make it hard to generalize and synthesize.

In the following sections, I will explore strategies that address the aforementioned problems. Specifically, given the limitation of SFX datasets, all the work carried out in this research utilizes small audio datasets (less than a few hundred files for a category with a maximum of four seconds of each sound) without descriptive texts.

CHAPTER 4

Conditional sound effects generation with regularized WGAN

4.1 Introduction

As illustrated in our literature review 3, the majority of research in neural audio synthesis (NAS) targets speech or music, whereas general sound effects such as environmental sounds or Foley sounds have received less attention. In NAS research, many works have focused on modelling time-frequency representations, which have multiple benefits mentioned in Section 3.4.2, such as dimensionality reduction, noise robustness, model training efficiency, and compatibility with visual models. However, as magnitude spectrograms lose phase information, reconstructing them into time-domain waveforms is usually a challenging topic. Therefore, we aim to address the issue presented in time-frequency modelling and audio reconstruction with vocoders [17].

In this chapter, I aim to study time-frequency representation modelling and focus on the research question of improving the quality and diversity of neural audio synthesized sound effects. To this end, I incorporate the phase estimation [142, 17] process into training to minimize reconstruction error. Specifically, we employ a GAN architecture to generate log-magnitude spectrograms conditioned on discrete categories of sound effects. To properly study whether this model is capable of synthesizing different categories of SFX, I proposed a categorization of sound suited for the task of SFX modelling. I have created a custom high-quality dataset consisting of five different types of SFX that fall under this categorization. I have demonstrated that this approach improves the synthesis quality by overcoming some of the problems associated with waveform reconstruction based on phase estimation. I then

also show the proposed method achieves a high Mean-Opinion-Score (MOS) similar to the original training sounds.

4.2 Background

As outlined in the introduction of the thesis 1, sound effects often play an important role in digital media or entertainment, such as films or video games, by enhancing the immersive experience [21]. In most cases, they are created from recorded sounds or synthesis engines[15]. A sound designer would then utilize established sound libraries to search for the most appropriate effects that best suit the scenes they are working on. Often, e.g., in a particular game setting, the set of sound categories that is required is small, while the variation within the set of sound categories is important.

The number of variations of a given sound type within a sound library is often limited. Using the same audio sample repetitively, e.g. in a game setting, often deteriorates the quality of the perceptual experience. Therefore it is desirable to have many variations of a particular sound type for developing scenes and actions [230, 231]. This happens to be an area of strength for generative models as they are capable of generating high-quality and diverse sounds that are conditioned on different classes[105, 83]. It is even more so in the domain of Foley sounds for moving images, where previous research [101, 191, 232, 198] have explored sound generation by conditioning on visual signals.

In neural audio synthesis, there are two main approaches for modelling sound: modelling the raw waveforms directly [115, 100, 68] or modelling the extracted time-frequency (TF) representations [105, 141, 69]. Since audio waveforms can almost completely describe the sounds, it is intuitive to model audio signals directly. Barahona-Ríos and Collins [100] and Marco et al. [99] have explored using conditional WaveGAN and HifiGAN to generate short knocking sounds and footsteps, showing promising results on capturing the fine acoustic details of various kinds given a small and refined dataset. However, both of these research works have only trained on a particular type of sound, namely, impacts comprising single

impulses in the waveform. Exactly how well different classes of sounds could be modelled with a conditional waveform-based GAN remains to be explored in detail.

On the other hand, TF representations such as magnitude spectrograms or mel-spectrograms offer many benefits as they provide visual representations that are arguably more interpretable by humans or machines. They are also able to capture and generate some long-range representations utilizing the power of convolutional neural networks (CNN). Often the time and frequency axes are organised to produce square images using short-time Fourier transforms (STFTs).

The TifGAN [141] model employs log-manitude spectrograms with a non-iterative phase estimation algorithm PGHI (Phase Gradient Heap Integration) [142] and achieves high quality sound reconstruction results. Gupta et al.[233] have further explored the generation performance of a progressive GAN model using audio textures such as noise bursts and chirps. They have intriguingly found that the result of using the PGHI method to reconstruct the phase from a magnitude spectrogram outperforms models that are trained using a two-channel representation comprising both the magnitude spectrogram and instantaneous frequency (IF) [234].

It was known that since phase information is abandoned when training on magnitude spectrograms, the reconstruction from such representations is not always perfect even with phase estimation algorithms such as PGHI[142]. This has led many to explore using a neural vocoder [77, 126, 96] model to predict the audio waveform from the magnitude spectrogram. Nonetheless, this involves training an additional model which increases computational resources and time.

4.2.1 Categorizing Sound Effects

From a sound modelling perspective, a clear and insightful categorization of various sound effects is necessary. However, rather than differentiating sounds by their recording environments [235], or by the kinds of the sound producing objects [236], or even by the types of

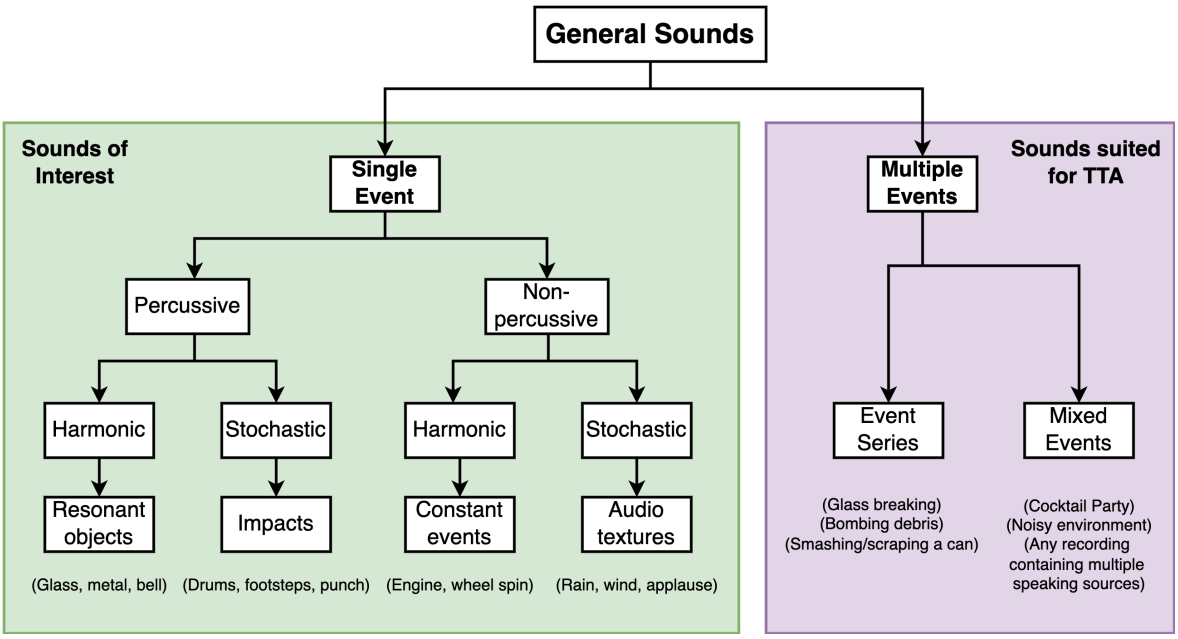


FIGURE 4.1. A categorization of different types of sound effects.

physical interactions that produce the sound [237], in generative sound modelling, we are more interested in their time dynamics and spectral contents. This is because generative models learn from audio samples or their representations directly. For example, for harmonically rich sounds such as violins, it is natural to model their spectrograms with density estimation methods. For impulsive signals such as explosions or fireworks, details in amplitude dynamics are crucial, which makes autoregressive models a better choice.

For the task of generative sound modelling as a statistical model that draws on samples, the audio recordings with some labels are the only accessible pieces of information in most cases. Therefore, we try to categorize the recorded audio samples instead of categorizing them by the audio systems that produce such sounds. We focus mainly on the statistical differences between the sounds' time domain and spectral domain information for the task of generative sound modelling.

As shown in the figure 4.1, based on the time-domain and spectral domain representations, we have categorized sounds into percussive or non-percussive, harmonic or stochastic, single or multiple events. Sounds in nature are complex. Therefore from a modelling perspective, a

consensus is necessary on the smallest discernable elements treated as single acoustic events, where each event is independent and is not interrupted by other events in a recording.

Impulsive sounds are ubiquitous. Such sounds usually have very short attack and sustained time and can be commonly seen when two objects come into contact or collide with each other. Depending on whether the waveform has harmonic distribution, many of the musical instruments could also be put into this class, such as a single note from guitar plucking or piano striking. It is worth noting that samplewise, a short bird chirping or dog barking could also be considered impulsive if it's short enough. This category is typically the easiest to model, as the duration of the sounds is usually clear from the short waveform and most of them could be easily contained within a fixed length of window. Density estimation generative models such as GAN [79] and VAE [18] could be a great choice for modelling such sounds, as they provide a great way for interpolating in latent space, which allows us to change the sonic characteristics of the sounds.

Non-percussive sounds can be seen quite commonly in environmental sounds and ambience. These sounds have stable and predictable frequency distributions across time. Many sounds in this category are completely stochastic, such as a recording of rain and air-conditioning. There are sounds, however, with organized patterns of spectrum change, such as car engine accelerating, that could be considered harmonic while non-percussive. Non-percussive sounds are often analogous to audio textures[2], which are termed as sounds having constant long-term characteristics. Audio texture synthesis itself has been a specific research field, and many attempts to apply generative models to audio textures have been made in the past years. It has been shown that many of them approach the task by borrowing ideas from image processing and by utilizing convolutional neural networks on 2D spectral representations of the sounds [166][238][139].

Mixed events are usually sounds having a series of audio events, resulting in drastic and irregular changes in the spectra or waveform. For example, crunching a pop can can create a series of audio events over a short duration of time that are hard to give a clear label over their associated time period. Such acoustic events could vary greatly across time and exhibit

stochastic patterns. Mixed events entail sound activities that involve multiple sound sources. For example, a recording of a cocktail party could be considered a mixed event as it involves multiple sounds having irregular patterns over time. For general sound effects, this category includes most sounds that may contain a series of physical activities or acoustic events, such as the collision and crash of multiple cars. As mixed events are more complex and contain multiple sound activities, they are inherently harder to model without appropriate or adequate descriptive labels. Most of the existing text-to-audio (TTA) research focus on this type of SFX, where a correlation of the semantics and dynamics of texts and audio is to be learned.

It is worth noting that this categorization is based on the perceptual differences of the sounds, where single acoustic events could be stacked to be treated as a different group. For example, depending on the context of the recording, raindrops could be considered impulsive if it's just a single drop, while intermittent rains could sometimes be classified as non-stationary sounds. Therefore, many sound recordings could actually be from different categories, even if they have the same descriptions. The purpose of this categorization is to suit for the task of data-driven sound synthesis.

In this thesis, we are more interested in the task of single acoustic event modelling. Therefore, we focus on the four sub-categories of SFX. In the following sections, we select sound effects that fall into such categories.

4.3 Experimentation Setup

To set up the experiments, we select a baseline GAN model to perform conditional sound generation. We carefully chose five categories of sound effects and then performed the same pre-processing. We then demonstrate our proposed method.

4.3.1 Baseline Model

We start with a baseline model which is based on a similar architecture as TifGAN[141] (see Fig 4.2). The baseline model is trained with Wasserstein GAN loss including gradient norm penalty [114]. We utilize a conditioning method that takes a one-hot vector indicating one of five different sound classes. We utilize a conditioning method that takes a one-hot vector indicating one of five different sound classes. In this way, the model is able to synthesize any one of five different types of sounds. The model is trained using log-magnitude spectrograms with size (256,128). For the purposes of comparison, all models use the same hyper-parameters as the baseline model (e.g., 100,000 iterations, a batch size of 16, 1e-4 learning rate).

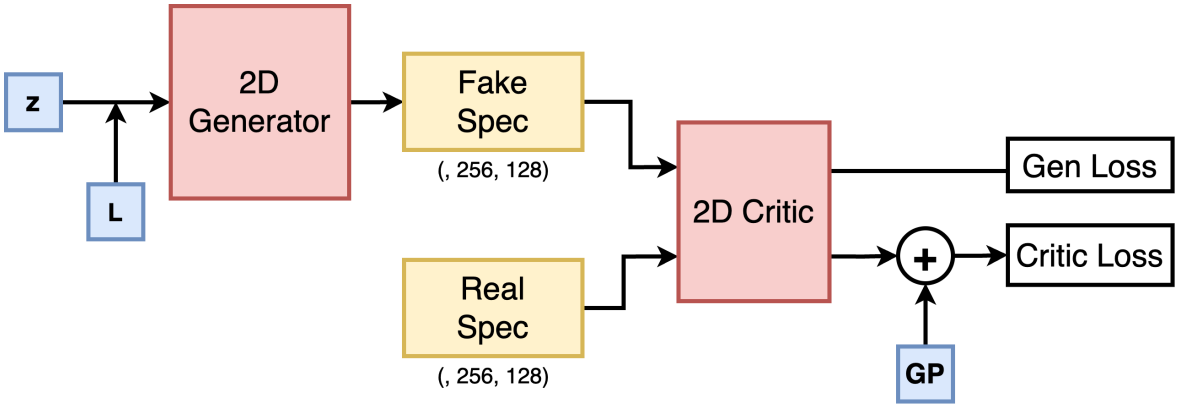


FIGURE 4.2. The baseline model is shown with Wasserstein loss formulation and gradient norm penalty training scheme.

4.3.2 Dataset

Compared with speech and music, which are usually normative and structured, sound effects are much more diverse, and recordings are usually susceptible to background noises. Common sound effects libraries such as Freesound.org[1] or Audioset[2] can contain many unwanted sounds in a recording and the quality can vary greatly depending on the recording device, environment, and post-processing. Therefore we created our dataset meticulously by selecting high-quality

[1] https://freesound.org/
[2] http://research.google.com/audioset/

Sounds	Description	Amount
Footstep	Transient impacts	1170
Gunshots	Transients with long decay	600
Rain	Long-sustaining non-pitched textures	1085
Engine	Long-sustaining pitched textures	1170
Birds	Transients with time-varying high pitches	1420

TABLE 4.1. Five types of sounds from real-world recordings. We gathered the sounds from BBC library [224], Soundideas [239] and Boom Library [223]. For experimentation, we curated the dataset so that they are one-shot contents for the footsteps and impacts category.

sound effects from commercial libraries including BBC sound effect[3], Soundideas[4] and Boom Library[5]. Given that these libraries require licenses for commercial use or release, we have sought approval from these libraries for academic purposes prior to the usage of such datasets.

For each sound category, we selected the sounds that capture slightly different sonic characteristics, such as different contact surfaces for a footstep or different rain environments recorded at different locations. By carefully and purposely selecting sounds with varying timbral characteristics within each class, we increase the within-class sound diversity. For each category of sound, there are roughly one thousand samples. Although this amount of data is much smaller than what is commonly required for most speech synthesis models, we found it sufficient for our experiments. This amount of training data is similar to previous works [100]. For the sake of comparison, we restrict all of the sound samples to be one second of audio, resulting in 16000 samples at a 16 kHz sample rate. Each recording sample contains a single example for the given sound category, e.g. a single shot for footsteps and gunshots, with clean onsets and offsets. The given formulation of the dataset simplifies the evaluation of the diversity of the sound synthesis.

4.3.3 Data Processing

The input audio data is zero-padded to a length of 16,384 so that the spectrogram image data are created with a shape of (256,128). To extract the spectrogram, we compute the STFT

[3] https://sound-effects.bbcrewind.co.uk/
[4] https://www.sound-ideas.com/
[5] https://www.boomlibrary.com/

with a Hanning window using the Tensorflow I/O implementation[6]. For the STFT, the size of the FFT is 254, the window size is 256 and the hop length is 64. Similar to [141], we normalize the magnitude of the STFT data to fall within a range of (0,1), take the logarithm to clip their log magnitudes below e^{-10}, then we scale the result down by dividing 5 and shift the output range to [-1,1]. During the generation stage, we apply the Griffin Lim algorithm from the Tensorflow I/O library[7] with the same windowing settings as previously described. The reason we chose Griffin Lim for phase estimation is because it is one of the most widely used methods for phase estimation. Another reason is that this method has been written as a differentiable algorithm in tensorflowio library. Therefore, we could take advantage of this to integrate it into the training process.

4.3.4 Improved Audio Reconstruction

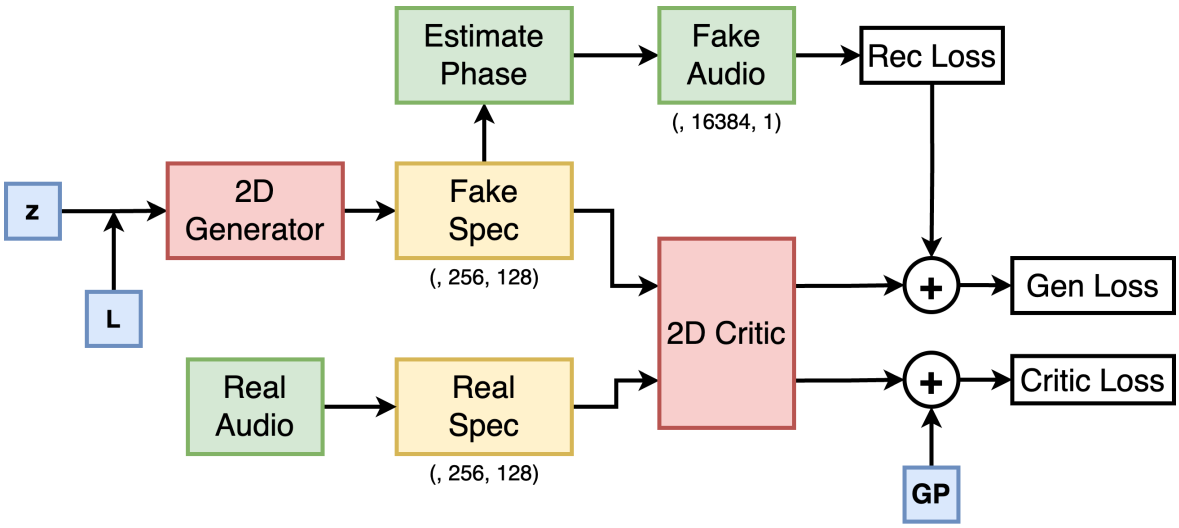


FIGURE 4.3. Our proposed architecture. It is based on a conditional WGAN model with custom reconstruction loss added to the generator loss. This regularizes the generated waveforms to follow the distribution of the training audio waveforms.

In order to improve the audio reconstruction process, we borrow an approach used in neural vocoders and add an additional loss term to our generator. Using the same baseline model

[6] https://www.tensorflow.org/io/api_docs/python/tfio/audio/spectrogram
[7] https://www.tensorflow.org/io/api_docs/python/tfio/audio/inverse_spectrogram

during training, we now compare the synthesized audio with a random audio sample of the same sound class chosen from the dataset. We use a Mean-Square-Error (MSE) loss function, L_{rec}, calculated as shown below:

$$L_{rec} = \frac{1}{n}\sum_{i=1}^{n}(R_i - F_i)^2\,, \tag{4.1}$$

where n represents the batch size, R_i is an audio sample of the same category randomly chosen from the dataset, and F_i is the audio waveform reconstructed from the magnitude spectrum produced by the generator (we use the Tensorflow I/O spectrogram inversion algorithm[8]). We multiply the MSE loss with a hyperparameter weight value, W, to prevent it from overtaking the generator loss. Finally we add the original generator loss L_g such that the final generator loss L_{gen} becomes:

$$L_{gen} = L_{rec} * W + L_g \tag{4.2}$$

It is worth emphasizing that L_g calculates the loss based on the generated spectrograms, while L_{rec} computes the loss according to the reconstructed audio. The goal of adding the reconstruction loss to the generator is to encourage the generator to synthesize TF representations whose inverted waveforms are similar to the training data. Previous research indicates that adding a reconstruction loss term to GANs reduces diversity [240]. In order to mitigate this side effect, we apply a small scaling factor as a weighting parameter to the loss, $W = 1e^{-4}$. While it is common to train the discriminator using more steps than the generator, we found instead that training the generator more times than the discriminator gave improved results[9].

4.4 Evaluation

Evaluating audio generation can be a difficult task because there is no ground truth waveform to compare with. Additionally, since hearing perception is subjective, the same waveforms

[8] https://www.tensorflow.org/io/api_docs/python/tfio/audio/inverse_spectrogram

[9] Our implementation is available online at https://github.com/Reinliu/CWGAN-SFX

might be perceived differently among different participants. Therefore, we resort to both objective metrics as well as human listening tests to evaluate the synthesis quality/realness and diversity. For evaluation, we decided to generate 1000 samples, a similar amount to our dataset for each category of sound.

4.4.1 Frechet Audio Distance

The Frechet Audio Distance (FAD)[210] is a reference-free evaluation metric for generated audio quality that measures the similarity between two audio signals by comparing their feature representations through an embedding layer of a pre-trained VGGish model. It has been quite popular over recent years in the neural audio synthesis field and has been used in research such as DarkGAN[84] and Diffwave[125]. Many studies have shown that this metric correlates well with human listeners' preferences [99] and is robust to many audio sources [234, 233]. The smaller the FAD score, the smaller the distance between the original audio and the generated audio and the better the performance of the generative model.

4.4.2 Number of Statistically Different Bins

Number of Statistically Different Bins (NDB) [216] is an algorithm that has been proposed to evaluate the diversity of generative models [105] and to provide guidance on whether a mode collapse has occurred. It applies K-means clustering of the training data into K-different 'bins' or diversity modes with Voronoi decomposition and then assigns each training and generated data to the closest bin based on an L2 distance. The NDB score is then calculated as the number of bins in a histogram that are considered to be significantly different from each other based on a two-sample binomial test between the training dataset and the generated dataset.

As mentioned in Section3.4, compared to audio waveforms, spectrograms tend to reveal more structural information about the sound. Therefore we compute the NDB score based on the magnitude spectrogram images of the training and generated audio, similar to the approach by Liu et al. [105]. To obtain the spectrogram, we used the number of FFTs as 1024, a window size of 1024 and a hop length of 32, which results in a 512 by 512 spectrogram image. The

We use 20 K-means bins for each sound class. The choice of the number of bins per class is actually dependent on the variations of the datasets. However, in our case, 20 bins should be sufficient for each single category, as they contain similar waveforms without too many variations. This setting is the same as Liu et al. [105]. In addition to the NDB/k score for each sound category, we compute an all-class NDB score by mixing all five categories of sounds together. In this case, we set the number of bins to be 100 $(20 * 5)$.

4.4.3 Human Listening Tests

We appreciate that objective metrics alone are not sufficient for neural audio synthesis tasks [203] as the quality or realness of a sound is usually dependent on the human listeners' hearing and preference. Therefore, we report a Mean-Opinion-Score (MOS) obtained from eight participant listeners. The participants consist of 3 females and 5 males with ages ranging from 26 to 50 with no previous background in ear training. Each participant was presented with the same pair of headphones and was asked to rate their preference for realness/quality for five types of sounds. The place of the experiment can be viewed in Figure .7.

There were four experiment conditions for each of the five classes of sound according to whether the audio was selected from either: the training audio from the dataset, the sounds generated from the baseline model, the sounds generated from the new model with the additional loss term, and sounds generated from a WaveGAN used as a control condition. For each sound class(footsteps, rain, etc), we randomly chose four clips of one-second sounds from the four sources mentioned above. Therefore, in total, there were 80 sounds that a listener needed to rate. For each listening test, the listener was presented one sound at a time and was told the category of the sound being played, but was asked to report their preference based on how realistic they perceived the sound to be (e.g., whether it appeared to be a real/natural audio source or a synthetic audio source). A preference scale from one to five was used: 1 (bad/synthetic) to 5 (excellent/natural). The example sounds could be accessed at[10].

[10] https://github.com/Reinliu/CWGAN-SFX/tree/main/Sound

4.5 Results

TABLE 4.2. Evaluation results comparing the baseline WGAN model, WaveGAN, and our approach based on three metrics: FAD (lower is better), MOS (higher is better), and NDB (lower indicates more diversity).

Model	**Footstep**			**Bird**			**Guns**			**Rain**			**Engine**			**All**		
	FAD	MOS	NDB	FAD	MOS	NDB	FAD	MOS	NDB	FAD	MOS	NDB	FAD	MOS	NDB	FAD	MOS	NDB
Original	-	**4.56**	-	-	**4.22**	-	-	**4.28**	-	-	**4.19**	-	-	**4.59**	-	-	**4.37**	-
Baseline	2.14	3.94	0.1	19.43	3.47	**0.25**	**4.65**	3.97	**0.1**	10.30	3.44	**0.15**	13.84	4.16	**0.2**	10.07	3.78	**0.16**
Proposed Method	**2.07**	4.22	**0.05**	**2.41**	3.91	0.3	5.05	4.16	0.15	**2.64**	4.03	0.15	**12.21**	4.47	0.25	**4.88**	4.16	0.18
WaveGAN	20.27	3.66	0.1	33.17	2.75	0.35	6.22	3.56	0.15	11.27	2.72	0.2	27.48	2.91	0.2	19.68	3.12	0.20

Following our evaluation methods, we show the corresponding results in terms of generation quality and diversity. The results were reported as FAD, MOS, and NDB scores.

4.5.1 Generation Quality

4.5.1.1 Frechet Audio Distance(FAD)

When considering the results across sound categories, please keep in mind that a single conditional WGAN produces all five classes of sound. We observe an overall improvement in audio quality using the proposed method. With the bird and rain sound classes, we see large improvement. Our interpretation for this finding is that these two classes of sounds are arguably more complicated than the other categories. The bird samples consist of several different species of birds chirping and the rain dataset contains recordings from many different environments. Large variations within a particular sound class likely make the job of the generator more difficult. The reconstruction loss likely assists with regularizing across these larger variations. We also observe that the conditional WaveGAN performs comparatively worse across all sound categories. We interpret this result as indicating that conditioning on different sound categories, rather than variations within a given sound class, was somehow more difficult for the WaveGAN which is based solely on the audio waveform.

4.5.1.2 Mean Opinion Score(MOS)

The Mean-Opinion-Score (MOS) results are shown in Table 4.2. The MOS values provide a subjective rating of the plausibility of the sounds. The results indicate that the listening

test results correspond well with the FAD results. For the bird and rain sound classes, we again observe relatively larger improvements with the proposed method. The results similarly indicate that the WaveGAN performs worse than the spectrogram-based methods, with an average of 3.12, 3.78 and 4.16, respectively, for the WaveGAN, baseline, and proposed methods. It is worth noting that the proposed method achieves a mean score that is similar to the samples from the audio dataset (labelled Original in the table).

4.5.2 Generation Diversity

The results for the objective diversity measure are shown in Tables 4.2. Notice that the NDB scores for each column were calculated when initializing each category with 20 bins for the clustering, while the NDB of all column is calculated by initializing with 100 bins for the K-means clustering. The results indicate that the baseline network better captures the diversity of the training dataset than the proposed method for all sound classes, except the footsteps sound class. This is generally to be expected because of the additional loss term in the generator. The added regularization could be considered as a reconstruction loss, where generated sounds are evaluated directly with original audio in the time domain. This could hamper the generation of new samples, as the generator was trained to synthesize similar ones to the original audio.

4.6 Chapter Summary

In this research I have studied the performance of NAS models for five classes of sound effects. The result and codes is available at our accompanying website [11]. I have shown that for this particular type of conditional generation across different sound classes, the WGAN model trained on log-magnitude spectrograms achieves better performance than the WaveGAN model. We proposed a regularization method based on the waveform reconstruction error to help improve the performance of the GAN's generator. The performance results indicate that there is value in this approach with regards to achieving substantial improvement in audio

[11] https://github.com/Reinliu/CWGAN-SFX

quality while suffering a small loss in diversity. However, to better encourage higher sample variations, future work could focus on improving the regularization loss term. In this research, the mean of generated audio in a batch is regularized to be similar to randomly chosen real audio in the same amount of the original batch of audio. However, this does not guarantee that the audio from each statistical bin to be selected equally throughout the training. Therefore in the following chapter, I aim to improve the diversity loss by studying different training and optimization techniques using similar NAS architecture for sound synthesis.

CHAPTER 5

Improving sample diversity with reconstruction loss in GAN

5.1 Introduction

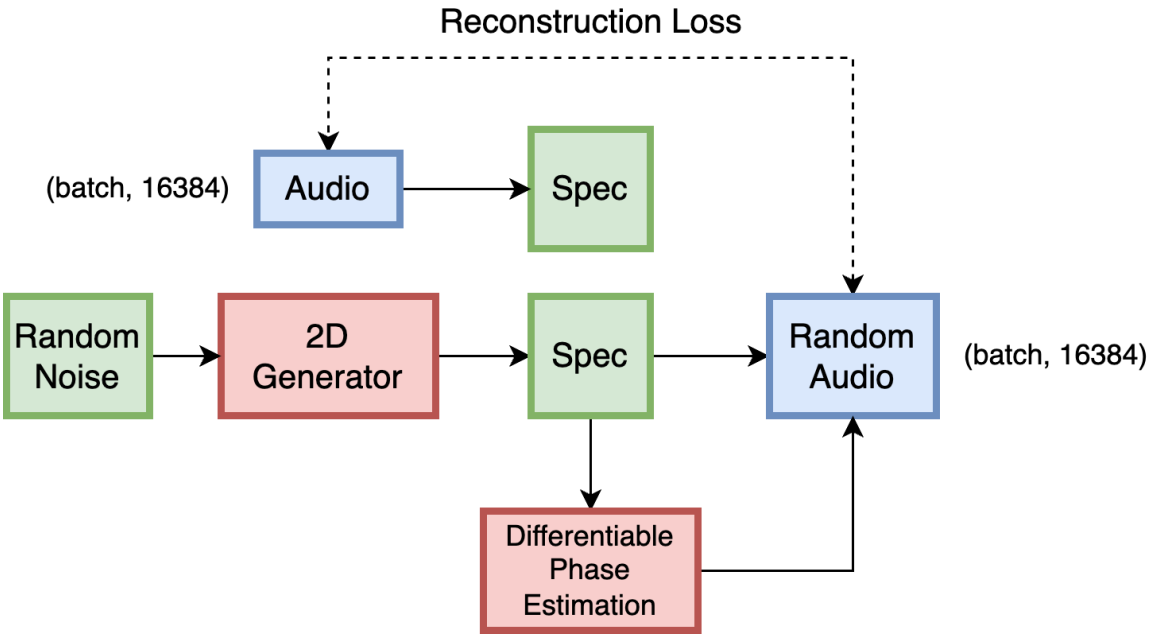


FIGURE 5.1. Issue with using reconstruction loss in GAN.

As shown in Figure 5.1, Chapter 4 leaves us an issue when using reconstruction loss under the GAN architecture setup. When GAN [39] models generate new instances, they usually randomly sample a vector from standard Gaussian distribution as input. Different from Variational Autoencoders (VAE) which reconstructs the original data, the generated contents from GAN, whether spectrogram images or digital audio samples, are actually randomly generated without explicit regularization. Our reconstruction loss 4.1 compares the mean of the generated audio waveforms relative to the average of the original audio waveforms.

This process could potentially hamper the generation diversity, as the generated ones are not explicitly paired for comparison with the original audio waveforms. Therefore in this research, I aim to study this diversity reduction and try to improve the balance of generation quality and diversity.

5.2 Background

Synthesizing sound effects with appropriate controls has been notoriously difficult. Traditional physical modelling synthesis algorithms require extensive knowledge about the materials, structures, and acoustics of the modelled sound event. In recent years, generative models have demonstrated their success in speech and music synthesis by relying on deep neural networks to approximate the probability density functions of the target audio distribution. Further, *text-to-sound* (TTS) generative networks such as AudioLDM[69] and AudioGen[108] have shown great semantics and high quality in sound effects generation by modelling time-frequency representations of sound such as Mel spectrograms.

However, sound and text are essentially different in that texts contain no temporal information as audio. Typically, fixed sequences or frames of raw waveform or spectrograms containing an entire sequence of sound events are modelled. Therefore, it is natively difficult to control or vary the waveform sensibly along the sequences. Although TTS excels in the quality of audio synthesis and offers a way of controlling sound generation via text, it is still limited by the expressivity of descriptive words especially in fine-detailed controls over the timbre of generated sounds.

On the other hand, generative adversarial networks (GAN)[79] are widely known for their generative performance in image generation[116, 241]. In the area of audio synthesis, GANs have been utilized to model audio waveforms directly[115], or time-frequency representations[141, 242] followed by a vocoder algorithm such as Griffin Lim[17] to convert the representations back to audio. Another popular GAN music synthesizer[12] achieves more coherent waveform generation by modelling "instantaneous frequency", which is the amount of change in unwrapped phase per frame.

However, even though this architecture performs especially well in periodic waveforms, such as musical timbres with closely-spaced harmonics, it remains difficult to generate coherent waveforms with fewer harmonics and abundant noise, such as speech or sound effects. To model general sound effects, many sophisticated text-to-sound models relying on large language models leverage pre-trained neural vocoders [95, 96, 77, 125, 126] to convert the mel-spectrograms into waveforms. TTS models are therefore only required to focus on generating mel-spectrograms from latent representations[69, 108], which is far more efficient than learning to generate long-range waveforms directory.

Although GANs possess the power of high-quality audio synthesis and parallel processing, the size of the generated contents is usually fixed, depending on the specific setup. Considering that sound effects are typically composed of highly variable acoustic activities with arbitrary length, it becomes crucial for a generative model to be able to generate variable-length contents. Furthermore, convolutional neural networks, especially transposed convolutional operations used frequently in GAN upsampling layers, could give rise to upsampling artifacts (tonal artifacts or filtering artifacts)[243]. In addition, GANs are also widely known for mode collapse issues due to the adversarial nature of such networks. If the discriminator performs well too quickly, it can cause the generator to converge to a narrow set of outputs that it finds most convincing, rather than exploring the full range of the data distribution.

In this research, I propose a neural audio synthesis model capable of controlling sound timbres while enabling the generation of arbitrary lengths of sounds using generative adversarial networks (GAN) [79]. Instead of generating sounds in the waveform domain, I model Mel spectrograms by leveraging large pre-trained neural vocoders for Mel spectrogram-to-audio conversion. To solve the diversity reduction issue covered in the previous chapter, I will compare three different training variations of WGAN models and try to pinpoint the best strategy for training GAN with audio samples. Our model is shown to be capable of synthesizing variable-length spectral contents, making it suitable for controllable sound effects generation.

5.3 Method

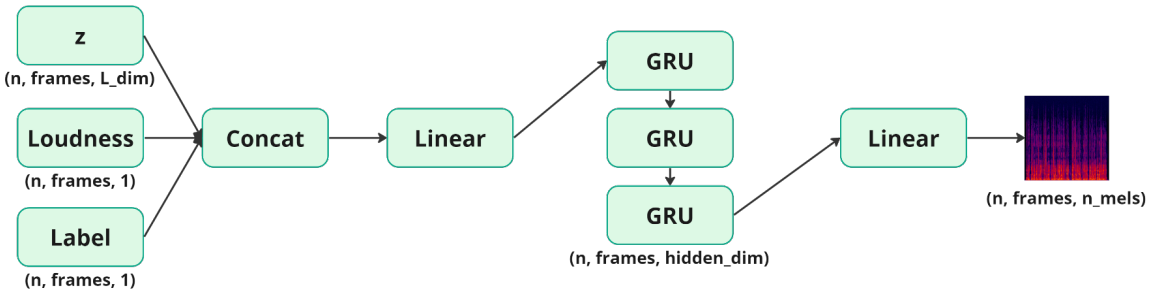


FIGURE 5.2. Generator architecture

5.3.1 Generator architecture

To be able to generate variable-length time-frequency features, I rely on recursive neural networks (RNNs) for the generator architecture to model Mel spectrograms, which could then be converted back to audio using a pre-trained vocoder[95]. More specifically, I employed a three-layer gated-recurrent unit (GRU) architecture with 512 hidden units as shown in5.2.

To provide meaningful control vectors for sound generation, in addition to the latent variable z, I further concatenate z with a class label embedding C and loudness vector L. The class embedding represents the category of a sound (eg. footsteps, gunshots, motor engines, etc) which guides the generator on which kind of sound to generate each time.

The loudness vector is an A-weighted loudness curve which we have extracted from each audio clip individually before training. It functions as an envelope indicator to guide how much energy each frame should contain for the generator. I then performed the same pre-processing as in Engel et al. (2019) [13] to obtain the A-weighted loudness envelope vector extracted from each training audio clip. The concatenated inputs are processed by a linear layer for our recurrent units to learn, and then transformed by another linear layer to output the Mel-spectrograms.

Note that the dimension of the output Mel spectrogram depends on the input loudness vector, where the length or number of frames of the output spectrogram is set as equal to the length of

L. The number of Mel bins depends on the specific configuration of the pre-trained vocoder, which, in our case, is 64. In this way, we obtain a variable-length Mel spectrogram generator conditioned on different sound effect classes as well as loudness envelopes.

Different from a vanilla GAN generator, the latent vector in our model only encodes the subtle timbral changes across different sounds, while the amplitude of the generated sound is then expected to be controlled independently by the guiding loudness vector L. By varying the latent variable without changing the loudness or label vectors, we expect slightly different versions of the target sound effects in the same rhythm.

5.3.2 Proposed model

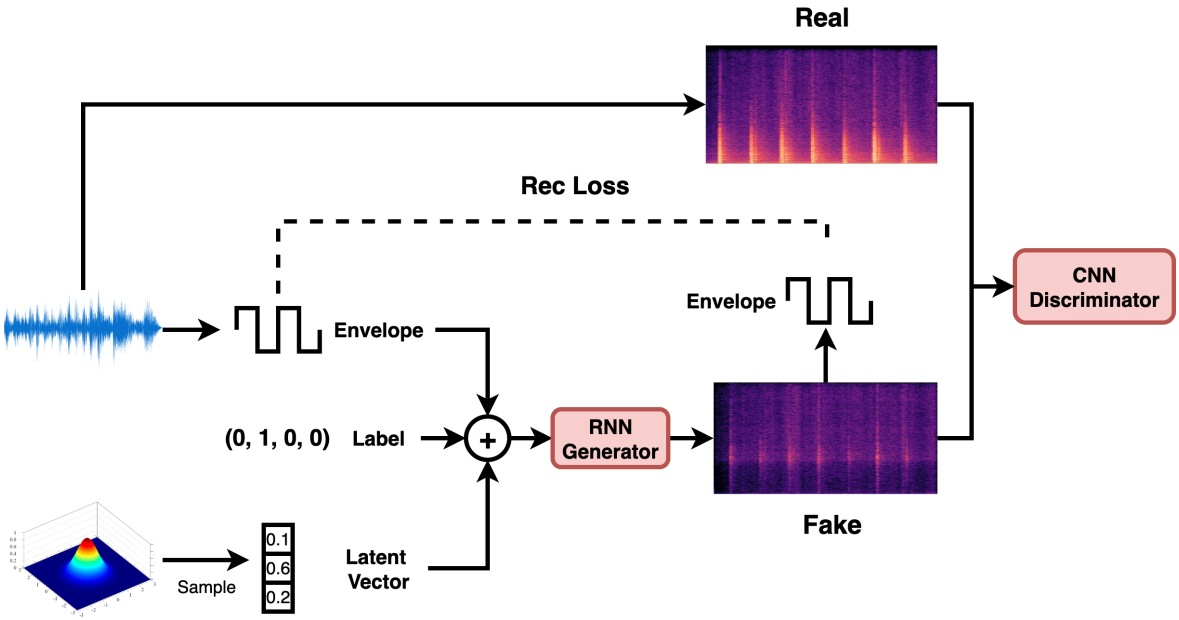


FIGURE 5.3. Proposed model architecture

I denote our proposed model STGAN: Spectro-temporal Generative Adversarial Network as shown in Fig. 8.3). It is based on a modified WGAN-GP model architecture[114] tasked to generate two-dimensional Mel-spectrograms. The generator relies on an RNN to generate locally-coherent mel-spectrograms in arbitrary length as introduced in the above section. The discriminator network then discriminates the generated mel-spectrogram and the real mel-spectrogram extracted from the training dataset.

I have utilized the same configuration for the convolutional operations as the original WGAN [113] configuration to learn the spatial patterns of the time-frequency representation: convolutional 2D layers + LeakyRELU activation. This configuration retains the processing powers from convolutional networks and maintains high training speed. Furthermore, as the generator is conditioned on class labels, I concatenate label embeddings onto the input layer of our discriminator.

5.3.3 Loss function

The loss function in our proposed method includes a combination of discriminator loss and generator loss. The discriminator loss remains the same with WGAN-GP as defined:

$$L_D = \mathbb{E}_{\hat{\mathbf{x}}\sim\mathbb{P}_g}\left[D\left(\tilde{\mathbf{x}}\right)\right] - \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}\left[D\left(\mathbf{x}\right)\right] + \lambda\mathbb{E}_{\hat{\mathbf{x}}\sim\mathbb{P}_{\hat{\mathbf{x}}}}\left[\left(||\nabla_{\tilde{\mathbf{x}}}D\left(\tilde{\mathbf{x}}\right)||_2 - 1\right)^2\right], \tag{5.1}$$

where $\hat{\mathbf{x}}$ are the samples generated by the generator, $\mathbb{P}_g$ is the model distribution, i.e., the distribution of the samples generated by the generator, while $\mathbb{P}_r$ is the distribution from real samples, $D\left(\tilde{\mathbf{x}}\right)$ represents the discriminator's output for the generated samples, and $\mathbb{P}_{\hat{\mathbf{x}}}$ calculates the average output of the discriminator over the generated samples.

To encourage our generator to learn all of the modes from the training dataset and to better guide our generator for convergence, I introduced a reconstruction loss to penalize the generated spectrograms which have distinct loudness envelopes. Our reconstruction loss is a Mean-square error loss (MSE loss) calculated on the loudness difference between generated Mel-spectrograms and those extracted from training audio datasets:

$$L_{mse} = \frac{1}{n}\sum_{i=1}^{n}(l_f - \hat{l}_r)^2, \tag{5.2}$$

where l_f denotes the loudness envelope calculated from the generated Mel spectrograms and $\hat{l}_r$ is the loudness envelope extracted from the real audio. Our generator loss is then defined as the combination of the vanilla generator loss and the MSE loss:

$$L_G = -\mathbb{E}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}\left[D\left(\tilde{\mathbf{x}}\right)\right] + W_{mse} \times L_{mse}\,, \tag{5.3}$$

where W_{mse} is a weighting factor multiplied by the reconstruction loss to stabilize the training. In our case, we leave it as 1.

5.3.4 Inference

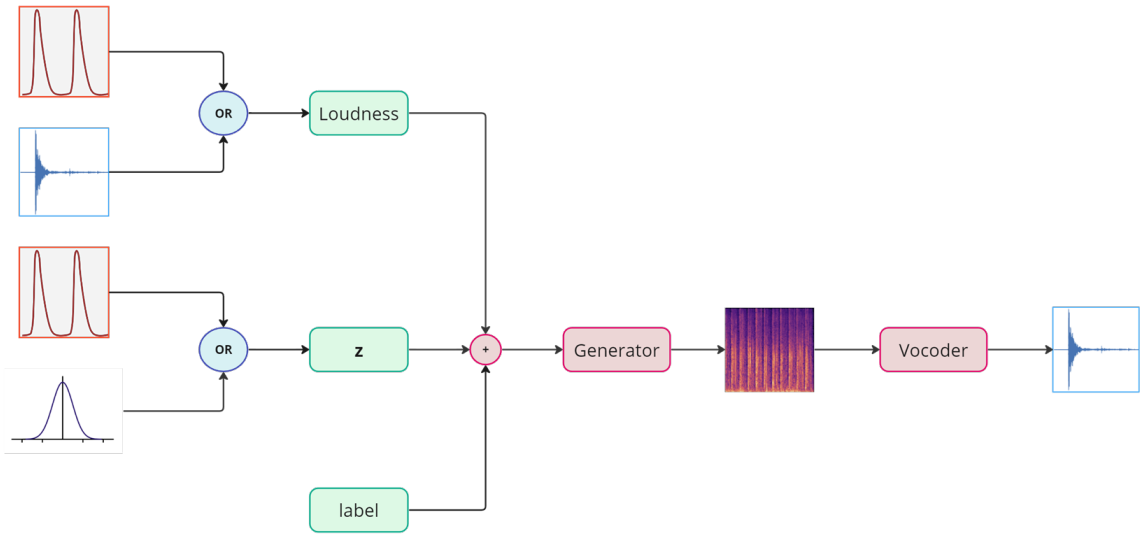


FIGURE 5.4. Model inferencing

After the model has been properly trained, we freeze the generator for the Mel spectrogram generation. Similar to the training stage, we still need to provide the loudness envelope and latent vectors. However, instead of extracting loudness from another sound and sampling a vector from a Gaussian distribution, we instead provide self-defined envelopes and latent vectors set arbitrarily via a manual user interface, with the constraint that the loudness and latent vector values are within an appropriate range.

For latent vectors sampled from Gaussian, an effective range of values would be within $(-3, 3)$. After passing in the required vectors, the generator generates a corresponding Mel-spectrogram, which is then converted back to audio by a pre-trained high-performance neural vocoder. In this work, I used the HIFIGAN[95] vocoder pre-trained on Audioset[185], a large audio dataset containing various categories of general sounds.

5.4 Experiments

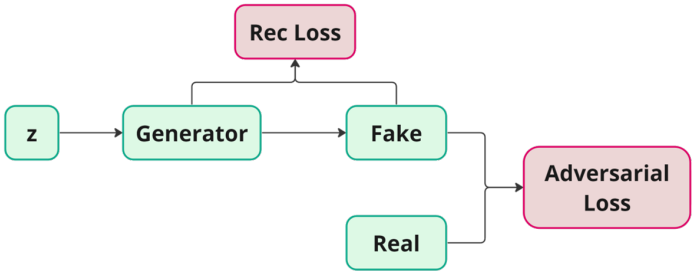
5.4.1 Training

To train our proposed model, I used the DCASE 2023[227] sound effect dataset. The dataset contains 7 categories of common sound effects, each of which are 4 seconds sampled in 22.5kHz. For consistency, I sampled all sounds in 16kHz and selected a frame size of 160, resulting in 400 time frames for each loudness envelope and Mel spectrogram. I then used an Adam optimizer with a learning rate of $l = 0.0001$, beta1=0.5, beta2=0.999. The model was trained in 10,000 iterations on an RTX 3080, which takes only 2.5GB VRAM within 20 training hours. Our implementation is available at: https://github.com/Reinliu/STGAN

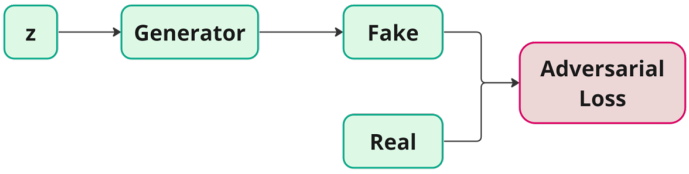
5.4.2 Evaluation

To understand the effect of our reconstruction loss on the quality and diversity of the generated samples, I compare our model with a few different configurations as shown in Fig. 5.5): STGAN with both the reconstruction loss and adversarial loss, STGAN-AL with only the adversarial loss, and STGAN-GEN-RL with only the reconstruction loss. To study the performance of the RNN-based generator, I have also trained a WGAN-GP model conditioned on class labels, which utilizes transposed convolutional neural networks for its generator.

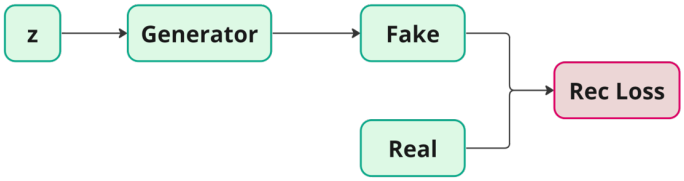
I selected a group of test sound effects for each category as the target sounds. The loudness envelopes were extracted from such targets and pass them into the four variant models to output the generated Mel spectrograms. Audio waveforms are then generated using the same pre-trained HIFIGAN vocoder[69]. In this way, we obtained a set of pair-wise target-and-generated Mel spectrograms and sound effects for each sound category. Our model is evaluated with different configurations based on the metrics below.



(a) STGAN: Our proposed model with adversarial loss and reconstruction loss added on generator loss.



(b) STGAN-AL: The same STGAN model with only adversarial loss.



(c) STGAN-GEN-RL: The same generator applied with an MSE loss on the generated output. The discriminator is completely abandoned in this setup.

FIGURE 5.5. Three different setups for evaluation

5.4.3 Metrics

FAD: Frechet audio distance is a metric calculated as the Frechet distance (also known as Wasserstein-2 distance) between the multivariate Gaussian distributions of the real and generated audio features. It measures the similarity between the distributions of real and generated audio samples. The FAD metric is commonly used for evaluating the audio quality of generative models[69, 242, 100, 99, 244]. Therefore, we decided to rely on FAD to evaluate the generated sound effect quality. A lower FAD score indicates that the distribution of generated audio is closer to the distribution of real audio, implying improved quality and

realism in the generated audio.

FID: Frechet inception distance[211] is a metric used to evaluate the quality of images generated by models such as Generative Adversarial Networks (GANs). It measures the similarity between the distribution of generated images and the distribution of real images. I computed the FID score on our Mel spectrograms. A lower FID score indicates that the generated images are more similar to the real images, implying better quality and realism of the generated images.

NDB: Number of statistically different bins[216] is a metric for assessing the diversity of images generated by a model. This metric is particularly useful for determining whether a generative model is capable of producing a varied set of outputs or whether it is limited to a narrow subset of the potential output space such as the known mode collapse issue in GAN. It has been used to evaluate the generated sound diversity in research [105, 203]. To compute the NDB, I clustered our Mel spectrograms into 100 bins. We then assign each generated Mel spectrogram to the nearest cluster bin and compare the distribution of generated samples across these bins with the distribution of real samples. The NDB score is calculated as the number of bins where the occupancy of generated samples significantly differs from that of the real samples. A lower NDB score indicates a more diverse range of outputs from the generative model.

LSD: We compute the pair-wise Log-spectral distance between the target sound effect and the generated sound effect to measure the timbre variations. We report on the averaged LSD score across all 64 frequency bins as follows: $LSD = \sqrt{\frac{1}{N}\sum_{f=1}^{N}\left[10 \cdot \log_{10}\left(\frac{S_1(f)}{S_2(f)}\right)\right]^2}$, where S_1 is the target spectrum and S_2 is the generated spectrum. Because all testing models are conditioned on the same loudness envelopes, they are expected to synthesize a similar spectrum with slightly different timbral variations, depending on the variability of the model. A larger LSD score indicates a higher spectral difference, meaning that the model is capable of producing more diverse variations in its spectrum for the same conditioning vector.

5.5 Results

Metrics **Description**	**FAD** ↓ Audio quality	**FID** ↓ Spec quality/diversity	**NDB** ↓ Mode collapse	**LSD** ↑ Spec diversity
STGAN	**10.21**	**79.60**	7.14	0.24
STGAN-AL	14.45	124.59	15.57	**0.25**
STGAN-GEN-RL	12.27	99.59	**1.43**	0.11
WGAN-GP	13.72	108.59	32.29	0.17

TABLE 5.1. Quantitative evaluation results based on various metrics

The sound examples could be seen at our Appendix .9. In Table 5.1, we show the results of the quantitative evaluations. Our proposed method, STGAN, achieves the best FAD and FID scores, indicating that this approach is capable of synthesizing high-quality Mel spectrograms and, as a result, audio waveforms. On the contrary, WGAN-GP performs the worst in FAD, potentially due to the upsampling artifacts that we discussed earlier. With respect to the FAD, our model surprisingly outperforms the STGAN-GEN-RL, which should theoretically yield a lower FAD score attributed to the pixel-level spectral loss. We hypothesize this is because the combined loss enforces consistency in the generated output, making sure that essential characteristics and modes of the spectrograms are preserved. This could also be seen from the NDB scores, where incorporating the reconstruction loss on loudness envelopes helps prevent mode collapse.

The STGAN-GEN-RL captures most of the modes from the dataset, which is logical in that the MSE loss penalizes any difference from the real samples. As for the STGAN-AL configuration, we found this setup can output higher sample diversity than all other configurations but performs the worst in terms of audio or spectral quality. This corresponds with the observation that there is generally a tradeoff between generated sample quality and diversity. On the other end, even though the discriminator applies a gradient penalty to improve training stability, the WGAN-GP model still encounters mode collapse as indicated by the NDB measurement. We hypothesize this is because transposed CNN operations are adept at learning global and spatial patterns quickly.

From a traditional GAN setup, we observed the CNN-based generator becoming too powerful in the early training stage and leading to mode collapse. Lastly, although STGAN-AL

achieves the highest LSD score, we found that STGAN, with both reconstruction loss and adversarial loss, still performs well in the generated sample diversity. This is largely because we only compute the reconstruction loss on the high-level loudness features without penalizing pixel-level spectral differences, thereby allowing higher sample diversity and variations.

5.5.1 Creative Application

Because of the recursive nature of our generator, our proposed model is capable of synthesizing arbitrary-length Mel spectrograms by inputting loudness envelopes with any length. Above we show an example of a generated footstep sound effect 5.7 by providing a recording of voice 5.6 as the guiding sound for loudness extraction. The generator successfully preserves the rhythmic patterns of the voice amplitudes, while changing its timbre/frequency distribution into a particular type of footstep sound.

From the audio waveforms, it can be seen that the temporal sonic events of the generated sound match exactly with the original voice. Additionally, it is also shown that the generated footsteps contain more tails in its envelopes, meaning that the loudness envelope functions primarily as a subtle indicator of sonic events, rather than as a rigid, discrete one-hot vector for controlling generated sounds.

In terms of amplitude, the generated ones are louder than the original voice, which corresponds to the general amplitude of the footstep category. This shows that the loudness envelope mainly serves as a relative rather than an absolute indicator for the dynamics. The absolute amplitude is more tied to the original data distribution (footstep sounds in this case) rather than determined by the loudness envelope.

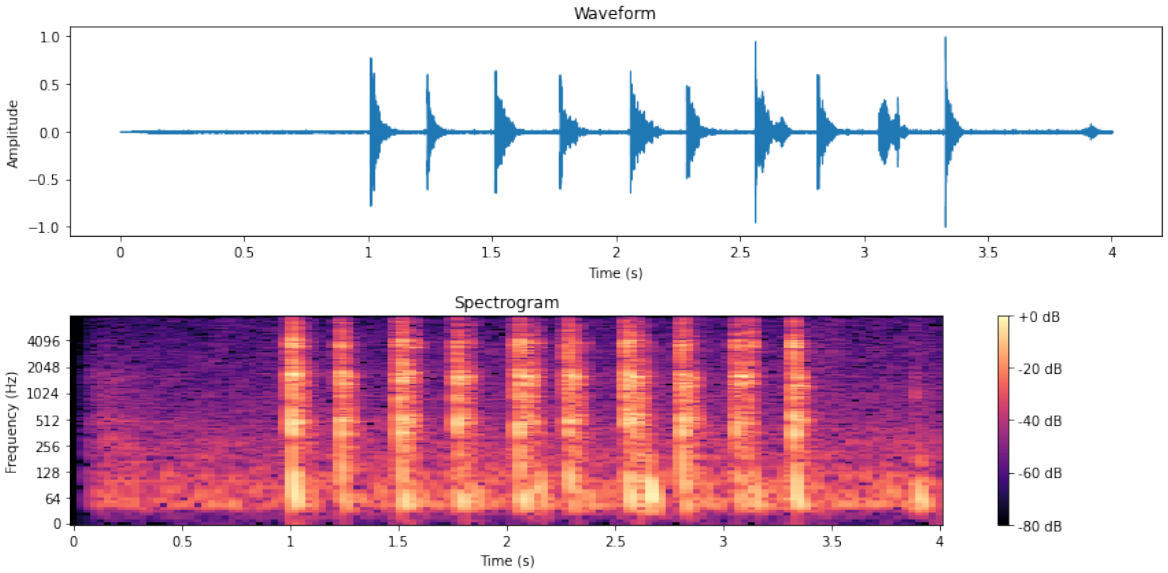


FIGURE 5.6. Guiding voice

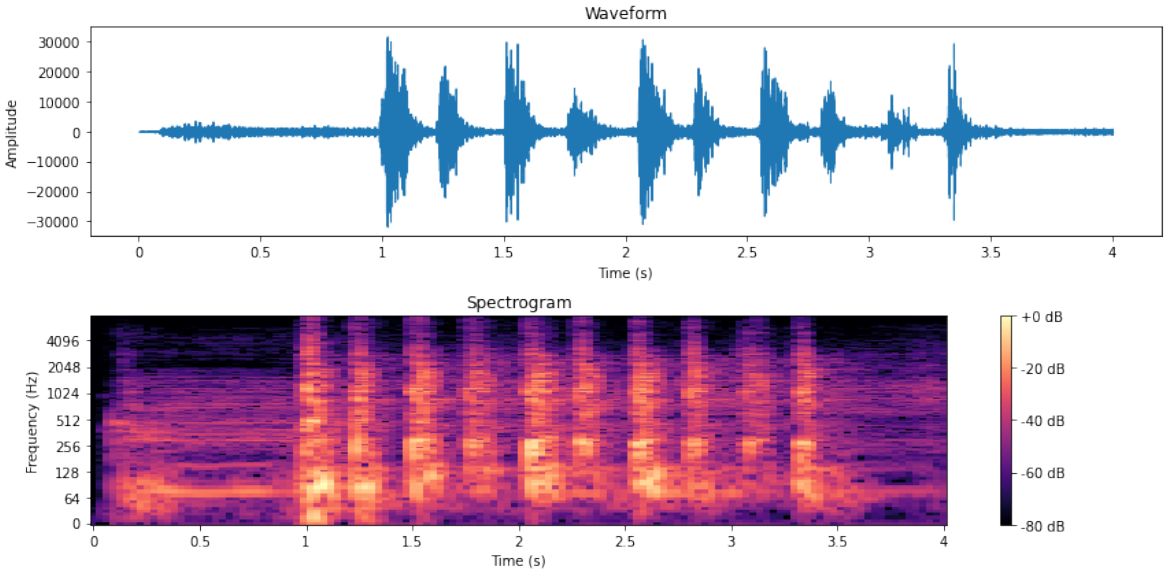


FIGURE 5.7. Generated Footstep

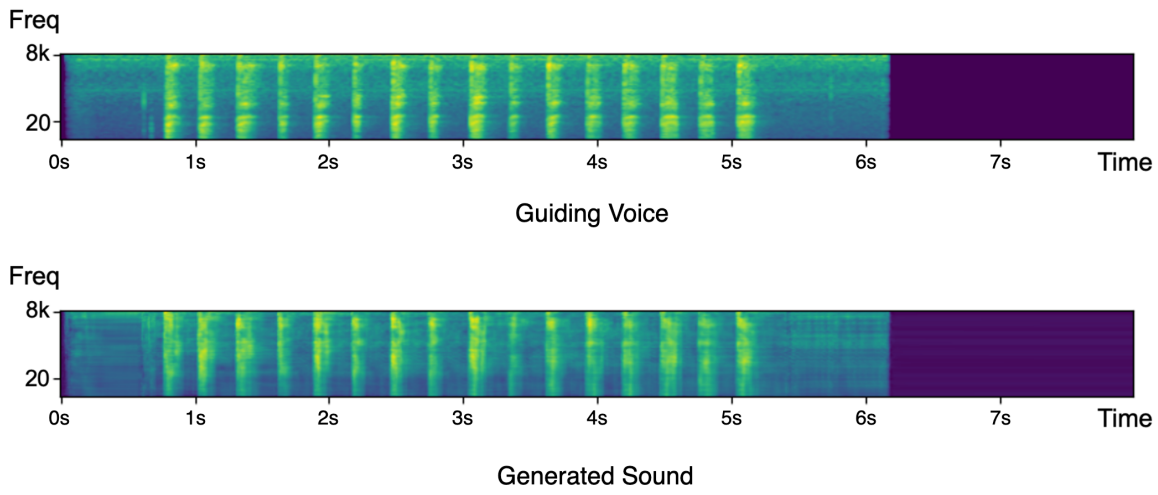


FIGURE 5.8. Variable-length spectrogram generation using RNN. The generator is composed of three layers of GRUs, which sequentially model the frequency contents of the spectrogram. Notice that our model is trained solely on an audio dataset that each audio is exactly four seconds in length. Owing to its auto-regressive nature, this architecture allows the generator to synthesize continuous spectral contents sequentially. This means that our decoder has learned the subtle temporal relationships between audio samples. At the inference stage, our model is capable of synthesizing variable length spectrograms, which can then be converted back to the time domain with neural vocoders [95].

5.5.2 Variable-length Spectrogram Generation

5.6 Chapter Summary

In this research, I proposed a neural audio synthesis model capable of arbitrary-length sound effects generation with timbre controls using GAN by relying on recurrent neural networks for spectrogram generation. I found that the frame-level concatenation of a preprocessed loudness envelope with the latent vector allows for independent control of the waveform amplitude as well as spectral characteristics of the sound at the same time.

Furthermore, I reported the trade-off between generation quality and sample diversity by integrating an additional reconstruction loss into GAN architecture. Our results indicate

that GAN sound synthesis is stabilized by adding an additional reconstruction loss derived from or based on acoustic features. Our proposed model provides independent control of amplitude and timbre while maintaining high fidelity and can hopefully contribute to creative applications in-game audio and interactive media.

CHAPTER 6

DDSP-SFX: Acoustically-guided sound effects generation with Differentiable Digital Signal Processing

---

6.1 Introduction

Sound effects are diverse and usually difficult to describe using words. To this end, many synthesis algorithms focus on using vocalizations to guide sound generation because voice can be an intuitive control interface for end-users. The supervised phoneme-based control-synthesis approach, which converts human-produced phonemes directly to sound effects, has been widely explored [245, 246, 247, 248].

However, such methods typically require ground-truth labelling between the phonemes and the corresponding sound effects. The unsupervised approach, which entails extracting acoustic features from the guiding sounds and applying such features to the generated sound effects, has also proven to be very effective in musical sound modelling. In Section 3.1, we had a detailed discussion about each type of generative model for sound synthesis. Particularly, the neural+DSP hybrid approach has drawn much attention to the field of music instrument modelling owing to its mixed usage of machine learning to model implicit relationships and DSP algorithms to define explicit sound synthesis equations.

For example, differentiable digital signal processing [13] is a popular NAS architecture introduced to take advantage of pre-built digital synthesizers for waveform generation. Neural networks are then used to estimate the parameters for the corresponding synthesizers conditioned on pre-processed acoustic features. Owing to the modular approach and conditioned

features, DDSP is lightweight to train and use while offering controllable audio synthesis over pre-defined audio features such as pitch and loudness.

Although Lundberg [133] has applied DDSP to model motor engine sounds, there have been few attempts at modelling other sound effects, especially impulsive sounds such as footsteps or gunshots, two commonly used sound effects in games. To properly model the transient information, Barahona & Collins [249] proposed an alternative technique to model the inharmonic sounds through a controllable filterbank based on a pseudo-quadrature mirror filterbank (PQMF) [250] synthesizer under a similar DDSP framework.

However, due to its design of using a large number of filter banks, it generally takes a lot of computational resources and time to synthesize high-quality sound effects with high dynamics. Additionally, it remains a critical issue as to how to effectively control the subtle timbre variations of a particular sound effect driven by vocalizations, given the differing acoustic characteristics between voice and various desired sound effects.

In this research, we explore sound effects modelling driven by acoustic features of target sounds utilizing the DDSP architecture. We are particularly interested in controlling the subtle timbre characteristics of the target sounds using a guiding sound, such as varying the type of engines for a motor sound with fixed pitch and loudness. To this end, we propose a novel approach for frame-level timbre variation independent of other acoustic attributes. Our method only requires a trained decoder to transfer the pre-defined acoustic envelopes and timbre envelopes from one type to another type of sound, without relying on an additional encoder. We further introduce an architecture targeted at the synthesis of impulsive sound effects based on the DDSP framework.

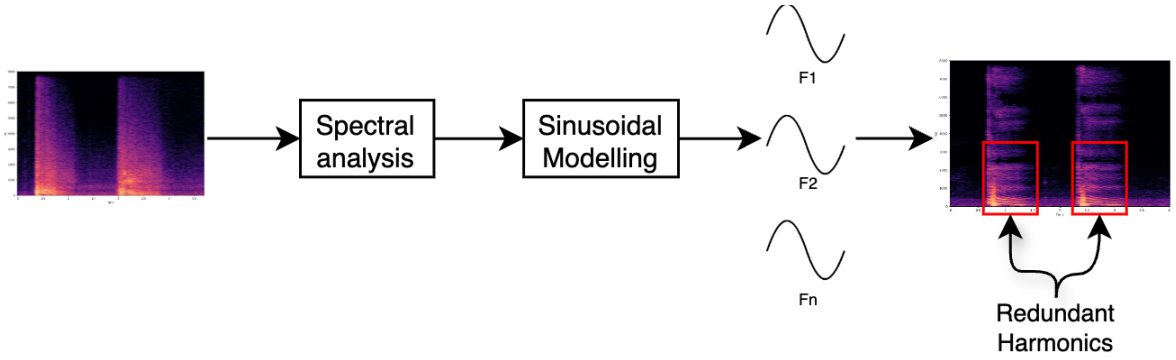


FIGURE 6.1. Issue of DDSP for SFX synthesis (1): DDSP places too much emphasis on sinusoidal modelling that results in artefacts.

6.2 Issues with DDSP

6.2.1 Redundant Harmonics

When using the original DDSP architecture to synthesize sound effects, in our preliminary study we realized the decoder does not learn to attenuate its sinusoidal synthesizer even though the reference sound is completely inharmonic. In Figure 6.1, we show how redundant harmonics look in the spectrogram, which is synthesized from the sinusoidal synthesizer. This can be a critical issue for modelling many categories of sound effects, such as impulsive rigid body impacts or inharmonic environmental noises.

6.2.2 Temporal Timbre Control

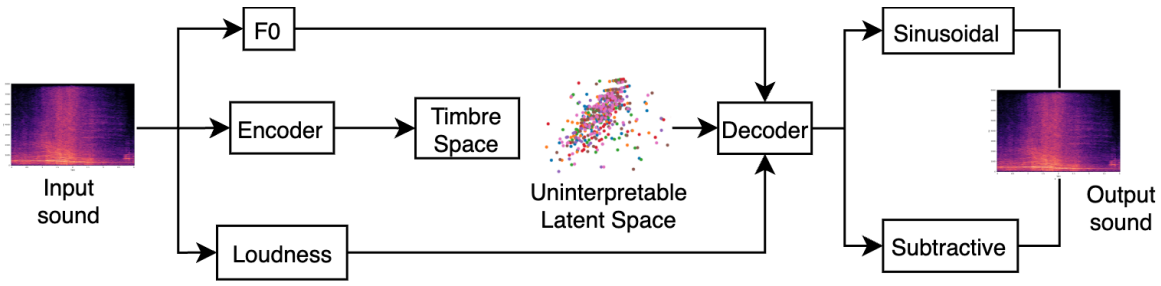


FIGURE 6.2. Issue of DDSP for SFX synthesis (2): Uninterpretable latent space in DDSP autoencoder.

The original DDSP is based on an auto-encoder architecture, whose latent space is a compressed representation of the audio. Without explicit remapping [251, 252], or distillation [84, 129], latent space of generative models are most of the time uninterpretable [163], meaning that varying the latent vector linearly does not result in meaningful output changes.

As shown in Figure 6.2, the latent space restores all the timbre information of the training audio. During the inference stage, the generation is only valid when another piece of audio is used as a reference by obtaining the latent representation from the reference using the encoder, as well as other features extracted explicitly. This process is called timbre transfer, as the decoder takes the reference sound's features while retaining the timbre of the original audios it has been trained on.

The DDSP architecture, therefore, is not capable of controlling the timbre with some parameters, as the latent space remains uninterpretable. It also relies on reference sounds to output meaningful sounds.

6.3 Proposed method

We base our model on the vanilla DDSP architecture [13], which uses a sinusoidal model and a subtractive noise model as synthesizers. To better adapt the DDSP architecture to synthesize high-quality sound effects, we integrate a separate transient modelling method in the synthesizer and introduce a technique to avoid harmonic artifacts. To be able to control the timbre variations effectively, we propose an encoder structure that allows for frame-level timbre control. Our complete model architecture is depicted in Figure 8.3.

Before the training stage, in addition to the pitch and loudness vectors described in DDSP [13], we also need to pre-process the audio to extract a few acoustic features. The acoustic features and the latent variable output from the encoder are concatenated together to be processed by the decoder, which is composed of several Multi-layer Perceptrons (MLPs) [253] and Gated Recurrent Units (GRUs) [254] as illustrated in the architecture. The decoder finally outputs the required synthesis parameters for the synthesizer, namely, frequencies and amplitudes of the

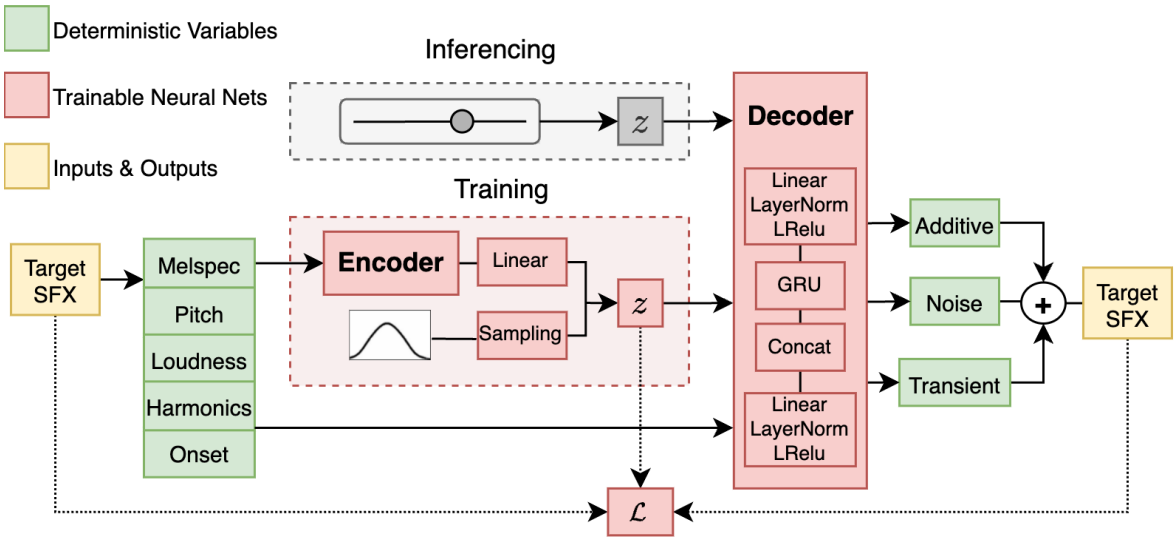
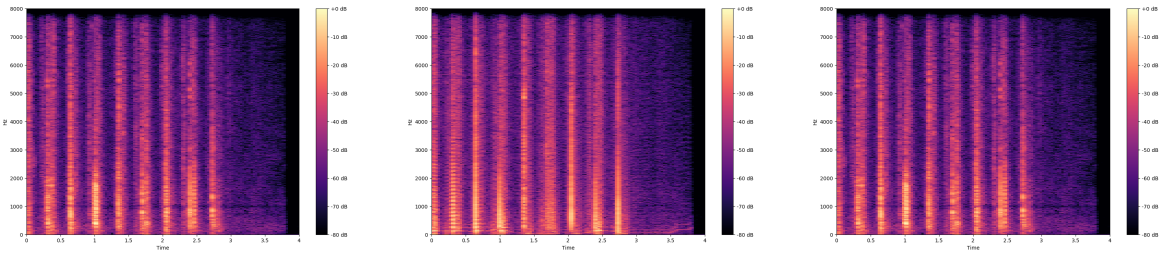


FIGURE 6.3. Proposed model architecture. Blocks in green are definitive features or algorithms. Blocks in pink are composed of learnable neural networks. Blocks in grey indicate they are used during the inference phase to replace the trained encoder.

sinusoids for the additive synthesizer, amplitude vector for the subtractive noise synthesizer, and finally, frequencies and amplitudes for the transient model. Lastly, all the synthesized outputs are combined together as the synthesized out. In the following sections, I will elaborate on the proposed techniques.

6.3.1 Harmonic Indicator

A harmonic plus noise model [36] performs well when the target sound is harmonic. However, when integrated into DDSP, it could lead to harmonic artifacts or distortion for sounds with little to no harmonic components. This is because both the harmonic synthesizer and the subtractive noise synthesizer are treated equally in the synthesis. Through our preliminary study, we found that when modelling impact sounds such as footsteps and gunshots, the original DDSP architecture tends to emphasize the harmonic model even though this may lead to unnecessary synthesized harmonics. The reconstruction loss, multi-scale STFT loss, only considers the summed energy across each frequency bin. This loss function is not able to penalize the excessive modelled harmonics, especially when the target sound contains mainly



(A) Reference footstep (B) DDSP (C) DDSP-SFX

FIGURE 6.4. Harmonic artifacts with DDSP. Notice the horizontal lines prevalent in DDSP synthesized footsteps, especially in the first two events. By using a harmonic indicator, we are able to attenuate the harmonic synthesizer and thereby rely mostly on our subtractive noise and transient model for synthesis. To listen to how these sound like, please refer to our accompaniment website: https://reinliu.github.io/DDSP-SFX/.

noise, with flat energies across each frequency. In figure 6.4, we show an example harmonic artifact generated by DDSP of a footstep sound.

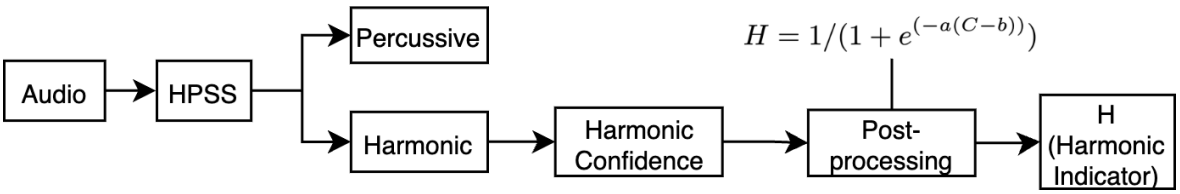


FIGURE 6.5. Harmonic analysis for guiding DDSP. We first separate the audio into harmonic and percussive components. A harmonic indicator that indicates whether the sound is harmonic, is returned and used as input to the decoder, which then learns to attenuate the harmonic synthesizer when the harmonic indicator score is lower and vice versa.

Therefore, to properly model the inharmonic sounds under this architecture, we decided to employ a harmonic detector to determine the degree of harmonic components present in the sound and then train the model to attenuate when few harmonics are detected. This process is shown in Figure 6.5. Specifically, during pre-processing, we scan the entire sound with a pitch detector [167] across every frame and obtain a confidence score C (0-100%) for the pitch estimation of the modelled sound. Rather than using the scores directly as guiding information, we wish to obtain a smoother and flatter curve to 'activate/deactivate' the harmonic synthesizer. To this end, we input C to a custom Sigmoid function to smooth

out the output value as shown below:

$$H = 1/(1 + e^{(-a(C-b))}), \tag{6.1}$$

where a determines the steepness of the curve, and b determines the horizontal shift for the Sigmoid function. As both a and b are hyper-parameters, we later found setting $a = 10$ and $b = 0.7$ effective in training on our dataset. Here, we call the output value H from the Sigmoid function as the harmonic indicator. This indicator is further passed as input to the decoder as conditioning information for the presence of harmonic components. When H returns a low value (indicating low confidence of harmonic information present in a frame), the decoder learns to attenuate the harmonic synthesizer and vice versa.

6.3.2 Transient modelling

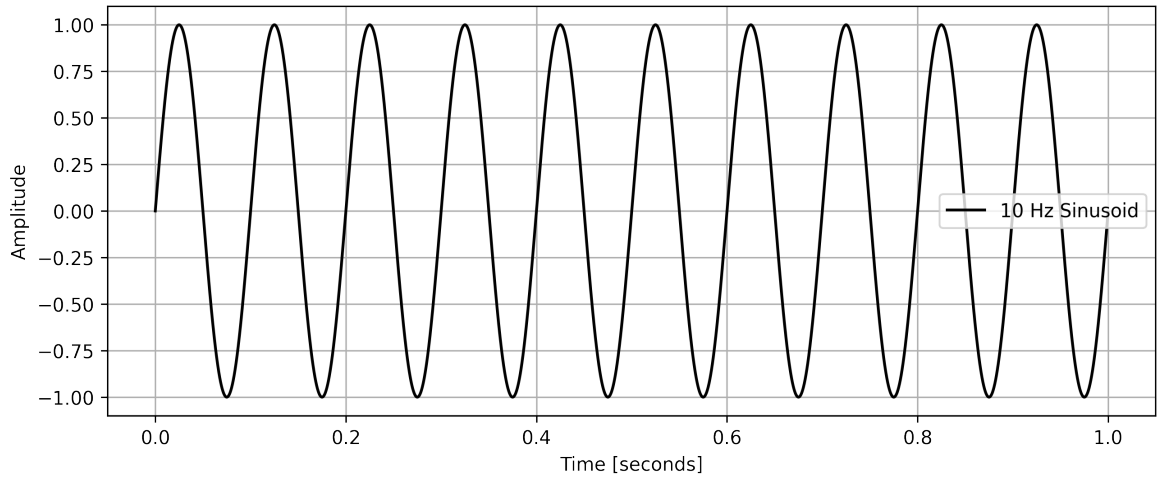


FIGURE 6.6. A 10hz sinusoid signal in the time domain.

From an analysis perspective, a signal in the time domain could be separated into three parts [255], harmonic components, transients (which are modelled as impulses [134, 42], and residual noise. A transient signal (e.g., gunshots, footsteps) has a sharp attack and short sustain, which can be difficult for sinusoidal modelling or subtractive noise modelling.

To synthesize transient signals, we employ a similar modelling approach as [134, 133]. Consider an exponentially decaying signal 6.7, which sounds like an impulse with high

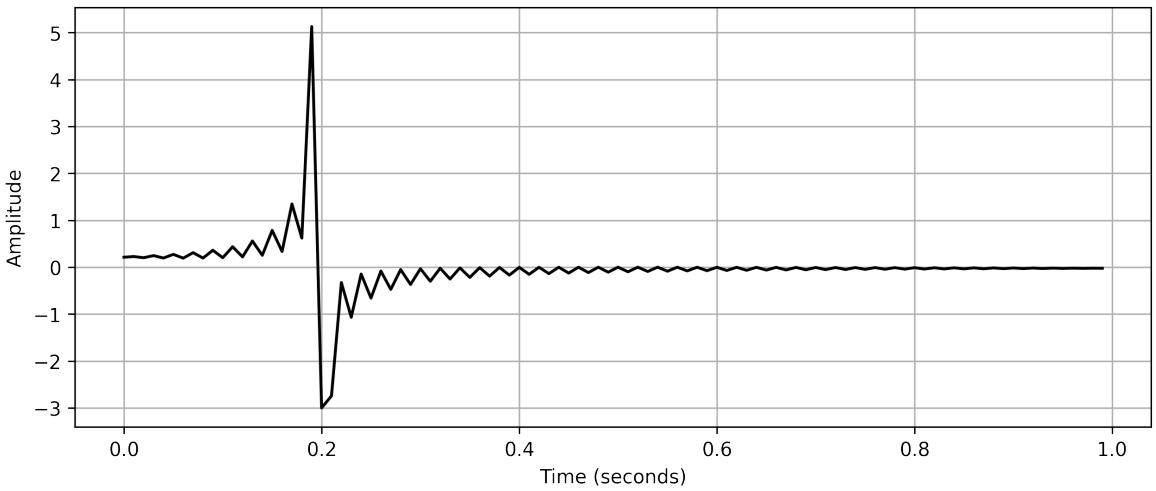


FIGURE 6.7. The exponentially decaying signal. It has been converted to time domain using inverse DCT from the 10hz sinusoid signal.

amplitude in a short time. Its discrete cosine transform (DCT) would look like a slowly varying sinusoid signal 6.6. We could take advantage of this interesting property to model the transient signals, by synthesizing sinusoids in the discrete cosine domain and converting to the time domain using an inverse Discrete Cosine Transform (IDCT). Due to the nature of time-frequency transforms, this results in impulses in the time domain.

Instead of placing transients equal-distantly across the time frames as was done in [133], we instead chose to model them within each frame, as we wish to treat each transient differently and be able to control its timbre. This process is shown in Figure 6.8. A sinusoid in the DCT domain would translate to a single pulse in the converted time domain through IDCT. The frequency in the DCT domain controls the time location of the pulse, from the start 0ms to 10ms across 160 samples in a frame. Please note that even though the time is extremely short (10ms), changing the time location of the pulse would change the timbre of the impulse clip, as humans are very sensitive to phase changes in transient sounds.

For each of the 400 time frames of the 4s long signal, a decoder network will learn and output the parameters of the sinusoids used for transient modelling: i.e., the frequency F_n and amplitude A_n. If no transient is present in a given time frame, then A_n should be 0. This

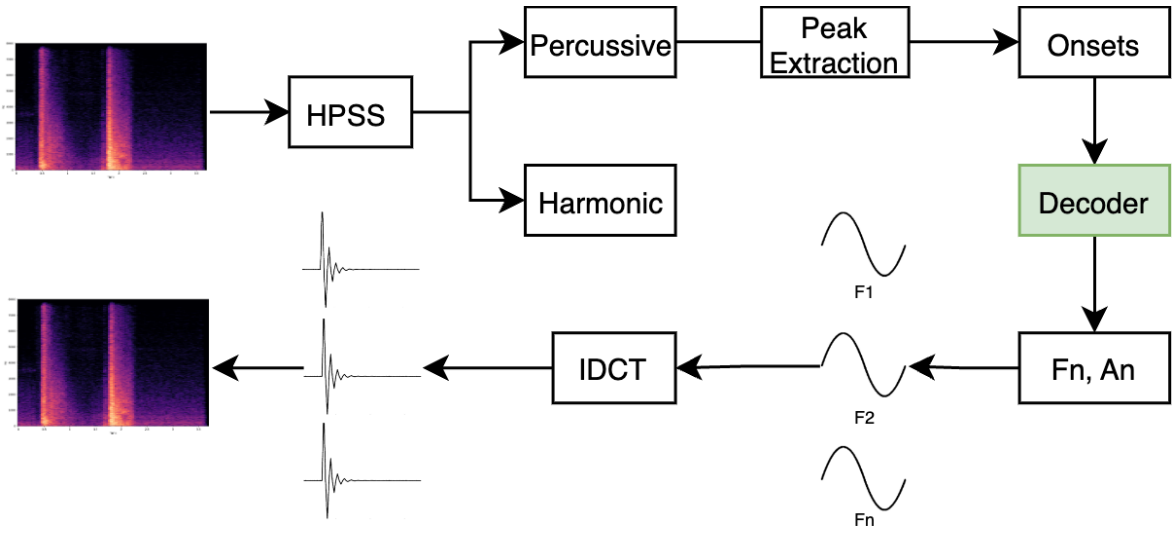


FIGURE 6.8. Transient synthesis approach. We synthesize several sinusoidal waveforms in the DCT domain, parameterized by the learnable frequency and amplitude variables output from the decoder, and then convert them to impulsive signals in the time domain via inverse DCT. The decoder receives an onset vector that records the transient amplitude envelope information extracted from the input audio waveforms.

provides a convenient means for controlling the transient signals. The transient modelling equation is defined as:

$$x[n] = \sum_{n=0}^{N-1} A_n \text{IDCT} \sin\left(2\pi F_n \frac{t}{f}\right), \tag{6.2}$$

where $x[n]$ is the modelled transient signal, t is the total time samples, f is the frame size and N is the total number of frames. In order to provide guiding information to the decoder, we extract peak onsets from each sound. The onset amplitude vector is constructed from the signal spectrogram using a margin-based Harmonic Percussive Source Separation (HPSS) method [256] with a conservative and large margin parameter value of 8 during the pre-processing stage. Local maxima (peak estimation) is performed to obtain the onset amplitude vector where the percussive events take place.

6.3.3 Frame-level timbre control

In addition to guiding the sound synthesis with explicit acoustic features such as pitch and loudness, we also wish to encode the subtle timbre variations of the generated sound into a controllable latent space. To this end, we employ a similar encoder structure introduced by Devis et al. [129] and train DDSP as a Variational Autoencoder (VAE) [18]. We first compute the mel-spectrograms (128 mel-frequency bands, 400 time frames) of our input sounds in the pre-processing stage. They become the input to our encoder, comprising three stacks of convolutional 1-D layers, RELU activation, and batch normalization layers followed by a linear layer. The mean and log-variance output from the encoder are then reparameterized to produce the sampled latent vector z:

$$z = \mu + \epsilon \cdot \sigma^2, \tag{6.3}$$

where ϵ is a random value sampled from a unit-Gaussian distribution, μ is the mean output from the encoder, $\sigma = e^{0.5 \cdot \text{logvar}}$, and logvar is the logarithm of the variance. In this way, we obtain a continuous one-dimensional latent vector along the time axis. The distribution of z is regularized to be close to a unit-Gaussian $\mathcal{N}(0, 1)$. After the model has been trained, we can deliberately modify the value of z within ± 3 as 99.7% of the values are within three standard deviations. The most "typical" timbres appearing in the data set are encoded with z close to 0, whereas "rare" timbres are encoded with z values farther away from 0.

Once the model has been trained, we can completely abandon the encoder structure. In the inference stage, apart from explicit control over pitch and amplitude, we can also output different timbres of the sound (eg. transforming from a BMW car engine sound to a Mercedes) by creating a control variable at hand within the same range ± 3 to replace z for the decoder. A global slider is used to adjust this pseudo latent variable and thus vary the timber of the output sound. In this way, we are able to create interesting sound effects by providing explicit and implicit control variables along the time axis. To demonstrate how it works, in section 7.5, we use human voice mimicking sound effects as an example to guide the sound synthesis.

6.3.4 Loss Function

Our loss function contains a regularization loss component and a reconstruction loss component. We use the same multi-scale STFT loss as used in DDSP for our reconstruction loss (FFT sizes: 2048, 1024, 512, 256, 128, 64). For the regularization loss, we apply a scaling variable β on the regularization loss term to prevent the reconstruction loss from overtaking the total loss [163]. The total loss function in our model becomes:

$$\mathcal{L} = \mathcal{L}_{rec} + \beta \cdot \mathcal{L}_{reg}, \tag{6.4}$$

where the $\mathcal{L}$ indicates the total loss, $\mathcal{L}_{rec}$ is the reconstruction loss, and $\mathcal{L}_{reg}$ is the regularization loss.

6.4 Experiments

6.4.1 Dataset

We use the publicly available sound effects dataset [227]. It includes 7 categories of sound effects, all of which are sampled at 22.5 kHz and have a length of around 4 seconds. Each category contains 581-800 audio samples. The data used for all of the experiments described in this paper consists only of footsteps, gunshots and motor sounds. This is because we found these three categories contain the most variations and they suit the best for our goal as to effectively control the timbre of the sounds. We refer to these three sound types as our three data sets. All sounds are trimmed to 4 seconds duration and down-sampled at 16 kHz to be consistent with our neural network architecture. We split the data 90% for training and 10% for testing.

6.4.2 Training

We train our model and the vanilla DDSP model on the same three data sets separately, resulting in three trained models for both the vanilla DDSP and our proposed method. Each

model is trained with a batch size of 16 on an RTX 6000 GPU for exactly 100,000 training steps. We chose a size of 1024 for the recursive hidden units of DDSP, 100 sinusoids for the harmonic synthesizer, 100 bands for the noise synthesizer, and a frame size of 160 samples. We use an ADAM [257] optimizer with a starting learning rate of $1e^{-4}$, which gradually decayed to $1e^{-5}$ after 80% of the steps. To maintain a stable regularization loss and balance it with the reconstruction loss, we initialize β with 0, and activate it only after 10% of the training steps with $\beta = 1$, and then scale it linearly to $1e^{3}$ until reaching 80% of the training steps.

6.4.3 Evaluation

To show that our latent space is capable of outputting discernable timbre variations, we performed a small-scale listening test on 26 participants. To guide the synthesis, we recorded three human vocalizations emulating the sound effect for each category of our data set as our out-of-domain guiding sounds. We then use the extracted acoustic features from these guiding sounds to perform timbre transfer. For our latent vector, we manually set z as 0, 0.1, 0.5, 1, 2, and 3 for each sound clip and then generate the sound effects correspondingly. We use $z = 0$ as our reference track and ask the participants to do a forced comparison test to see whether the individual tracks with z set as different values sound identical to or different from the reference. The result is shown in Section 6.5.3.

We evaluate our model in terms of synthesis quality. The timbre encoding performance is demonstrated qualitatively. For each model, we pair a reference sound with the generated sound. Each model is tasked to synthesize waveforms similar to those of the reference by taking in the extracted acoustic features from the reference. We compare the synthesis performance through a series of objective metrics and a subjective listening test. We conduct a second subjective listening test to understand the effectiveness of our timbre encoding.

6.4.3.1 Synthesis performance

Statistical similarity. Frechet Audio Distance (FAD) [210] is an audio quality evaluation method that compares the feature representations through an embedding layer of a pre-trained

audio classification model between the generated sounds and the reference sounds. We compute the FAD score with the VGG model [1] to understand the statistical similarity of our synthesized sounds compared with the reference sounds.

Spectral similarity. We report the log-spectral distance (LSD) using a multi-scale STFT to measure the spectral similarity between the synthesized sound and the reference sounds. It is computed as the average distance between two power spectra over all frames in the Euclidean space. We use the same window length, hop length and FFT size as we did in the pre-processing to compute the spectra.

Audio quality. We conduct a listening test to evaluate the generated sound quality. We randomly selected ten examples per category. For each question, we asked the participants to give an absolute category rating for each soundtrack from 1 (bad) to 5 (excellent). We aggregate our results per category of sounds, meaning that we average the ten-question results and obtain the variance for the aggregated data.

6.5 Results

6.5.1 Audio similarity

TABLE 6.1. *Objective audio synthesis performance results.*

Categories	Footstep	Gunshot	Motor
Metric	FAD ↓		
DDSP	5.356	5.213	**6.652**
DDSP-SFX	**1.529**	**2.004**	7.150
Metric	Log Spectral Distance ↓		
DDSP	0.114	0.585	**0.177**
DDSP-SFX	**0.103**	**0.446**	0.182
Metric	Multi-scale STFT ↓		
DDSP	1.63 ± 0.43	2.23 ± 0.72	$\mathbf{1.09} \pm \mathbf{0.06}$
DDSP-SFX	$\mathbf{1.55} \pm \mathbf{0.44}$	$\mathbf{2.03} \pm \mathbf{0.81}$	1.10 ± 0.07

[1] https://github.com/gudgud96/frechet-audio-distance

Referring to Table 6.1, our objective measures correlate well, suggesting a significant improvement in impulsive sounds (footsteps and gunshots) after we integrated our transient modelling method. For motor sounds that are steadily pitched across the entire signal, our method performs slightly worse than DDSP. This is expected because introducing the regularization term to the loss poses more challenges to the reconstruction. We also show that our transient modelling method can synthesize sharper attacks for impulsive signals in our accompanying website [2].

6.5.2 Audio quality

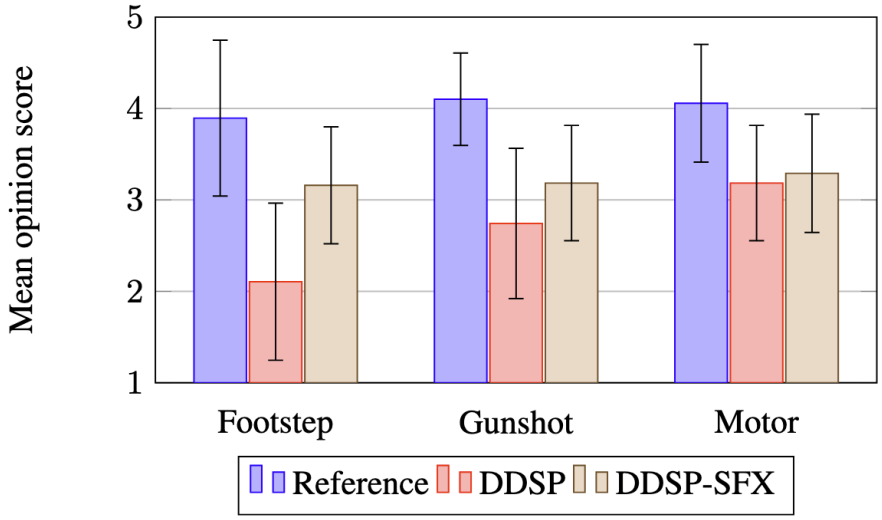
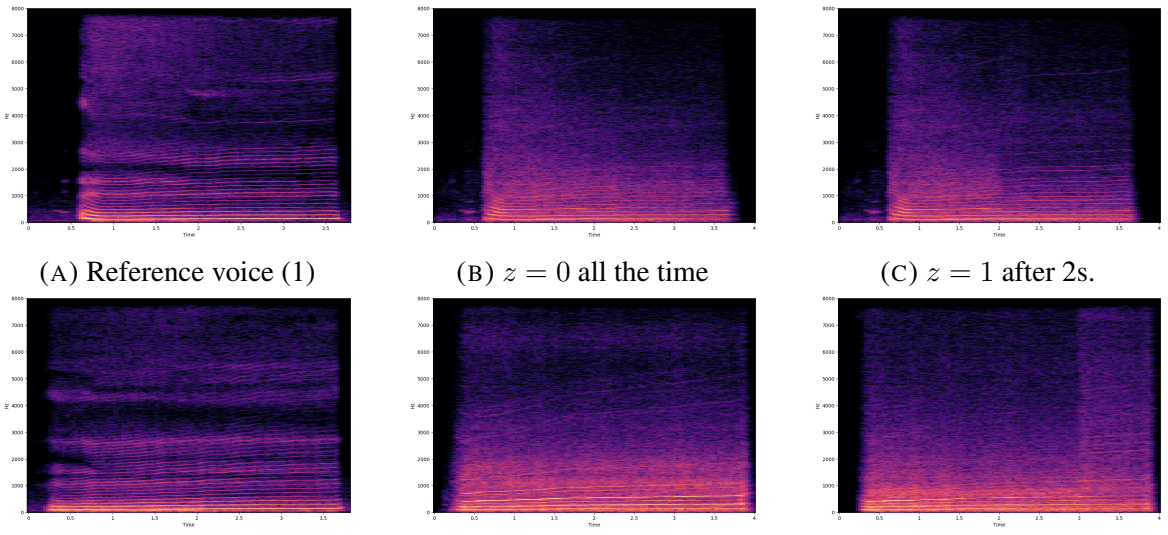


FIGURE 6.9. *User-rated audio quality of sound effects.*

There were 26 participants (M/F: 19/7; ages: 23-53; audio experts/non-experts: 14/12) in our subjective listening test. (The listening test sample questions and content forms could be found at Appendix .1) Each participant was requested to use a pair of headphones for the listening tests. Each question contains a reference sound (the ones we randomly selected from the dataset), a sound generated from DDSP and a sound generated from ours. The reference tracks are expected to receive the highest scores (4-5). Therefore, we removed two outliers that rated the reference tracks below 2 for over 50% of the questions, resulting in 24 effective

[2] https://reinliu.github.io/DDSP-SFX/



(A) Reference voice (1) (B) $z = 0$ all the time (C) $z = 1$ after 2s.

(D) Reference voice (2) (E) $z = 0$ all the time (F) $z = 1$ after 3s.

FIGURE 6.10. An example of how changing z affects the overall timbre. We use a reference voice mimicking motor sound as guidance shown in Figure 6.10a and Figure 6.10d. In figure 6.10b, we show what the spectrum looks like when $z = 0$ for all the time frames. In figure 6.10c, we show that the spectrum changes accordingly when we variate z after 2 seconds. In Figure 6.10f, we set the latent vector to 1 after 3 seconds. It was shown that the spectral distribution of the generated motor sounds changes as we variate the latent vector temporally. Surprisingly, this abrupt latent space variation did not result in clicks or glitches in the generated sounds. We attribute this to the use of an auto-regressive decoder with RNN.

participants. Fig. 6.9 shows a bar plot of the collected mean opinion scores for the sound quality with variance as error bars. Our subjective test results are similar to our objective measures, where we see significant improvements for the impulsive sounds. The footsteps generated by DDSP were rated lower than our method, as many participants noticed the harmonic artifacts. Further, the motor sounds were rated similarly among the two methods.

6.5.3 Timbre encoding

There were 19 participants (M/F: 11/8; Age: 23-53; Audio experts/non-experts: 9/10) in our second listening subjective test. Table 6.2 shows the percentage of the participants who rated the synthesized sounds as different from the reference sounds when we vary the value of z. Our subjective test results indicate that most participants recorded timbre differences

TABLE 6.2. *Percentage of sounds rated as different from reference.*

z	Motors	Gunshots	Footsteps
0.1	0.255	0.235	0.216
0.5	0.451	0.431	0.510
1	0.549	0.608	0.745
2	0.686	0.804	0.843
3	0.941	0.826	/

for $z > 1$. This follows reasonably because the encoder learns to encode the "most typical" timbres across the data set within one standard deviation. Larger values of z indicate a rarer timbre across the whole data set. Depending on the variations available within the dataset itself, the value of z for which users start to tell the timbre differences will likely change accordingly. Lastly, to demonstrate the effectiveness of our timbre encoding, we show how varying z temporally could contribute to timbre variations in Figure 6.10. For more examples, please refer to our supplement website.

6.6 Discussion

In this paper, I integrated DDSP with a transient model and show that it improves the synthesis result for impulsive sound effects. I proposed a simple method that achieves timbre variation of the generated sound effect while also enabling deterministic attribute transfer given a limited dataset. I further demonstrate the out-of-domain timbre transfer capability by using human vocalization as guiding sounds. I hope our method will contribute to creative sound design by allowing users to create realistic sound effects using their own voices as guiding sounds. Future work may include training our model on a larger audio dataset with more variations to enable more expressive timbre control.

CHAPTER 7

ICGAN: An implicit conditioning method for interpretable feature control of neural audio synthesis

---

7.1 Introduction

In recent years, neural audio synthesis has achieved excellent performance in speech modelling [77, 96] and music generation [110, 12]. However, it remains difficult to model sound effects with pure generative models in that neural networks typically offer a low degree of interpretability and controllability. A common approach to achieving sound generation control is via conditioning on external information.

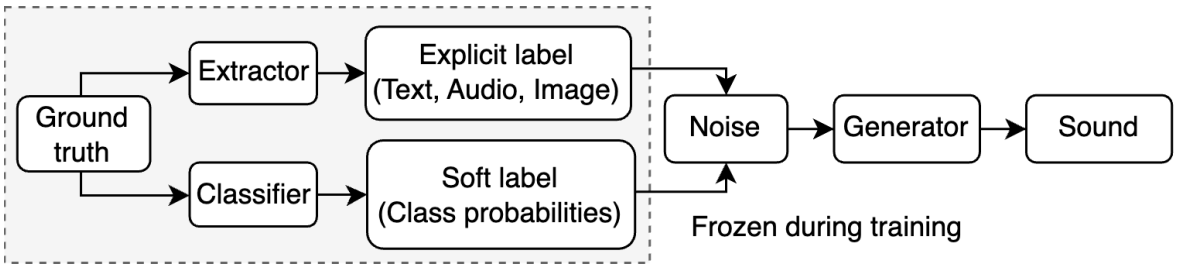


FIGURE 7.1. Conditioning labels for generative models. Broadly speaking, they can be separated as explicit labels or implicit labels(soft labels) [84]. Explicit labels are usually supplied with the dataset such as text descriptions for an audio recording or video frames for a video. They can also be derived by defining explicit functions such as audio features (pitch, loudness, spectral density, etc). Soft labels are usually implicit representations obtained from the datasets, which record complex information and descriptions about the sounds, but usually remain uninterpretable due to their implicit nature when it is obtained.

Conditional generation is a common technique to constrain generative models to generate data based on specific input information, which can be categorical labels, texts, images, compressed embeddings, etc. Considering the limited availability of datasets for general sound effects, conditioning on discrete labels can be effective, as most datasets provide at least categorical information. For example, Barahona-Ríos and Collins [100] proposed to synthesize knocking sounds conditioned on different emotions to guide the GAN generator for audio generation. Similarly, Communita et al. [99] also studied GAN sound generation for modelling footstep sounds conditioned on discrete contact surfaces, while Liu et al. [159] focused on conditioning on different types of audio.

To be more effective in utilizing available audio datasets, DarkGAN [84] uses knowledge distillation (KD) to extract categorical information from large pre-trained classification models [88]. The classification model is able to output the associated probabilities (termed as soft labels) for each class, which could be utilized as conditioning information to guide the sound synthesis. Another relevant work by Gupta et al [258] also uses the soft labels returned from an audio classifier as the conditioning label, but the ground-truth labels are also supplied with their dataset, resulting in a supervised learning approach. Although such approaches achieved higher audio qualities without requiring large amounts of labelled datasets, the discrete nature of the conditioning method still poses drawbacks, including limited expressiveness [259, 260], lack of continuity [261], and failure to capture the hierarchical semantic relationships between classes [262].

Conditioning on text embeddings, on the other hand, provides a simple yet effective control with human languages. AudioLDM [69], for example, is capable of synthesizing intricate audio waveforms agnostic to sound types given input texts by training in a self-supervised fashion, which requires less label data. Nevertheless, text-to-audio synthesis models still require large amounts of audio datasets with descriptive labels in order to achieve reasonable performance. Additionally, texts themselves are also limited by the expressiveness in describing the subtle differences among different sounds.

Apart from conditioning on discrete labels, it is possible to use continuous vectors [263]

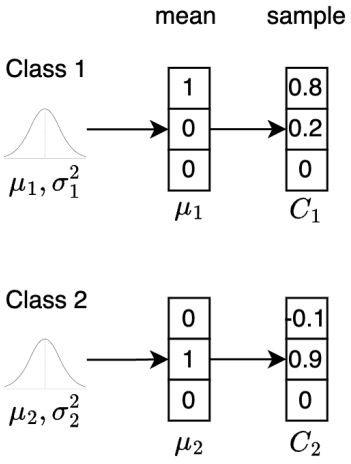
as conditioning information but it may require a specific training setup. Regression labels (continuous, such as ages and angles in the image) provide a smooth and expressive representation that can help capture hierarchical [264] and nuanced variations across different classes [265]. CcGAN [264] achieves this by assigning weights using a Gaussian kernel based on the distance from the target label, allowing the model to handle sparse data more effectively. The proposed hard vicinal and soft vicinal discriminator loss allows the generative model to smooth the transition across continuous labels. It has been demonstrated that by introducing uncertainty, label smoothing [266] helps generative models improve their training stability and robustness. However, in the domain of audio, it is generally very difficult to obtain regression labels, and the number of high-quality datasets for sound effects pales in comparison with images. This makes applying a similar conditioning method to sound effects modelling difficult.

In this research, I aim to study sound effects modelling under a GAN architecture by conditioning our generator on probabilistic soft labels. Different from CcGAN [264], I wish to blur the boundaries between discrete classes without requiring regression labels. To achieve this, we introduce an implicit conditioning method by manipulating discrete labels into continuous probabilistic vectors as conditioning. In Section 7.2, I will elaborate on the proposed method by integrating it into a WGAN [113] model for Mel-spectrogram generation. In Section7.3 and Section7.4, we discuss the training details and the experiments as well as the evaluation metrics we used. We show the control effectiveness of the soft-labelling approach and propose two metrics to understand and evaluate the controllability of our conditioning approach using a pre-trained audio classifier. The results are shown in Section 7.5.

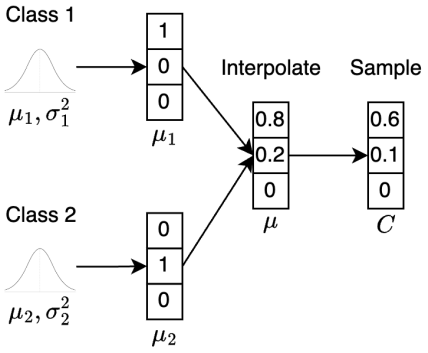
7.2 Methodology

7.2.1 Proposed conditioning method

We start by formulating our research goal. For a generator model, G, whose task is to output a two-dimensional Mel-spectrogram, M, we wish to condition the model on a continuous



(A) Sampled vectors as conditioning labels.



(B) Interpolate between two classes.

FIGURE 7.2. On the left we show how our conditioning labels are obtained by sampling from a Gaussian distribution parameterized by the mean and log variances learned from the encoder classifier. The conditioning labels C_1, C_2 are passed into the generator for conditional generation. On the right, we show how to interpolate between two classes to obtain the conditioning labels. Imagine a two-class audio dataset. We could obtain its 'latent representation' via an encoder under an auto-encoder architecture, which outputs the mean and variance of the input. By special configuration of the architecture, we could train the encoder to output specific mean values, such as 0 or 1, which we could use to associate meaning to corresponding classes. Vectors are then sampled from distributions parameterized by the means and variances output from the encoder and could be used as soft labels to guide sound generation. In this way, each sampled vector is actually associated with probabilistic meanings that relate to particular audio classes. Vectors sampled with a mean of 1 in the first channel would have a higher chance of outputting sounds that correspond to the timbre of class 1, while vectors sampled with a mean of 1 in the second channel would have a higher chance of outputting sounds that correlates to the timbre of class 2.

vector, C, which encodes the acoustic attributes of input sounds across different categories. To this end, we replace the one-hot vectors with a random variable sampled from a Gaussian distribution. Given a conditional vector C_s, sampled from a Gaussian distribution parameterized by mean and variance, our modelling objective is to estimate the conditional probability distribution $p(x|C_s)$ of the target data x. This can be formulated as:

$$p(x|C_s) = \int p(x|z, C_s)\, p(z|C_s)\, dz \tag{7.1}$$

where:

- x represents the target data we aim to model,
- C_s is the conditioning vector, providing context or attributes that the target data x should adhere to, and
  $C_s \sim \mathcal{N}(\mu, \sigma^2)$, where μ and σ are learned from a classifier.
- z is a latent variable capturing aspects of x not specified by C_s,
- $p(z|C_s)$ represents the distribution of latent variables conditioned on C_s,
- $p(x|z, C_s)$ denotes the likelihood of x given both z and C_s.

Specifically, we first employ a CNN-based encoder classifier as shown in Figure 7.3 to extract acoustical information from M. The encoder classifier learns the spectral-temporal information and is expected to output class probabilities of the inputs. Instead of using the probabilities directly as conditioning labels, we wish to obtain a control space that allows us to interpolate the sound characteristics smoothly. To this end, we transform the output logits from the encoder classifier into mean and variance variables, μ and σ^2, respectively, which are used for re-parameterization to enable gradient flow. Similar to a VAE[18], we sample a vector from standard Gaussian distribution as $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ to obtain the sampled vector $C_s = \mu + \epsilon \cdot \sigma^2$.

By introducing uncertainty and noise through the sampling process, we wish to construct a more continuous space of categorical information among different classes of sounds. Instead of relying on a simple concatenation process, we integrate the conditioning information via a feature-wise linear modulation (FiLM) [267] operation that is applied to both C_s and z. FiLM applies an affine transformation to the intermediate features of a neural network using scaling and shifting operations, mathematically represented as $FiLM(z, C_s) = f_\gamma(C_s) \cdot z + f_\beta(C_s)$, where f_γ and f_β are neural networks (typically MLPs) that map the conditioning input C_s to the latent space dimension. This technique allows for precise control over how external

information influences network activation values, leading to accurate and context-aware outputs.

7.2.2 Model architecture

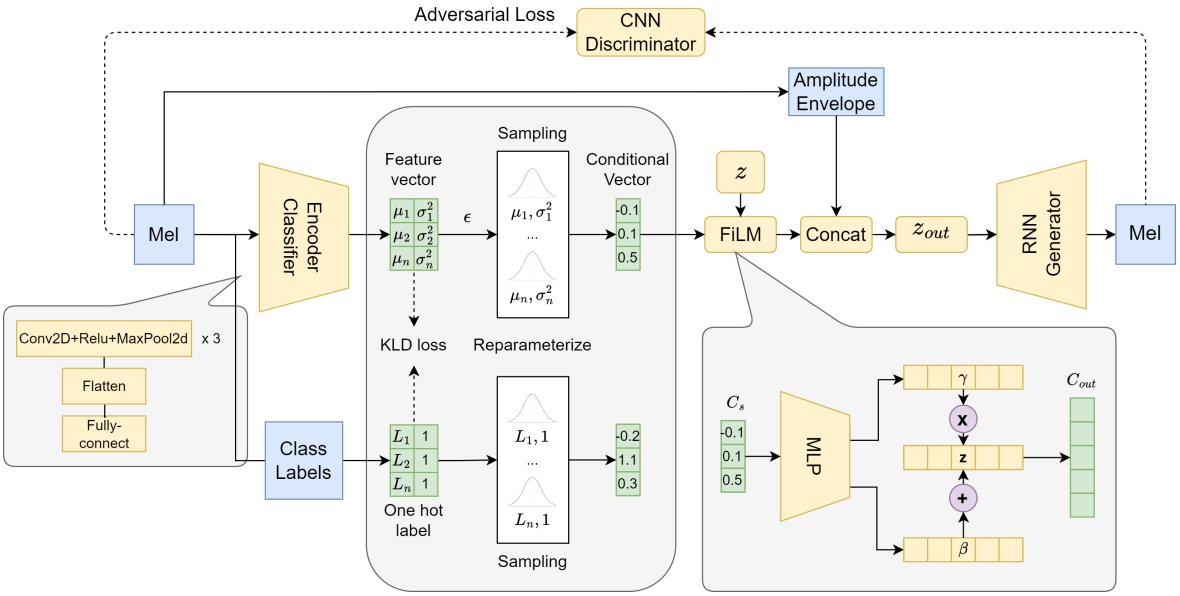


FIGURE 7.3. Proposed model architecture. The model is composed of a CNN encoder classifier, an RNN generator, and a CNN discriminator. Yellow boxes are part of the neural networks. Blue boxes indicate training inputs and outputs, while greens represent explicit variables.

To demonstrate how our conditioning method performs, we integrate our conditioning method into a GAN network for Mel-spectrogram generation. Mel-spectrogram correlates well with human perception and has received much attention in neural audio synthesis [105, 268]. It serves as an interpretable intermediate representation that could be effectively learned with small amounts of data. With the advancement of neural vocoders [96, 95], we can usually get decent audio reconstructions from Mel-spectrograms.

We denote our model as ICGAN (implicit conditioning GAN) with the entire model architecture as shown in Figure 7.3. In addition to conditioning the generator on the class labels, we also condition it on the extracted amplitude envelope from the Mel-spectrograms. Amplitude envelopes serve as a regulator that guides the generator to synthesize spectrograms with similar envelopes. This enables us to synthesize audio in a frame-level manner by inputting a

guiding envelope during the inference. Additionally, with the help of amplitude information, the generator model may achieve higher synthesis quality, which is discussed in Section 7.5. To obtain the amplitude envelope from the Mel-spectrograms, we sum and average the power amplitude across all frequency bins to get the frame-level amplitude envelope A:

$$A(t) = \frac{1}{F}\sum_{f=1}^{F} P(f,t) \ , \tag{7.2}$$

where $P(f,t)$ is the power at frequency bin f and time frame t for all frequency bins $f \in [1, F]$. In this way, the generator is guided by the amplitude information as well as the class-conditioning vectors. To synthesize the 2D spectrograms, we use the same RNN-based generator to model Mel-spectrograms as [161]. The generator is a three-layer GRU unit with an internal size of 512 followed by a linear layer. It sequentially transforms the input information to Mel-spectrograms with the same sequence length.

As for the encoder classifier and the discriminator, we adopt a simple LeNet-5 [269] as shown in Figure 7.3. Both the encoder classifier and discriminator models employ 2D CNN layers to extract spatial information from the input spectrograms. Once the model is trained, we transform the generated Mel spectrogram to audio waveforms using a pre-trained neural vocoder [95]. The HifiGAN vocoder was trained on the Audioset[185] dataset, which contains millions of labelled sound events. It is expected to be able to convert Mel-spectrograms back to audio waveforms agnostic to the audio source. Our complete model implementation can be accessed at the companion website[1].

7.2.3 Loss functions

Instead of providing explicit labels as input into the generator, we employ continuous categorical labels as conditioning vectors learned from the classifier. To facilitate the learning of the conditioning vectors, we force the re-parameterized Gaussian distribution to be similar to a Gaussian distribution formulated based on discrete class labels with a ground-truth mean and

[1] https://github.com/Reinliu/ICGAN

variance pair (refer to Fig. 7.3). We use the Kullback-Leibler divergence (KLD) to measure the similarity of the two distributions. We compute a regularization loss L_{reg} that incorporates the KLD using the one-hot vector label L, as well as the mean μ_n and variance σ_n^2 vectors obtained from the encoder classifier:

$$\mathcal{L}_{reg} = -\frac{1}{2}\sum_{n=1}^{N}\left(1 + \log(\sigma_n^2) - (\mu_n - L_n)^2 - \sigma_n^2\right), \tag{7.3}$$

where N is the dimension of the label space, which corresponds to the total number of classes, and one hot values L_n are used to indicate the target distribution mean. Apart from the regularization loss, we adopt the Wasserstein GAN (WGAN) [113] loss framework for the discriminator, mirroring its proven approach to evaluate the discrepancy between real and generated data distributions. The generator loss is shown below:

$$L_G = -\mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] + \mathcal{L}_{reg} \tag{7.4}$$

where $\mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[\cdot]$ denotes the expectation over samples $\tilde{x}$ generated by the generator's distribution $\mathbb{P}_g$ and $D(\tilde{x})$ represents the discriminator's (critic's) score for the generated samples. Notice that our generator loss also includes the regularization loss as illustrated above. In this way, the generator learns to accommodate changes associated with the classifier updating its output. This allows the generator and encoder to update their weights simultaneously within each batch, helping to stabilize the training and make the results more reliable.

With regard to the discriminator, we enhance training stability and enforce the Lipschitz constraint essential for optimal discriminator behaviour by integrating the gradient penalty mechanism that is a hallmark of the WGAN-GP model [114] as shown below:

$$\begin{aligned} L_D = {} & \mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] - \mathbb{E}_{x\sim\mathbb{P}_r}[D(x)] \\ & + \lambda \cdot \mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}\left[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2\right] \end{aligned} \tag{7.5}$$

- $\mathbb{E}_{x\sim\mathbb{P}_r}[D(x)]$ is the average score that the discriminator gives to real samples.
- $\lambda \cdot \mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}\left[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2\right]$ represents the gradient penalty. This term enforces a Lipschitz constraint and is crucial for the model's stability. The penalty is applied to the gradients of the discriminator's scores with respect to interpolated samples between real and generated samples, pushing these gradients to have a norm of 1. We set $\lambda = 10$ in our experiments.

7.3 Experiments

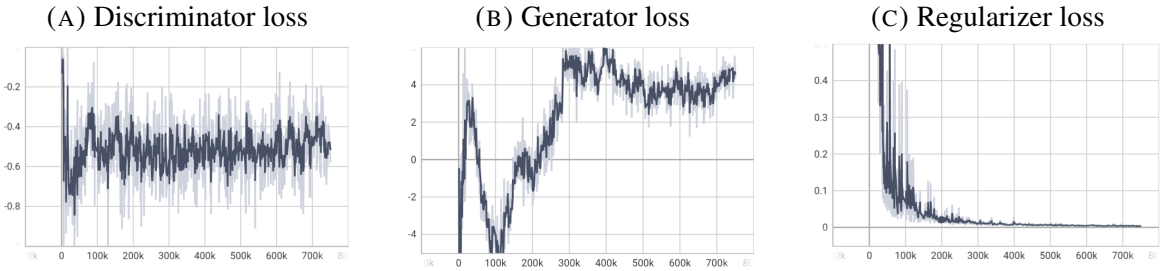


FIGURE 7.4. Visualization of the loss trend in our ICGAN model. We generally see a convergence after 300k iterations when all three losses are shown stable. The regularizer loss quickly converges while discriminator and generator loss oscillates around -0.5 and 4 respectively.

7.3.1 Dataset

As we are mainly interested in modelling sound effects, we meticulously construct an impact-sound-based audio dataset with sounds selected from the Boom Library[2] and BBC Sound Library [224]. Because of copyright issues, these datasets are not open to the public. Our dataset consists of three kinds of impact sounds: footsteps, gunshots, and hits. Each of these sound types is also comprised of several classes dependent on the labelling of such sounds. For example, footsteps contain 'Concrete', 'Gravel', 'Leaves', etc., while hits contain 'Punch face', 'Hit bag', 'Hit bone', etc.

[2] https://www.boomlibrary.com/

There are 6 subcategories of footstep sounds, 10 subcategories of gunshot sounds, and 9 categories of hit sounds. Each of the subcategories contains 100-200 sounds. In total, there are 4817 sounds and we split our dataset into 4335 (90%) for training and 482 (10%) for testing. Each of the sounds is sampled at 16 kHz and has a length of 4 seconds. We pre-process each sound to extract its Mel-spectrogram with a hop length of 64, and FFT size of 1024. All of the Mel-spectrograms have a length of 400 frames and 64 frequency bins. This setting corresponds to a pre-trained HiFiGAN vocoder [69] trained on the Audioset [185] dataset, allowing us to easily transform the Mel-spectrograms back to audio waveforms.

7.3.2 Training

We trained our model with the aforementioned datasets of three categories of sounds (footstep, gunshot, hits) for 10,000 epochs on an RTX 3080. Each model is conditioned with the variations within each category (eg. concrete, gravel, snow for the footstep category). We use an Adam optimizer [257] with a learning rate of $l = 0.0001$, $\beta_1 = 0.5$, $\beta_2 = 0.999$. During training, the discriminator loss stabilizes between -0.4 and -0.7 after about 100 k iterations indicating convergence. The regularization loss converges more slowly and requires approximately 300 k iterations as shown in Figure 7.4. As the regularization loss is added to the generator loss, the generator loss oscillates until the regularization loss converges after the 300 k iterations. Eventually, after the regularization loss stabilizes, the generator loss converges to a value of around four.

In addition to the in-class sounds (sounds within the same category), we briefly test the inference capability of our model on cross-class sounds. Therefore, we further trained our model on a sound effect dataset DCASE2023 [227] and conditioned it on the 7 categories such as motor engine, dog barks, and rain. We show the performance of the conditioning method in Section 7.5.

7.4 Evaluation Methods

7.4.1 Interpolation between two classes

Once the model is fully trained, we can abandon both the encoder and the discriminator. Instead of extracting information from a sound's Mel spectrogram, we can define an arbitrary conditioning vector C_s. Entries in the C_s vector close to unity indicate strong correspondence to a particular class within the sound type and values close to zero indicate no correspondence. In order to explore the influence of the conditioning vector, we interpolate 100 points between two target classes within a sound type. In other words, the values for one class increase from zero to one, while the values for the other class decrease from one to zero.

In order to evaluate the 'class' of the sound we use a separate pre-trained classifier that is unrelated to the sound generation (described in the next section). In Figure .10, we show four example pairs for each of the three sound categories. Each sound category was trained using a separate model with its own dataset. We randomly picked two classes within each category (eg, Footstep: concrete and gravel). Considering Figure .10, we generally see smooth transitions from one class to another. Nonetheless, the sharpness of the transition seems to vary as does the completeness of the transition.

7.4.2 Proposed metric for control effectiveness

To measure the effectiveness of our conditioning method, we incorporate a pre-trained classifier that indicates the probability for each class of sound. This essentially links the conditioning information with the generated output. To this end, we use PANNs [88], a large-scale audio classification model trained on vast amounts of sounds [185] with 14 CNN layers. It was reported to achieve great acoustic event detection and audio tagging performance agnostic to sound types. Based on the pre-trained PANNs model, we fine-tuned and trained it on our dataset until it converged and reached an evaluation score of 79.5%.

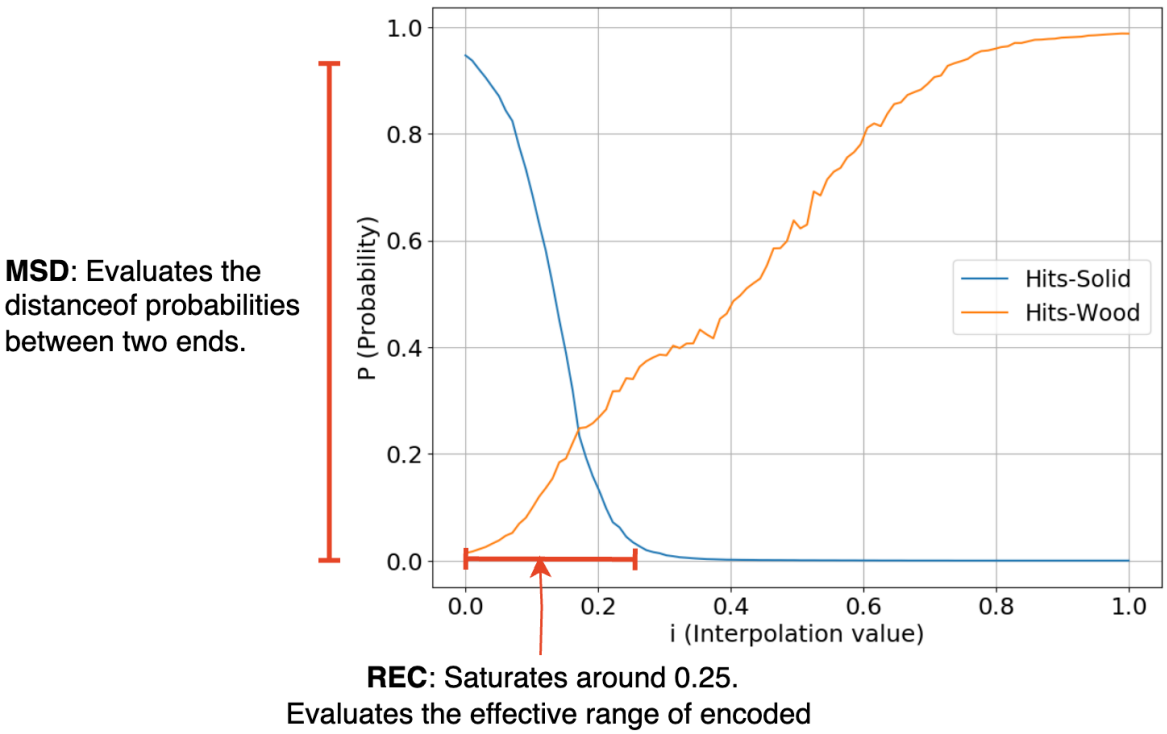


FIGURE 7.5. Interpretation about MSD and REC. The REC measures the horizontal distance of the effective control range where it reaches its saturation. The MSD measures the vertical maximum difference of returned probabilities between two ends.

We propose two evaluation metrics to evaluate our conditioning method: Maximum separation distance (MSD) and range of effective control (REC) as shown in Figure 7.5. The MSD is designed to quantify the breadth of the model's conditioning space. It does so by measuring the maximal extent to which the model's output can be varied by adjusting the conditioning vector to the limits indicating only one of the two target classes. In other words, ideally the conditioning vector can shift the generated sound completely from one class to the other. Practically, it can be incomplete. We measure the MSD score as the average of the total range of the softmax probability score for each of the two target classes:

$$\text{MSD} = (P(j)_{max} - P(j)_{min} + P(k)_{max} - P(k)_{min})/2, \tag{7.6}$$

where j, k are two interpolation variables with a range varying from zero to one.

The range of effective control (REC) is a metric that focuses on the practical operational range of the conditioning vector to change the model's output. It is calculated by determining the interval within which changes to the conditioning vector lead to significant changes in the output. More specifically, in Figure .10, we see the curves can saturate as the interpolation values are varied towards the ends of the interpolation range. This is to say, the slope of the curves can become flat towards the ends. Thus, we can define a threshold value for the absolute value or magnitude of the slope of the curve. If the magnitude of the slope is too small, less than some threshold, θ, we say the response has saturated. We define the REC score as shown in equation 7.7, $i \in [0, 1]$.

$$\text{REC} = i_{\text{end}} - i_{\text{start}} \tag{7.7}$$

where:

- i_{start} is the first interpolation value where $\left|\frac{\Delta P(i)}{\Delta i}\right| > \theta$,
- i_{end} is the last interpolation value within the specified range where $\left|\frac{\Delta P(i)}{\Delta i}\right| > \theta$.

The REC score measures the segment of the output response curve for which the rate of change in the class probability is greater than the threshold value, θ. We compute the REC score for all pairs of classes under consideration, to explore how effectively the interpolation between classes controls the timbre attributes of the sounds.

7.4.3 Evaluation setup

In order to evaluate the synthesis performance of our model, we need to compare sounds generated by the model with real sounds. Therefore, we use our test dataset (refer to Section 7.3.1) to provide reference sounds. We extract the amplitude envelopes of the reference sounds and use a conditioning vector obtained by sampling a Gaussian model with a one-hot ground truth class label vector corresponding to the class of the reference sound as the mean for the Gaussian model. We provide the amplitude envelope and conditioning vector as inputs to our ICGAN model and thus obtain one synthesized sound for each reference sound.

We compare our ICGAN model with LTS (Latent timbre synthesis) [270], which enables a smooth transition from one sound to another by interpolating the VAE latent space. We denote ICGAN as the test in which we interpolated only conditioning vectors. Since LTS can only interpolate the latent space from one target sound to another and it does not take in any amplitude information as conditioning, for a fair comparison, we interpolated the amplitudes using ICGAN between two target sounds in the same way as LTS and named it ICGAN-w (with amplitude interpolation).

For the ablation study, we have also retrained our model by completely removing the amplitude envelope from the conditioning and named this ICGAN-n (no amplitude information). Finally, We added a comparison with the classical conditioning method as [99], namely, by inputting one-hot representations of the discrete labels into the generator and discriminator networks. We denote this model as conditional GAN (CGAN). Please note that we also interpolated values in the conditioning space of CGAN even though, during training, it was only accepting one-hot vectors. The reason for this is to test whether the model is able to extrapolate smooth transitions between discrete values.

All results are shown in Table 7.1. We evaluate the quality of the synthesis by comparing the similarity between the synthesized sounds and the reference sounds. We select a range of evaluation metrics including Frechet Audio Distance (FAD) [210], Frechet Inception Distance (FID) [211], and Log-Spectral Distance (LSD). Both FAD and FID are common metrics for evaluating the quality of generated data (audio and images) in generative models. They rely on pre-trained classifiers to extract features from both real and generated samples.

The distance scores are then reported by computing the mean and covariance of these feature sets. This statistical approach captures not just the sound quality, but also the diversity within the datasets. For FAD, we use the VGGish [162] classifier with a sample rate of 16kHz, and for FID, we employ the InceptionV3 [271] classifier. Additionally, we compute the pairwise log spectral distance between the generated Mel-spectrogram and the reference Mel-spectrogram. We report the final LSD as the average score from each category.

7.5 Results

7.5.1 Conditioning performance

Since we have several classes of sounds for each sound type or category, there are several possible pairs of classes with which to interpolate between. We refer to these pairs of classes as interpolation pairs. For the purposes of showing results, we average the results across a sample of interpolation pairs. The average results obtained for the MSD and REC are shown in Table 7.1. When computing the REC, we use a threshold value given by $\theta = 1e^{-2}$.

For the MSD score, we found that CGAN achieves the highest score. This makes sense because it was hard conditioned on discrete values between two ends, namely 0 and 1. The classification could successfully tell the class on the two ends. On the contrary, REC reflects that CGAN gets much lower scores because the transition between two ends is rather discrete and drastic. Our ICGAN-w, in which we interpolated the amplitudes together with the conditioning space, achieves the highest REC overall and performs second best in MSD. This shows the potential of our method, as it no longer requires interpolating in the uninterpretable latent space to shift from one sound to another. Although interpolating the conditioning space only did not yield the best performance, we nonetheless found it useful as it provides a straightforward and interpretable control over the sound categories, which could not be easily achieved using other models relying on extensive labels.

Additionally, the results for the REC score indicate that the effective range of conditioning varies across different sound types. We found the Hits category obtains the highest REC overall, whereas the DCASE dataset attains the lowest. We hypothesize that this can be attributed to the degree of variance within different sound classes. To test this, we extract and average the MFCC features for each sub-category, then compute and normalize the pairwise Euclidean distances between categories, and finally get the averaged similarity scores for each dataset. We found that our Hits dataset did have higher variances among different sub-categories, with a normalized similarity score of 0.568, which is higher than Guns 0.451,

Footsteps 0.316 and DCASE 0.342. The high degree of variations among different sub-categories could enable the encoder classifier to efficiently learn the intricate differences and thereby output meaningful and smooth values, which likely results in the smooth transitions of different classes. However, how exactly dataset variations could contribute to different control affordances needs further investigation.

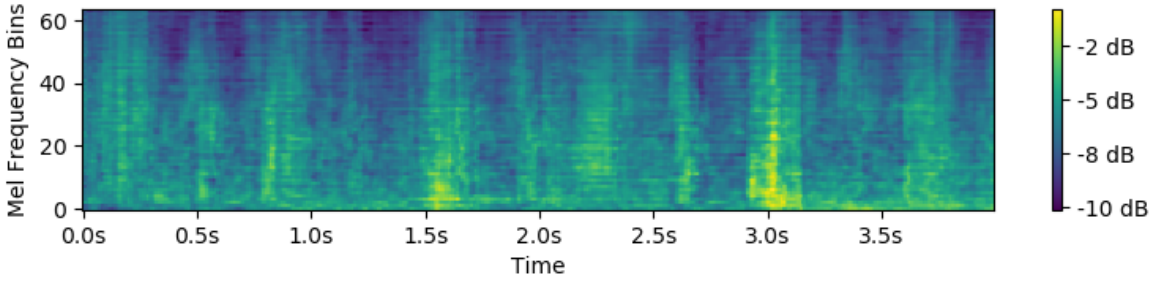
7.5.2 Synthesis performance

In Table 7.1, we found that our model excels in statistical similarity metrics reflected in FAD and FID scores, indicating a high degree of performance in the quality and diversity of the generated data. However, from the log spectral distances which compare the pairwise signals directly, we realized the explicit conditioning method achieves higher accuracy in the spectrogram reconstruction. This is because our implicit conditioning method introduces uncertainty through sampling, resulting in lower accuracy for reconstructing signals with definitive labels.

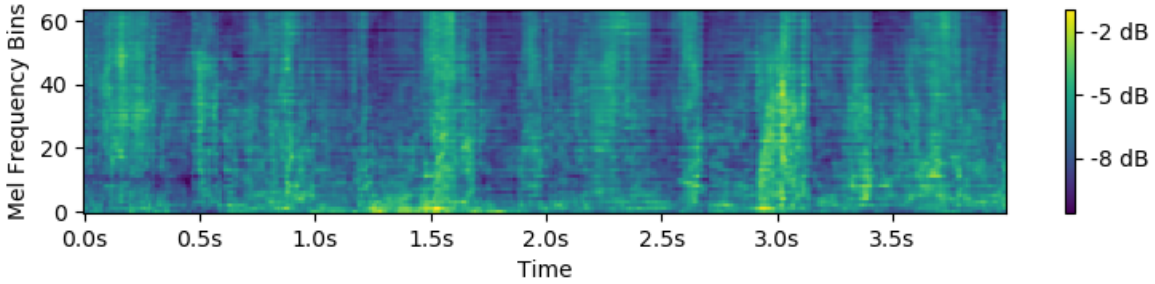
On the contrary, this also brings higher sample diversities reflected in the statistical measures. We also noticed an increase in dissimilarity between the target and generated data trained by the DCASE Foley dataset. It becomes more difficult for both models to reconstruct the signals when the conditioned data vary more. We have also observed that after removing the extracted amplitudes, the audio quality was reduced significantly. We believe the amplitude information provides important guidance for the generator for synthesis.

7.5.2.1 In-domain Sound Generation

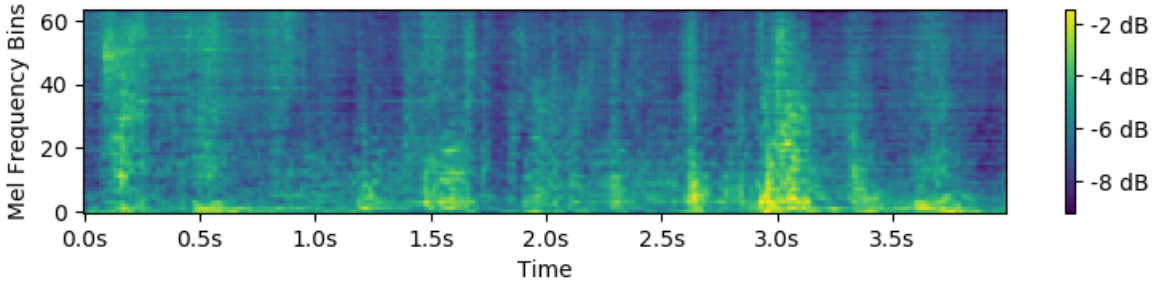
In Figure 7.6, we show the synthesized Mel spectrograms corresponding to different classes within the Foodsteps sound category: concrete, leaves, and gravel. Note that all three sounds were generated using the same amplitude envelope, and the spectrograms exhibited similar temporal changes. The amplitude conditioning helps the model maintain the overall contours of the spectrogram while offering controllability varying timbral characteristics.



(A) Footstep-concrete



(B) Footstep-leaves



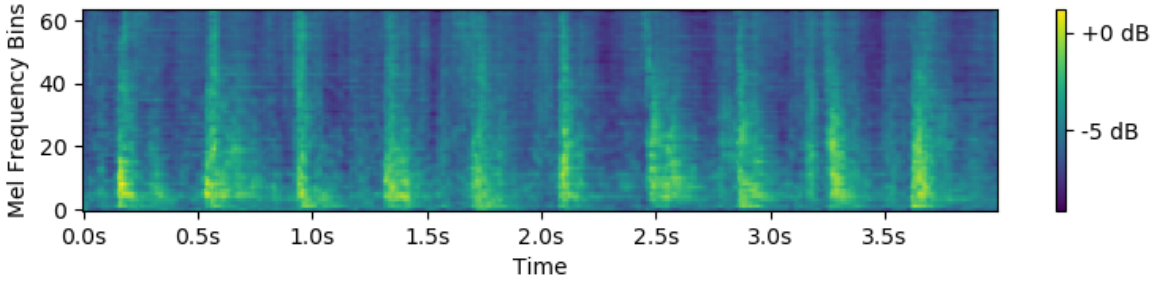
(C) Footstep-gravel

FIGURE 7.6. *Generated in-class sounds. The models are trained in the same footsteps category with different variations denoted by its property.*

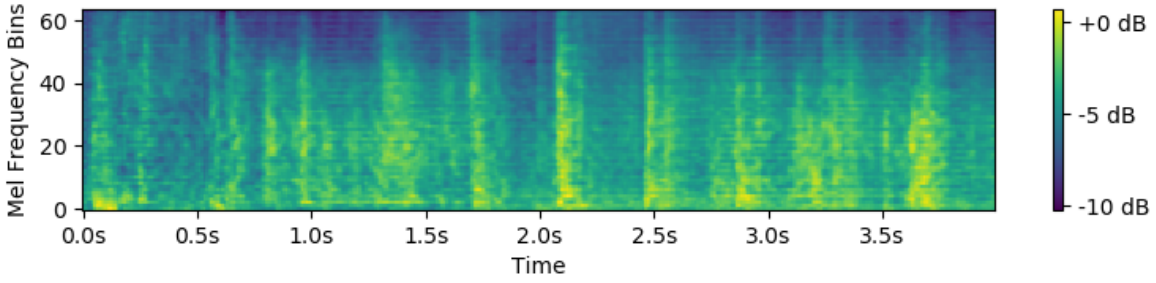
7.5.2.2 Out-of-domain Sound Generation

In addition to conditioning on in-class sounds, our model could also be trained to synthesize out-of-domain sounds. In Figure 7.7, show an example of footstep sounds 7.7a, gunshot sounds 7.7b, and dogbark sounds 7.7c. The sounds are generated from the ICGAN model trained on the DCASE Foley 2023 Dataset [227].

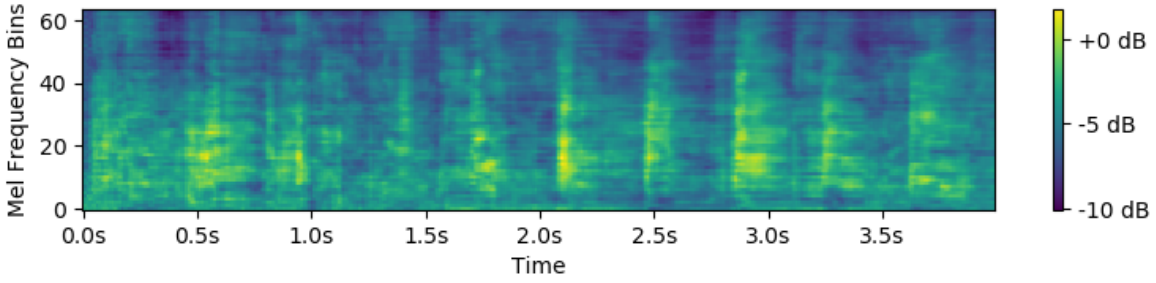
This time, we manually set the conditioning vectors that correspond to three channels of the sound categories: footstep, gunshot, and dogbark. The conditioning space now represents



(A) Footstep



(B) Gunshot



(C) Dogbark

FIGURE 7.7. *Cross-domain sound visualization. We trained the single model conditioned on different classes using the DCASE [227] dataset.*

a much greater variation in spectral and timbral quality, because they are trained on sounds coming from different categories. We observed that the model is able to output sounds that follow similar timbres of these categories. (The complete demonstration of cross-domain sounds could be listened to in our accompaniment website [3].)

However, while the spectral characteristics vary among the three categories of sounds, the sound generator often fails to output high-fidelity sounds. We noticed a lot of glitches in the tones generated when we set the conditioning vectors high (eg. larger than 1). This can

[3] https://reinliu.github.io/ICGAN-Paper/

likely be attributed to the limited variance given our dataset. The model is not well-trained when there are gaps or discrepancies across the sound categories. Therefore, to further help improve the generation quality of out-of-domain sounds, future work could include exploring larger datasets with more variances in the temporal and spectral information.

7.6 Discussion and Conclusion

In this research, I proposed a novel conditioning method that translates discrete labels into a continuous probabilistic labelling space. I have applied this implicit conditioning method to the domain of neural audio synthesis and explored model performance in terms of the effectiveness of the conditioning vector in controlling the sound class and the audio quality of the generated sounds. I primarily focused on in-domain variations in sound effects but briefly considered cross-domain sound effects. Our approach shows promise when compared with conventional conditioning methods, but requires more development.

I have shown that the conditioning vector can effectively interpolate between sound classes and enable a smooth transition in the timbre space without relying on excessive labels. I have also observed that the control effectiveness seems to relate to the variations of a dataset, but I will leave this question to future investigation. I hope our proposed conditioning approach sparks more creativity in the sound design and synthesis field and allows for interesting future explorations in the continuous conditioning space, such as interpolation involving more than two dimensions and also mixing the features of various target sounds. Future work could include incorporating a training mechanism that encourages a more consistent and controllable conditioning space.

Dataset	Model	Control		Audio Quality		
		MSD↑	REC↑	FAD↓	FID↓	LSD↓
Footsteps Similarity: 0.316	ICGAN	0.683	0.392	**2.85**	**64.61**	0.119
	ICGAN-w	0.807	**0.696**	/	/	/
	ICGAN-n	0.744	0.610	16.66	163.4	0.118
	CGAN	**0.851**	0.022	4.90	102.5	**0.086**
	LTS	0.759	0.667	8.86	145.6	0.125
Guns Similarity: 0.451	ICGAN	0.637	0.579	**2.64**	**60.44**	0.235
	ICGAN-w	0.733	**0.721**	/	/	/
	ICGAN-n	0.722	0.629	7.09	177.4	0.197
	CGAN	**0.841**	0.018	2.88	76.43	**0.187**
	LTS	0.637	0.711	4.97	126.9	0.289
Hits Similarity: 0.568	ICGAN	0.647	0.863	**1.27**	**60.58**	0.136
	ICGAN-w	0.795	0.903	/	/	/
	ICGAN-n	0.773	0.807	5.19	110.6	0.146
	CGAN	**0.911**	0.011	2.43	95.57	**0.124**
	LTS	0.677	**0.936**	2.93	97.87	0.176
DCASE Similarity: 0.342	ICGAN	0.509	0.210	**6.956**	**79.60**	0.272
	ICGAN-w	0.727	**0.569**	/	/	/
	ICGAN-n	0.641	0.513	16.49	185.5	0.292
	CGAN	**0.818**	0.026	10.21	108.6	**0.246**
	LTS	0.693	0.512	7.67	131.4	0.321

TABLE 7.1. Comparison of model synthesis performance and control affordances. ICGAN is the proposed model as illustrated in 7.3. ICGAN refers to our proposed method by interpolating only conditioning space. ICGAN-w is the same ICGAN model but we also interpolated the extracted amplitudes from two target sounds. Therefore the audio quality is the same as ICGAN. ICGAN-n is a re-trained model that completely removes amplitude information. CGAN denotes the conditional GAN using concatenation. LTS refers to 'Latent Timbre Synthesis' [270] which does latent space interpolation between two target sounds. Up-arrows indicate the higher the score the better, and vice versa. To understand whether different datasets might have an effect on the results of our experiments, we extracted and averaged the MFCC features for each sub-category, computed and normalized the pairwise Euclidean distances between categories, and derived averaged similarity scores. The computed similarity scores are shown on the first column of datasets.

CHAPTER 8

Simi-SFX: A similarity-based conditioning method for controllable sound effect synthesis

---

8.1 Introduction

Timbre is an audio term used to describe the unique acoustic characteristics of sounds, often referred to as "the colour or tone of a sound"[272]. In the realm of sound effects—non-musical and non-speech sounds—timbre serves as a critical audio descriptor, enabling listeners to identify sounds and contributing significantly to the realism and immersive quality of audio experiences[273]. For instance, the sound of footsteps can vary dramatically based on the surface being walked upon. Materials such as wood, gravel, or metal each produce distinct acoustic signatures that reflect their interaction with the environment [44, 99, 274].

The ability to accurately and meaningfully interpolate between these timbres is essential for sound designers aiming to create dynamic and contextually rich soundscapes that mirror varied environmental interactions [275]. This capability is particularly important in industries such as film, gaming, and virtual reality, where nuanced sound design enhances narrative depth and user engagement, ultimately enriching the overall auditory experience [15].

Generating sound effects with controllable variations has been a persistent challenge in audio synthesis [276, 131, 277]. Traditional approaches often rely on sophisticated physical models, which demand a deep understanding of signal processing parameters and algorithms. To streamline this process, physically driven neural audio synthesis methods have emerged, leveraging neural networks to reconstruct physical priors from sound, thereby enabling control over the synthesis [278].

While these methods are powerful, they often lack intuitive controls, making them less accessible to non-experts. Additionally, the perceptual subtleties of audio DSP parameters are not always intuitive, posing challenges even for experienced users. The limited availability of diverse, labelled datasets further hinders the development of flexible and broadly applicable sound synthesis systems, underscoring the need for more user-friendly and adaptable approaches.

In the era of neural audio synthesis, generative models are often conditioned on various modalities, such as class labels [159, 100], textual descriptions [69, 83], and visual inputs [197, 101], to guide sound production. A detailed overview of conditioning labels is shown in Figure 7.1. While these approaches enable diverse forms of control, they frequently struggle to capture the nuanced timbral differences inherent in distinct environmental contexts. This limitation arises from the reliance on descriptive, hand-crafted labels, which are often insufficient for achieving fine-grained control over sound expression.

The Differentiable Digital Signal Processing (DDSP) framework [13] offers a promising alternative by conditioning sound synthesis on detailed features, enabling rapid training and high-quality audio generation. However, despite its strengths, DDSP faces challenges in certain areas, such as morphing between timbres and effectively modelling rigid-body impact sounds—critical components of sound Foley. Addressing these limitations is essential for advancing the flexibility and realism of neural audio synthesis systems.

In this research, I pursue two primary objectives: (1) to generate diverse variations of sounds using a controllable guiding variable that modulates timbre, and (2) to synthesize high-quality inharmonic and impulsive sound effects, such as footsteps and gunshots. While most sound effect datasets are categorized by sound type, I aim to extract and leverage timbral information both within individual categories and across different sound types. This approach captures the similarity of a sound relative to all categories, preserving essential timbre cues that serve as valuable conditioning information.

To ensure high-quality sound synthesis while maintaining computational efficiency for both training and inference, I evaluated my method using a modified DDSP network specifically

tailored for synthesizing impulsive broadband sound effects. This framework enables precise control over timbre while meeting the demands of diverse and dynamic soundscapes.

8.2 Related Works

In the domain of neural audio synthesis, the latent space has been extensively explored to control and interpolate the styles of high-level features of the target audio [279, 280, 270]. StyleGAN [281] introduces an unsupervised approach that uses intermediate latent vectors to encode different characteristics of the synthesized output, allowing for style mixing. Although originally developed for visual data, this approach can be adapted to sound synthesis to capture and combine diverse stylistic attributes [282]. However, it remains unclear which specific high-level feature is controlled by each sub-latent component of the latent vector.

Given the complexity and intricacy of the latent space, additional effort is often required to establish meaningful control parameters that correspond to specific acoustic attributes. This can be achieved through techniques such as perceptual timbre remapping [172, 251] or adversarial fine-tuning [129]. Audio datasets, particularly in the sound effects domain, often lack detailed descriptive annotations. To address this, many research efforts have focused on extracting meaningful timbral information from audio samples using an analysis-synthesis approach. A straightforward method for achieving a degree of audio control involves using pre-processed audio features, such as pitch [14], loudness [13, 244], or spectral centroid [170], to guide synthesis. These models are typically trained in a supervised learning manner, which often reduces the requirement for extensive datasets.

However, the complexity and diversity of sound effects pose significant challenges. Many variations in sound effects cannot be effectively controlled using only DSP-extracted audio features. For instance, it is exceedingly difficult to transform the timbre of a footstep on a concrete floor into a footstep on snow or sand using solely DSP-based audio descriptors. While class-conditional sound synthesis methods [99, 100, 159] can capture subtle differences between categories, such as different types of footsteps, their capacity for generating new

timbres is limited by the discrete nature of one-hot vector representations commonly used in class-conditional models.

To better leverage the information available in sound datasets, Nistal et al. [84] proposed a knowledge distillation approach that enables semantic control over sound categories. This method uses 'soft labels' derived from a large audio tagging system to provide richer conditioning information. Furthermore, Liu et al [283] introduced an implicit conditioning method that employs continuous vectors sampled from a distribution to represent and condition the generator on each class, enabling smooth timbre interpolation across audio categories. However, since the conditioning vectors are sampled from a standard Gaussian distribution, they are unbounded, which can make them challenging to control effectively.

Unlike pure generative models, differentiable digital signal processing (DDSP) [13] integrates DSP components directly into an autoencoding network. By doing so, it leverages the structured priors provided by DSP principles, which streamline the training process, while also harnessing the neural network's ability to accurately fit the timbre of target sounds. However, the original DDSP employs a harmonic-plus-noise synthesizer, which is primarily designed for harmonic signals and thus inadequate for impulsive sounds, as shown in [244]. For many sound effects that include narrow-band components or transients, the original time-varying FIR filter struggles with the trade-off in time-frequency resolution.

Consequently, several approaches have been proposed to extend DDSP for modelling inharmonic and impulsive sound effects. Lundberg [133] explored vehicle engine noise modelling by integrating a transient synthesizer [134] into DDSP. Liu et al.[244] expanded this idea to model rigid-body impact sounds such as gunshots and footsteps. Serrano[284] focused specifically on modeling footsteps using the original time-varying FIR filter, conditioned on a Ground Reaction Force (GRF) curve. Meanwhile, Diaz et al.[56] developed a differentiable IIR filter bank for modelling rigid-body sounds conditioned on the shape and material of objects. For a more generalized approach, Barahona-Rıos[170] introduced a series of multi-rate filterbanks to synthesize source-agnostic sounds with fine-grained control.

8.3 Methods

8.3.1 Similarity Score

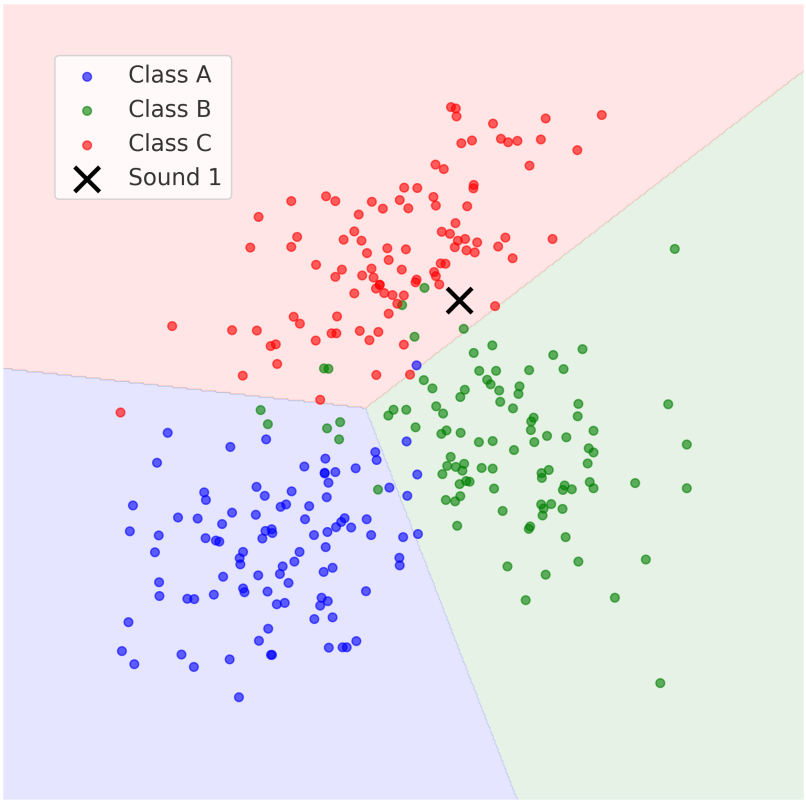


FIGURE 8.1. A visualization of a three-class embedding clusters. Notice that depending on the distance from each cluster's centroid, a sound might possess the sound characteristics of multiple classes. The closer the sound is from a centroid of a class, the higher the chance that it possesses the feature associated with that class.

Our method for conditioning the generative model draws inspiration from statistical evaluation metrics used in deep generative models. For instance, Frechet Audio Distance (FAD)[210] employs large pre-trained audio representation models, such as VGGish[162] and PANNs [88], to extract meaningful timbre information from both the reference audio dataset and the generated audio. FAD measures the Wasserstein-2 distance between two multivariate Gaussians—one representing real audio embeddings and the other representing generated audio embeddings.

To illustrate, consider a footstep dataset with three categories, each exhibiting a unique timbre, as shown in Figure 8.1. For each sound category, we compute embeddings for all audio samples, obtaining the mean and covariance for the category. To extract these embeddings, we utilize the Contrastive Language-Audio Pretraining (CLAP) model [89], which was trained on the extensive LAION-630k dataset containing natural sound effects. CLAP was shown to excel in modelling timbre similarities by Tian et al [285].

Given an audio signal, such as "Sound 1" in Figure 8.1, we can quantify its statistical similarity relative to each sound category. Inspired by anomaly detection techniques, where similarity scores are computed to identify outliers, we calculate distances using the Mahalanobis distance (MD) [286]. The Mahalanobis distance is a multivariate measure that assesses the distance between a point and a distribution. In our context, for an audio signal x and a group of audio samples characterized by mean μ and covariance matrix Σ, the Mahalanobis Distance (MD) is defined as:

$$MD(x) = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)}, \tag{8.1}$$

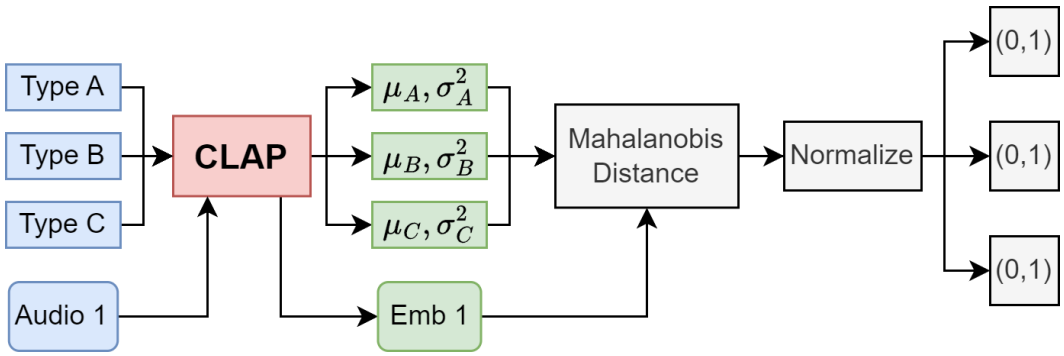


FIGURE 8.2. The process of similarity score extraction. We use Contrastive Language Audio Pretraining (CLAP) to extract embeddings of each sound. The embedding group of the audio class can be treated as a cluster of sounds with certain characteristics. For any audio, we can measure the Mahalanois distance relative to each class. The returned score is normalized across each channel to the range [0,1] for easy control.

Since the calculation of the MD requires the covariance matrix, Σ, to be invertible, we regularize Σ by adding a small value, a process known as diagonal loading: $\Sigma \leftarrow \Sigma + \epsilon I$,

where I is the identity matrix of the same dimension as Σ. This ensures the determinant is non-zero. The computed MD can be interpreted as a measure of similarity to a class (e.g., how similar a gunshot recording is to a balloon explosion).

For a dataset with three categories, the MD for a given sound can be computed relative to all three categories, resulting in a three-dimensional vector, as illustrated in 8.2. However, due to variations in intra-class and inter-class variances, the MD values can exhibit significant range disparities. To address this, we normalize the MD for each channel using min-max normalization, yielding the normalized Mahalanobis distance (MD_N):

$$MD_N = \frac{MD - MD_{\min}}{MD_{\max} - MD_{\min}}. \tag{8.2}$$

The normalized Mahalanobis Distance falls within the range [0, 1]. A lower MD indicates that the point is closer to the distribution centre of the corresponding class, signifying a stronger presence of the timbre associated with that sound category and vice versa. Using this approach, we derived a compact vector that encodes timbre information, which can be intuitively controlled via a fader or knob [173]. When applied to a generative model, this enables the creation of timbral variations.

8.3.2 Design of the transient model

Our transient modelling approach is founded on a simple yet effective principle: the inverse discrete cosine transform (IDCT) of a sinusoidal signal corresponds to an exponentially decaying impulse, as introduced in [134]. This technique has been successfully applied to DDSP in combination with harmonic-plus-noise synthesis to model various sounds, such as footsteps [244], vehicle engines [133], and piano sounds [287]. Our approach builds upon DDSP-SFX [244], where the decoder outputs frequency and amplitude values to synthesize sinusoids, which are then converted to impulses in the time domain using iDCT. Since multiple transient signals may be required within a single frame of sound, we synthesize $f/2$ sinusoids per frame, where, f denotes the frame length and acts as a sub-sampling rate. The synthesized sinusoids are then modulated by an amplitude vector, A, which is learned from the decoder.

As the decoder operates at a lower sampling rate, A is upsampled to the target sampling rate Fs to match the length of the entire signal. This upsampling is achieved using a combination of transposed 1-D convolution and ReLU activation. The complete transient model can be formally expressed as:

$$x[n] = \sum_{n=0}^{N-1} \sum_{k=1}^{K} A_{n,k} \, \text{IDCT} \left(\sin \left(2\pi F_{n,k} \frac{t}{f} \right) \right), \tag{8.3}$$

where:

- K is the number of sinusoids synthesized per frame and $K = f/2$ in our case.
- $A_{n,k}$ is the amplitude of the k-th sinusoid in the n-th frame.
- $F_{n,k}$ is the frequency of the k-th sinusoid in the n-th frame.
- IDCT is the inverse discrete cosine transform.
- t is the time index.
- f is the frame length.

8.3.3 Loss function for the transient model

As the FIR-filtered noise synthesizer operates at a lower sampling rate, it may inadequately capture the details of transient energies. NoisebandNet [170] addresses the trade-off between frequency and time resolution by designing a large-scale filter bank, albeit at the expense of increased computational cost. However,the equidistant placement of transients along the time axis, as proposed in [134, 133], does not fully resolve the issue of missing transient energies.

To better guide transient synthesis, we propose incorporating a dedicated loss function. First, we separate the percussive components of the training audio using harmonic-percussive source separation (HPSS) [256], with a kernel size of 31 and a margin of $(1, 3)$. Next, local peak estimation is performed to identify spectral peaks along the time axis, which are then converted into one-hot vectors, with non-peak regions set to 0. To ensure that synthesized impulses are placed only at these peak locations, we create a sparse waveform X_s by multiplying the one-hot peak vectors with the original waveform. Finally, an L_2 loss is computed directly between

the transient signal and the sparse peak waveform, providing a more targeted approach for transient synthesis.

8.3.4 Architecture

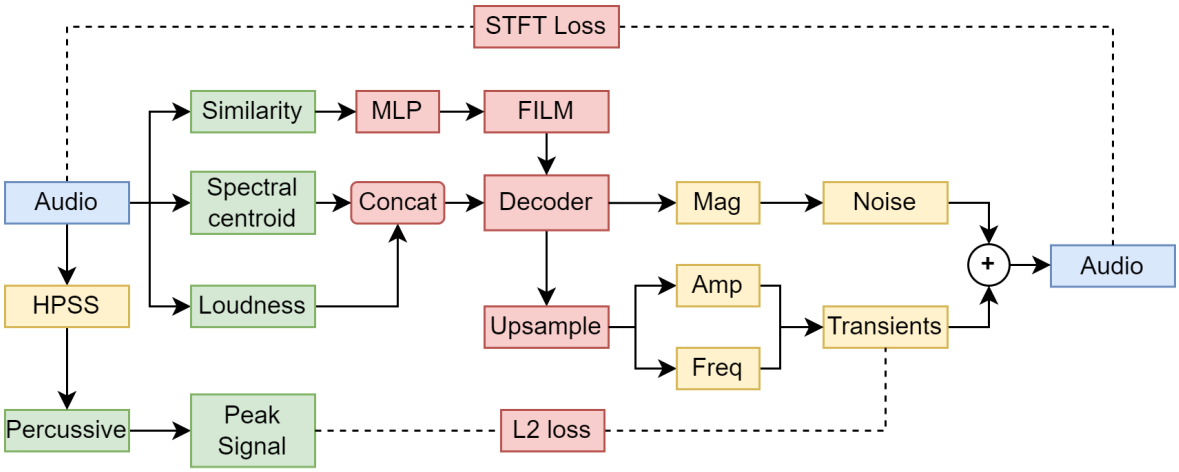


FIGURE 8.3. Overview of our proposed model architecture. Spectral centroid and loudness are extracted explicitly from input audio during the pre-processing stage. Similarity scores are also obtained by measuring the Mahalanobis distance between each audio embedding and group audio embedding of each class. A decoder is conditioned on these three features to generate parameters required for noise and transient synthesizers. The synthesized audio is used to calculate the multi-scale STFT loss for audio reconstruction. Furthermore, to help the transient synthesizer precisely place the transient energies across the time axis, we compute an L2 loss relative to extracted peaks signal from input audio by performing harmonic-percussive source separation.

In Figure 8.3, we present our model architecture, which builds upon DDSP. Similar to NoisebandNet [170], we extract spectral similarity and loudness features from the input audio, sampled at $F_s = 44100$ Hz, using a window size of $W = 256$. These features are concatenated and passed into the decoder to generate parameters for the noise and transient synthesizers. Additionally, we compute a similarity vector by calculating the MD relative to each class, as illustrated in Figure 8.2. The similarity vector is an n-channel vector where n represents the total number of classes. This vector is first smoothed using a Multi-Layer Perceptron (MLP) layer. Subsequently, we employ Feature-wise Linear Modulation (FiLM) [267] to project the similarity vector onto the decoder with a sequence length matching that of the input features.

We utilize the same decoder architecture as DDSP [13], which is responsible for generating the magnitude frequency response for the noise synthesizer, as well as the amplitude and frequency parameters for the transient synthesizer. The outputs of the noise and transient synthesizers are combined and subsequently processed using the same reverb algorithm as DDSP [13].

8.4 Experiments

8.4.1 Dataset

TABLE 8.1. Number of Sounds in Footstep and Impact Categories

Footsteps		Impacts	
Category	Amount	Category	Amount
All	2127	All	1666
Board	83	Smash	153
Concrete	278	Gunshots	280
Gravel	292	Knocks	321
Leaves	378	Explosion	169
Snow	253	Footsteps	528
Squeaky	65	Bounce	155
Stairs	207	Punch	60
Tile	201		
Wood	370		

In this research, we focus primarily on the generation of rigid-body impact sounds. Most existing sound effect datasets categorize sounds based on broad classifications (e.g., footsteps, rain, birds), with limited detailed separation within categories (e.g., footsteps recorded on different materials). To address this, we sourced 2,127 footstep sounds from Freesound [288] and categorized them into nine groups based on the material of the contacting surface, as detailed in Table 8.1. The timbral differences across these categories provide a suitable framework for testing the effectiveness of our proposed conditioning variable in controlling the timbre of synthesized sounds. In addition to the Footstep-set, we collected another 1,666 impact sounds commonly used in games from Freesound. This broader classification allows us to evaluate the effectiveness of our conditioning approach under greater variance across

class embeddings. All sounds were sampled at 44.1 kHz with 16-bit depth and a duration of 4 seconds. The datasets are publicly available at: https://zenodo.org/records/14286414

8.4.2 Training

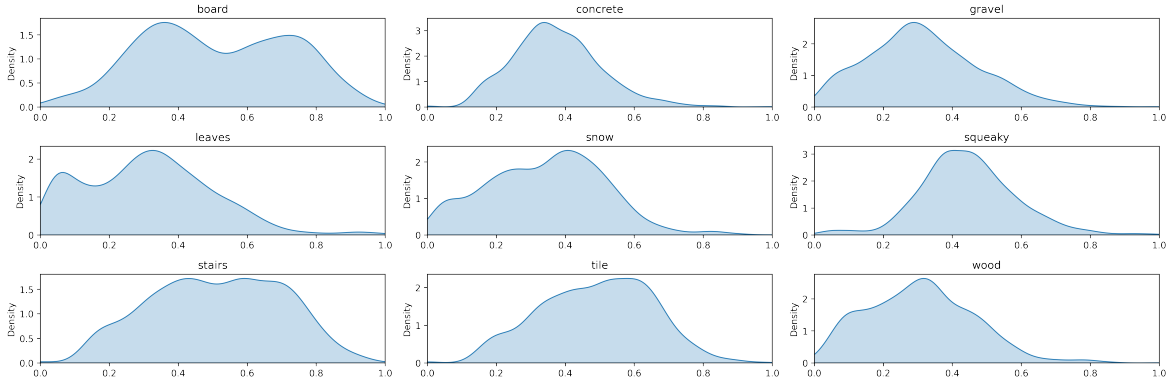
To set up the training, we first extract spectral centroid and loudness features from each audio sample, using an FFT size of 256 and 50% overlap. This process results in feature vectors with a sequence length of 690. For the decoder, we use a hidden size of 512 across all experiments. The noise synthesizer is configured with 100 frequency bands. As detailed in Section 8.3, we use 128 frequencies for sinusoidal modelling, which corresponds to half the frame size. To prevent aliasing, the frequencies are clipped to the range $(0, 128)$. For the multi-scale STFT loss [144], we compute spectrograms using FFT sizes of 2048, 1024, etc. down to 16, dividing by a factor of two. The model is trained for 5,000 epochs on an NVIDIA RTX 4070 Ti GPU, with a batch size of 16. We use the ADAM optimizer [257] with an initial learning rate of of 1×10^{-4}, which decays to 1×10^{-5} after 80% of the training epochs. Training takes approximately 17 hours for the Footstep-set and 13 hours for the Impact-set.

8.4.3 Fine-tuning based on similarity

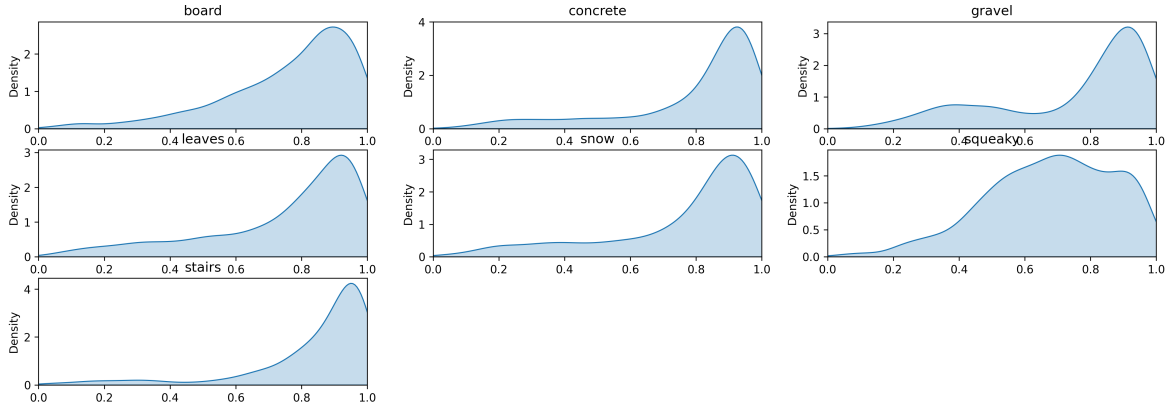
Since the extracted similarity score is normalized to the range $[0, 1]$, its distribution is not constrained to follow any specific form, unlike in ICGAN [283]. To better understand its probability density, we perform Kernel Density Estimation (KDE) as follows:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{8.4}$$

where $\hat{f}(x)$ is the estimated density function at point x, n is the number of data points, x_i are the data points, $K(\cdot)$ is a Gaussian kernel function, $K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$, and h is the bandwidth, a positive scale factor that affects the smoothness of the density estimation. In Figure 8.4a, we present the KDE results for each category in our footsteps dataset. The plots



(A) KDE of the footsteps dataset.



(B) KDE of the impacts dataset.

FIGURE 8.4. Kernel Density Estimation of the distribution of similarity scores of Footstep-set. The horizontal axis represents the similarity scores, ranging from [0,1]. The vertical axis shows the estimated density values. We noticed different distributions of similarity scores across different categories.

reveal significant variability in the probability density of similarity scores across classes. For instance, certain ranges of similarity scores (e.g., 0.8-1.0 for concrete) are underutilized during training compared to others (e.g., 0.3-0.5 for concrete). This imbalance can hinder the model's ability to generate diverse timbre variations.

To address this, after the model achieves satisfactory synthesis performance, we fine-tune the similarity conditioning layers to account for the underutilized ranges. Instead of using the similarity scores extracted from the dataset, we input pseudo-scores sampled uniformly from the range $[0, 1]$. This ensures that all score regions are captured and utilized for guiding sound synthesis. As illustrated in Figure 8.5, during fine-tuning, we freeze the decoder

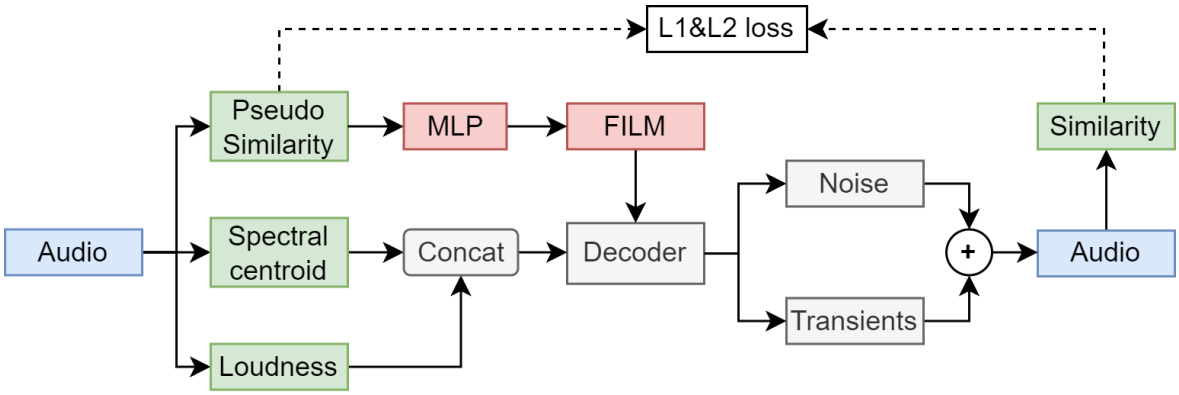


FIGURE 8.5. Fine-tuning process. The weights of the decoder are completely frozen during the fine-tuning stage. For each batch, we sample a vector from a uniform distribution in [0,1] and pass it to the decoder. We then measure the similarity score of the generated audio and evaluate a mixture of L1&L2 loss. In this way, the MLP + FILM layers are trained to match the input similarity scores with generated sounds with diverse timbres.

layers and update only the similarity conditioning layers. Since the objective is no longer to improve synthesis quality, we omit the reconstruction multi-scale STFT loss. Instead, we treat fine-tuning as a regression task, minimizing the error between the ground-truth similarity score (sampled from the uniform distribution) and the score measured from the generated audio. The loss function combines L_1 and L_2 losses and is defined as follows:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(s_i - \hat{s}_i)^2 + \frac{1}{n}\sum_{i=1}^{n}|s_i - \hat{s}_i| \tag{8.5}$$

where s_i represents the ground-truth similarity score, $\hat{s}_i$ is the predicted score and n is the batch size. Similar to the training process, we employ a batch size of 16 and a learning rate of 1×10^{-4} with the ADAM optimizer [257]. The fine-tuning is conducted for a total of 10,000 epochs on an NVIDIA 4070 Ti GPU. This process takes approximately 10 hours for the Footstep-set and 8 hours for the Impact-set.

8.5 Evaluation

8.5.1 Synthesis Performance

We evaluate the synthesis performance of our method against three neural audio synthesis models: the original DDSP (using only subtractive noise synthesis), the NoisebandNet model, and ICGAN [283].

For the DDSP model, we maintain the same configuration for its noise synthesizer and training settings as in our approach. This includes parameters such as the number of FFTs, multi-scale STFT loss, batch size, and training epochs. The model is also guided by spectral centroid and loudness features, extracted in the same manner as in our method, to inform the decoder during sound synthesis.

For the NoisebandNet model, we implement its filter bank using 2,048 frequency bands, a hidden size of 128, and a window size of 32. The model is trained separately on our two proposed datasets, using 10,000 epochs, a batch size of 16, and a learning rate of 0.001.The ICGAN model, which operates in the spectral-temporal domain, requires the extraction of Mel-spectrograms from each audio sample. These spectrograms are computed with a time resolution of 690 and 64 frequency bins. The model is then trained for 10,000 epochs with a batch size of 16 and a learning rate of $1e{-}4$.

To evaluate synthesis performance, we use two objective metrics: Frechet Audio Distance (FAD) [210] and Log Spectral Distance (LSD). FAD is a statistical evaluation metric that measures the overall similarity between two distributions: the reference dataset and the reconstructed dataset. In contrast, LSD provides a pairwise comparison between a reference sound and its corresponding generated sound. To generate the reconstructed dataset, we use the test sets of our datasets as input audio. For each test sample, we extract the required audio features to guide the models and set each model to evaluation mode to synthesize the reconstructed audio. This ensures that every reference sound in the test set is paired with a corresponding generated track from each model. FAD and LSD calculations are then performed using these

reference and generated datasets, allowing us to objectively assess the synthesis performance of each model.

8.5.2 Objective measure of controllability

In our study, we employ regression analysis, specifically Ordinary Least Squares (OLS) [289], to quantitatively evaluate the effectiveness of the conditioning method used in audio signal synthesis. Regression analysis is a robust statistical technique for quantifying the relationship between independent variables and a dependent variable. The OLS method, a fundamental and widely used form of regression analysis, estimates the parameters of a linear regression model by minimizing the sum of squared residuals. This approach ensures that the best-fit line closely approximates the true data points.

In our analysis, the independent variable, c, represents similarity scores interpolated between 0 and 1, indicating the proximity to a specific class A (ranging from close to distant). The dependent variable, y, is the measured similarity of the synthesized sound relative to each class. Ideally, when $c = 0$, the synthesized sound should exhibit a lower distance to class A, indicating high similarity. Conversely, when $c = 1$, the synthesized sound should demonstrate a higher distance, reflecting low similarity. This setup allows us to directly evaluate the model's ability to maintain meaningful conditioning during synthesis.

8.5.3 Regression Analysis Setup

To set up the experiment, we generated variations of each reference sound in our test set by interpolating the similarity score for one channel from 0 to 1 in 100 steps, while keeping all other channels fixed at 1. This resulted in 100 synthesized versions of each sound, all guided by the same reference. To measure the similarity of each output sound relative to each category, we opted to abandon the pre-trained CLAP model for embedding extraction to minimize potential biases. Instead, we used PANNs [88], a model extensively employed in audio tagging and sound event detection. A PANNs model pre-trained on Audioset [185] was fine-tuned to classify the synthesized sounds into their respective categories. After fine-tuning,

the model achieved a classification accuracy of 94.7%. Subsequently, we removed the final layer of the model to use the remaining network as an embedding extractor.

Using these embeddings, we calculated the Mahalanobis Distance (MD) of each synthesized sound's embedding relative to the embedding clusters of each class, following a process similar to the one described in Figure 8.2. For consistency and ease of visualization, the MD values for each channel were normalized to the range $[0, 1]$. Finally, each synthesized sound's similarity metric was paired with its corresponding guiding vector value, creating a dataset with independent variables (interpolated guiding vectors) and dependent variables (normalized MD values). This dataset forms the basis for evaluating the relationship between the guiding vectors and the resulting synthesized sound similarities.

Using Ordinary Least Squares (OLS) regression, we analyzed the relationship between the guiding vector values–linearly interpolated from 1 (least similar) to 0 (most similar)–and the normalized Mahalanobis distances, which represent the timbral similarity to each target class. OLS was chosen for this analysis because of its ability to provide straightforward parameter estimation by minimizing the sum of squared differences between observed and predicted values in a linear model. Through preliminary analysis, we observed that the relationship between the guiding vector values and the normalized distances is better described by an exponential rather than a linear trend. To accommodate this, we transformed the exponential regression model into a linear form by taking the natural logarithm of both sides. This transformation allows us to perform OLS regression on the modified model:

$$\ln(y) = \ln(a) + b \cdot x \tag{8.6}$$

where y is the dependent variable (normalized MD), x is the independent variable, $\ln(a)$ is the transformed coefficient to be estimated as the intercept, and b remains the slope of the regression line. The objective of ordinary least squares (OLS) regression is to minimize the

sum of squared residuals:

$$\min_{\ln(a),b} \sum_{i=1}^{n} \left(\ln(y_i) - (\ln(a) + b \cdot x_i)\right)^2 \tag{8.7}$$

where n is the number of interpolation steps. After estimating the parameters, the model for predictions is given by:

$$y_{\text{pred}} = a \cdot e^{b \cdot x_{\text{pred}}} \tag{8.8}$$

8.6 Results

8.6.1 Synthesis Performance

An impactful footstep often contains a sudden burst of energy that is challenging to model solely by filtering white noise through a linear time-varying finite impulse response (LTV-FIR) filter. In Figure 8.6, we present a randomly selected footstep sound, its reconstructed version using our proposed approach, the synthesized noise component, and the synthesized transient component. The reconstructed sound effectively captures the spectral content of the reference sound.

While the noise component, modelled with the LTV-FIR filter (using 256 FFTs), accurately reproduces the frequency spectrum of the reference audio, it falls short of representing the temporal dynamics characteristic of the original sound. This limitation is evident in the lack of temporal sharpness and impact, which are crucial for delivering the precise auditory cues associated with the source material.

To address these shortcomings, we incorporate transient synthesis (illustrated as vertical impulses with evenly distributed frequencies over a very short duration). This addition enhances the model's capability to reproduce the rapid onset and decay times that contribute to the immediacy and impact of natural sounds. Moreover, by regularizing transient synthesis with an additional L_2 loss based on a sparse, peak-detected signal, the decoder learns to emphasize amplitudes specifically at onset locations while attenuating amplitudes elsewhere. This targeted approach improves the realism and perceptual quality of the synthesized sounds.

TABLE 8.2. Model Performance Comparison Across Datasets and Measurements.

Models	Footsteps	Impacts
	FAD↓	
Ours	6.0 ± 1.39	2.06 ± 0.91
DDSP	10.0 ± 2.39	6.07 ± 3.53
Noisebandnet	4.06 ± 1.08	1.89 ± 1.04
ICGAN	12.52 ± 4.81	4.92 ± 0.84
	LSD↓	
Ours	0.14 ± 0.03	0.11 ± 0.05
DDSP	0.22 ± 0.05	0.30 ± 0.11
Noisebandnet	0.13 ± 0.02	0.11 ± 0.04
ICGAN	0.20 ± 0.06	0.18 ± 0.08

Since our datasets consist of multiple categories, we computed the mean and standard deviations for the entire dataset to ensure robust evaluation. Both the FAD and LSD results demonstrate that the performance of our model is comparable to that of the NoisebandNet model, which employs a large filterbank noise synthesizer with pre-stored noise bands. Although our architecture uses the same noise synthesizer as the original DDSP, the addition of the transient synthesizer significantly enhances the placement of transient energies at a sub-frame level. This improvement is crucial for reconstructing fine-grained details of impulsive sounds. Additionally, Noisebandnet requires significant amount of memory to store pre-computed coefficients for the filterbanks, thus making the training more difficult to accomodate a very large dataset. Furthermore, as Noisebandnet does not implement a latent space, it is not able to achieve timbre control due to the limitation of the explicit features it relies on such as loudness and spectral centroid.

The ICGAN model performs similarly to DDSP but falls short of delivering satisfactory quality compared to the other models. We hypothesize that this shortfall may be due to the implicit nature of its conditioning method for encoding class information. This approach introduces stochasticity into the conditioning process, potentially degrading audio fidelity.

Additionally, we observed significant differences in results across the two datasets we evaluated. The Impact-set achieved better results than the Footstep-set, despite identical configurations for both datasets. We hypothesize that this disparity arises from the lower variance in the Impact-set compared to the Footstep-set.

Overall, our approach demonstrates its effectiveness for synthesizing sound effects rich in dynamics and transients. It achieves high-quality audio synthesis without the need to train a large generative model, making it a practical solution for various sound synthesis applications.

8.6.2 Conditioning performance

In Figure .11, we present the plots of measured normalized Mahalanobis Distances (MD) as a function of interpolated similarity scores ranging from 0 to 1. The relationship between the normalized MD of each synthesized sound and its conditioning similarity score closely follows an exponential trend. Ordinary Least Squares (OLS) regression yields a mean R^2 value of 0.4774 for the Footstep-set model and 0.6041 for the Impact-set model, indicating a strong correlation between the independent variable c and the dependent variable y.

The plots reveal that almost all sound categories exhibit clear separations between the presence of a particular feature (at $c = 0$) and its absence (at $c = 1$). This demonstrates that the model successfully learns to output distinct timbres based on the proposed similarity scores. Additionally, the regression lines indicate a positive correlation between the normalized MD and the input similarity scores. This suggests that the conditioning method effectively encodes timbral information unique to different sounds, independent of other acoustic features such as loudness and spectral centroid, which were held constant during this test.

Notably, a clear separation is observed in the similarity score range of 0.8 to 1.0 across most categories. This implies that values below 0.8 have limited influence on encoding timbre variations, potentially indicating a model bias toward generating outputs with specific modes. To address this limitation, future research could explore strategies to expand the effective range of control. This may involve employing pre-processing techniques to flatten

the similarity score distribution or mapping the scores to a more meaningful scale to better utilize the entire range.

8.7 Creative Usage of Timbre Control

In Figure 8.7, we qualitatively demonstrate how interpolating between two conditioning similarity scores produces distinct timbres. For this example, we randomly selected a sound from our test dataset and extracted its loudness and spectral centroid as input. All channels of the similarity score were fixed at 1, except for the first channel (C_1) and the second channel (C_2) which correspond to footsteps on metallic boards and footsteps on gravel, respectively. These channels were interpolated from 0 to 1 (C_1) and 1 to 0 (C_2).

The resulting spectrograms reveal a clear progression in timbre. Initially, the signals exhibit prominent harmonics in the higher frequencies, characteristic of footsteps on metallic boards. As the interpolation progresses, these harmonics gradually transition into noisier signals lacking harmonic structure, which are indicative of footsteps on gravel. For a more detailed visualization and auditory experience, please visit our accompanying website [1]. We have also provided a colab notebook for interactive application of our research [2].

8.8 Chapter Summary

In this research, I proposed a similarity-based conditioning method designed to control timbre variations in sound effects. This method was implemented within a lightweight neural audio synthesis model and evaluated using our custom sound effect dataset, demonstrating its creative potential for sound design. I have shown that this conditioning approach enables the interpolation of subtle sound characteristics unique to different sound categories, providing nuanced control over the synthesis process.

[1] https://github.com/Reinliu/Similarity-Score

[2] https://colab.research.google.com/drive/1gjLFRM8DzigBnoUKtBYDlp0wgi8Rk1oc?usp=sharing

The synthesis quality achieved by our method is comparable to state-of-the-art data-driven sound effect synthesis algorithms. Furthermore, I have conducted a regression analysis to evaluate the relationship between the synthesized sounds and the conditioning vectors. The results indicate that our method effectively separates sound characteristics along the extremes of the conditioning vectors, validating its ability to generate distinct timbral variations.

However, I have also observed that the effective range of control is densely concentrated around similarity scores of 0.8 to 1.0, limiting the method's overall controllability. Future research could focus on improving the distribution of the conditioning similarity scores, ensuring a flatter, more uniform range to enhance the granularity and effectiveness of timbre control.

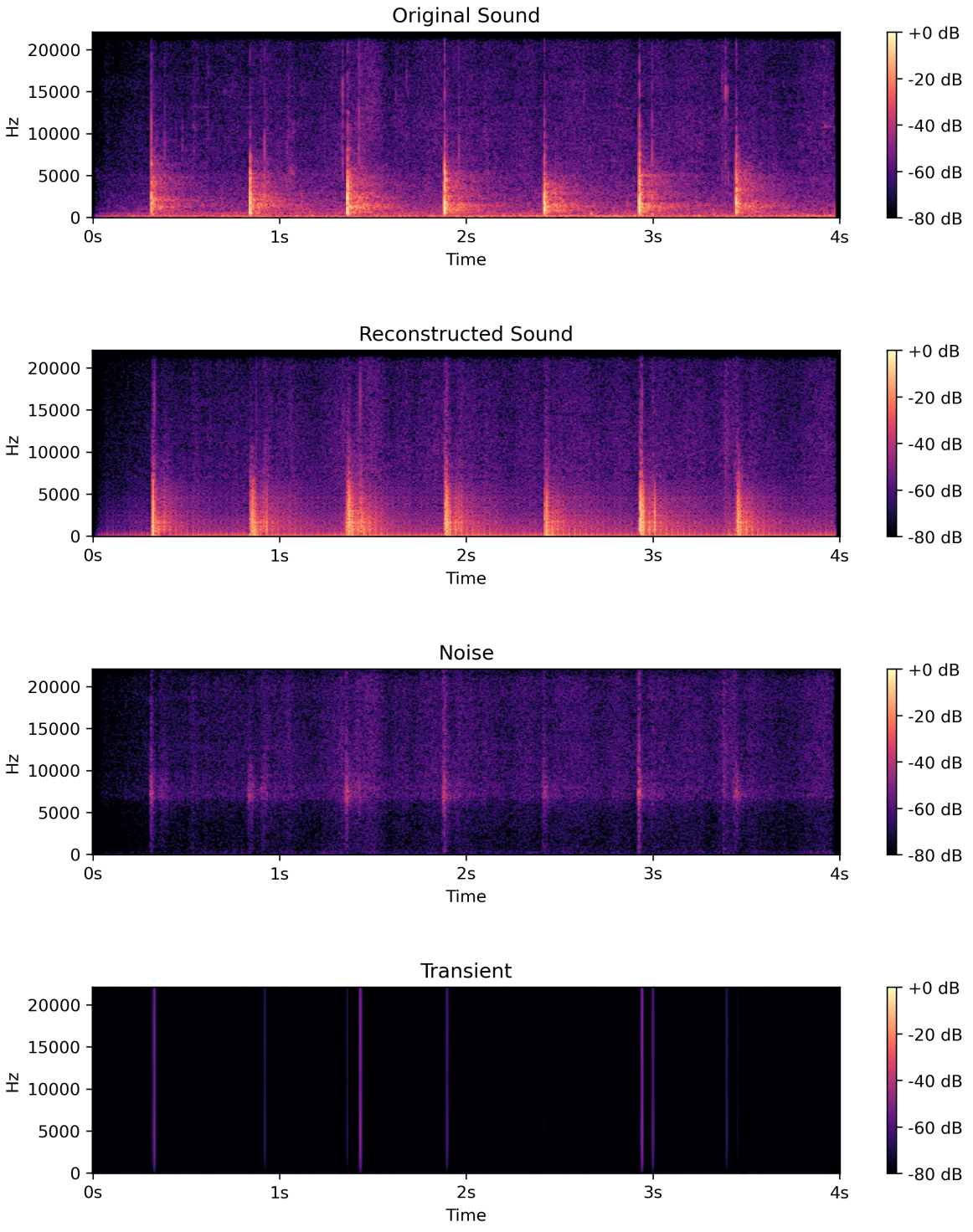


FIGURE 8.6. Sound reconstruction based on a reference track. The spectral centroid, loudness envelope, as well as similarity scores are extracted from the original sound and used for audio reconstruction. The reconstructed, noise, and transient separations are shown from top to bottom.

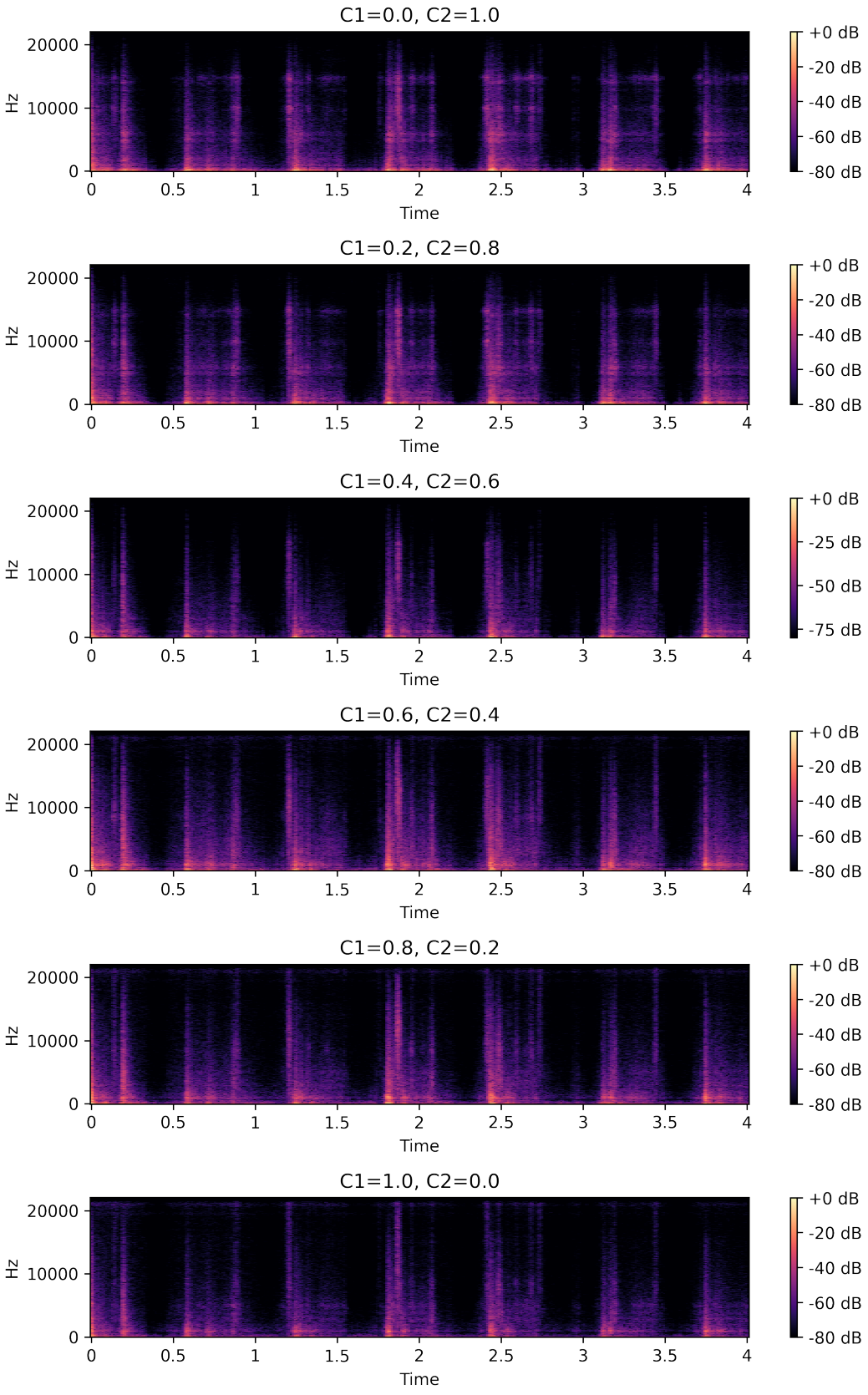


FIGURE 8.7. An example of interpolating between two conditioning similarity scores. We fixed all other channels as 1 for the whole time, and interpolated the first channel C_1 (Footstep: Board) from 0 to 1, while the second channel C_2 (Footstep: Gravel) from 1 to 0. A lower conditioning score of C_1 or C_2 indicates a higher similarity with class one or two, respectively.

CHAPTER 9

Conclusion

In this thesis, I reviewed the various methods of sound effects synthesis. In the era of deep learning, neural audio synthesis models employ different conditioning labels 7.1 to associate audio samples with particular interpretable information that could be used for control. In this research, I have proposed several approaches to enable controllable sound generation under limited audio datasets.

Particularly, in Chapter 4 and Chapter 5, I aim to answer the research question: **How to generate a wider range of sounds using neurla audio synthesis?** By adapting simple GAN architectures such as WGAN [113], I explored magnitude spectrogram generation and trained our model on different types of sound effects, ranging from harmonic to inharmonic, impulsive to sustained sounds. As GAN-based models usually suffer from mode collapse [112], I incorporated a reconstruction loss on the global energy levels of the audio samples, specifically the root mean square (RMS) values of the audio waveforms, rather than the full waveform representation. Through experimentation, I found this approach helps balance the trade-off between quality and diversity of the generated sounds. For the data scarcity problem, in Chapter 8, I introduced two audio datasets targeted at common sound effects modelling. They are publicly available from Zenodo [1].

To address the research question **How to provide meaningful controls for neural audio synthesis**, in Chapter 6, I studied featured-conditioned SFX generation using DDSP by training on three categories of common sound effects without text descriptions. DDSP was known for its lightweight and flexible architecture, especially for musical instrument modelling. To better model inharmonic sound effects, I adopt Verma's [134] method for

[1] https://zenodo.org/records/14286414

transient synthesis and adapt the DDSP on a VAE architecture. By meticulous design of the encoder, I obtained a low-dimensional latent space that encodes timbre information on the frame level. This approach was tested on a group of audio experts and non-experts. The results show that its performance is comparable to the original audio, and by varying the latent vector, I was able to modify the spectral distribution, or timbre, (shifting from the timbre of a Mercedes engine to a BMW engine) in an implicit way. Despite the ability to alter the timbral characteristics, the non-linear relationship between the latent vector and the spectral changes makes it difficult to vary the sounds controllably. Sound designers must iteratively explore the latent space, often moving from common ($z = 0$) to more unique timbres ($|z| > 0$), to achieve the desired sound modifications.

To provide more interpretable control over the sound characteristics given limited audio datasets with only categorical information, in Chapter 7 and Chapter 8, I explored conditioning methods for generative sound modelling. Inspired by DarkGAN [84] which utilizes pre-trained audio classifiers to generate soft labels for each sound, I conditioned our NAS models on continuous pre-processed soft labels similar to CCGAN [264]. This approach introduces uncertainty between the boundaries of discrete classes, which allows for possible interpolation in the conditioning space. In Chapter 7, I started with a simple yet effective approach that utilizes latent vectors as the conditioning labels for GAN training, and I refer to this model as ICGAN. To no surprise, this approach significantly outperforms one-hot vector conditioning in timbre interpolation but is inferior to direct latent space interpolation between two audio targets as introduced from latent timbre synthesis [270]. This approach exhibits greater performance in our quantitative evaluation metrics by incorporating amplitude vectors for interpolation.

However, as uncertainty was introduced between discrete classes, although it is capable of interpolating timbre between one class to another, the sound quality deteriorates significantly when they are trained on sounds from different categories. Therefore in Chapter 8, I have refined this approach by conditioning synthesis models on similarity scores that are obtained from large pre-trained audio representation models. This time it was built on top of our DDSP-SFX model, and this model is denoted as Simi-SFX, where similarity scores were

used to guide the sound synthesis of sound effects. This approach not only enables timbre interpolation across discrete classes, but also restricts the parameter to be within a controllable range $[0, 1]$.

To appropriately evaluate the controllability of a NAS model and differentiate the timbre differences among generated sounds, in Chapter 6, I tested the perceptual capability of differentiating between sounds in slightly varied timbres. Results show that many participants were not able to tell the difference when the latent variable is set too low ($|z| < 0.5$). This means that the variation of timbre is not significant when we change the latent variable. I hypothesize that this is because the training dataset does not capture the full variations of sounds. In Chapter 7 and Chapter 8, I have introduced several metrics to measure the effectiveness of timbre encoding. From the quantitative analysis, it was shown that our approach could interpolate sound timbres by varying the conditioning vectors. However, the effective range of timbre encoding (REC) does not exactly overlap with the full range of conditioning vectors. For fine-grained control, improvements could be made on remapping the conditioning timbre space to enable smoother interpolation within the full range.

Nevertheless, the Simi-SFX model integrates many components of this research, including variable-length audio generation, the ability to vary timbre using a controllable range, the refined approach towards transient modelling under DDSP, etc. Additionally, Simi-SFX is completely unsupervised and is easy to train with small datasets as few as ten minutes of audio recording. When using the similarity score to control the timbres of sounds, our approach is highly flexible as it does not require any reference sound. I hope this research could help sound designers in their creative workflow and spark more creativity in the sound design industry.

9.1 Limitation and Reflection

Admittedly, there are limitations of my approach towards controllable audio synthesis. Although soft labels extracted from pre-trained audio classifier [88, 89] were shown to correlate with the perceptual characteristics [285], timbre is a complex attribute of sound which can

not be easily deciphered with one particular acoustic feature. To this end, the degree of controllability is relied on the performance of pre-trained audio classifiers. Furthermore, as similarity is only a relative attribute across various kinds of sounds, it becomes difficult from a control perspective when the types of sounds grow extensively. Therefore, this approach works most effectively when users predefine a limited number of sound categories to achieve fine-grained timbre control.

Beyond the technological aspect of generative audio synthesis, there are also social and ethical aspects of how to best utilise such algorithms. First, commercializing the generative samples from a pre-trained NAS model on licensed datasets without appropriate attribution or licensing would inevitably jeopardize the community of sound effects libraries. Nowadays, many voice-cloning and audio deepfake technologies [290] pose issues in the privacy of personal data. The use of such neural audio synthesis technologies needs to strictly follow the ethics and social regulations.

9.2 Future Outlook

This thesis explores neural sound effects generation conditioned on class labels, feature vectors, and soft labels. I focused primarily on the use of GAN 4, 5, 7, VAE 6, 7 and DDSP 6, 8 architectures on the generation of audio in the time domain 6, 8 and time-frequency domain 4, 5, 7. I proposed a controllable SFX generation approach by conditioning NAS models on similarity scores. The regression analysis shown in Figure .11 leaves space for further work on matching the range of effective timbre control with the conditioning vectors to improve expressive timbre control. As introduced in Section 3.1, numerous hybrid models combine different generative models and DSP-based methods. Therefore, future work could be done to study other hybrid approaches for sound effects synthesis. For example, incorporating diffusion models with transformers or other autoregressive models to generate coherent time-varying waveforms in the time domain could enhance audio samples' continuity while enabling timbre control.

Beyond the approaches explored in this thesis, text-to-audio (TTA) systems are maturing rapidly, with autoregressive language model style models such as AudioGen [108] and latent-diffusion methods such as AudioLDM [69] improving textual alignment and sample quality, while language-modelling over audio tokens (AudioLM [183]) targets long-range structure in environmental sounds and speech [108, 69, 132]. Multimodal learning such as video-to-audio [197, 291, 175] further suggests that visual priors can help tighten the temporal synchronization with audio and capture and preserve semantics of acoustic events. Nowadays, many popular generative audio approaches have been utilized in the industry and incubated into powerful tools towards controllable audio generation. For example, creative integrations of text-to-SFX with the addition of voice for temporal control such as Adobe Firefly [292] are bringing controllable SFX generation into everyday workflows. Paired with large audio–language encoders for instruction following, retrieval, and attribute grounding, these trends point to hybrid systems where global descriptors (from language) and local structure (from explicit control maps) are necessary for generating high-fidelity and controllable SFX [89].

Spatial audio generation is also becoming a popular research topic, with end-to-end FOA/ambisonic synthesis and even language-driven spatialization beginning to appear, indicating that control over where sound occurs will progress alongside control over what and how it sounds [293, 294, 295]. Once the intricate relationship between textual descriptions and spatial audio is built with contrastive learning approaches [89, 196], it would become possible to render moving-source spatial audio in real-time with generative models.

From a control perspective, ControlNet-based approaches were shown to be good at enforcing time-local patterns (envelopes, onsets, rhythmic activations). It can also be layered on top of class/feature/descriptor controls to increase precision without sacrificing timbre flexibility [174]. Meanwhile, many work in timbre space exploration [251, 129, 270] tries to distill and extract important semantic representations of audio timbre from learned unstructured latent space. The approach proposed in this thesis can be a good addition of audio control methods, where different techniques could be combined to offer better controllability to the users.

Given the recent advancements in neural audio synthesis in modelling general sound effects, I believe my approach could be a good supplement in providing meaningful timbre controls for the industry or academia. This approach could be easily incorporated into multi-modal sound synthesis networks, where texts/visuals could be used to control the semantic aspect of audio content. My control approach would then be able to supply such systems in offering fine-grained timbre control with a simple-to-use control knob. In the future, I hope that large audio datasets could be utilized to test the performance of my approach for controlling a wider range of sound effects.

Bibliography

[1] C. Hausman, F. Messere and P. Benoit. *Modern Radio and Audio Production: Programming and Performance*. Cengage Learning, 2015.

[2] Lie Lu, Liu Wenyin and Hong-Jiang Zhang. 'Audio textures: theory and applications'. In: *IEEE Transactions on Speech and Audio Processing* 12.2 (2004), pp. 156–167. DOI: 10.1109/TSA.2003.819947.

[3] Ehab A. AlBadawy et al. 'Vocbench: A Neural Vocoder Benchmark for Speech Synthesis'. In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pp. 881–885. DOI: 10.1109/ICASSP43922.2022.9746698.

[4] Daisy Stanton, Yuxuan Wang and RJ Skerry-Ryan. 'Predicting expressive speaking style from text in end-to-end speech synthesis'. In: *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2018, pp. 595–602.

[5] Mingjian Chen et al. 'AdaSpeech: Adaptive Text to Speech for Custom Voice'. In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=Drynvt7gg4L.

[6] Younggun Lee and Taesu Kim. 'Robust and fine-grained prosody control of end-to-end speech synthesis'. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 5911–5915.

[7] Guangzhi Sun et al. 'Generating diverse and natural text-to-speech samples using a quantized fine-grained vae and autoregressive prosody prior'. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 6699–6703.

[8] Yukiya Hono et al. 'Hierarchical Multi-Grained Generative Model for Expressive Speech Synthesis'. In: *Interspeech*. 2020. URL: https://api.semanticscholar.org/CorpusID:221802404.

[9] Chung-Ming Chien and Hung-yi Lee. 'Hierarchical prosody modeling for non-autoregressive speech synthesis'. In: *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2021, pp. 446–453.

[10] Mingjian Chen et al. 'Multispeech: Multi-speaker text to speech with transformer'. In: *Interspeech*. 2020.

[11] Minchan Kim et al. 'Expressive Text-to-Speech Using Style Tag'. In: *Interspeech*. 2021. URL: https://api.semanticscholar.org/CorpusID:232478871.

[12] Jesse Engel et al. 'GANSynth: Adversarial Neural Audio Synthesis'. In: *International Conference on Learning Representations*. Vol. abs/1902.08710. 2019. URL: https://api.semanticscholar.org/CorpusID:67856213.

[13] Jesse Engel et al. 'DDSP: Differentiable Digital Signal Processing'. In: *International Conference on Learning Representations* abs/2001.04643 (2020). URL: https://api.semanticscholar.org/CorpusID:210473083.

[14] Jesse Engel et al. 'Neural audio synthesis of musical notes with wavenet autoencoders'. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1068–1077.

[15] Andy Farnell. *Designing sound*. London, UK: Mit Press, October 2010. ISBN: 9780262014410. URL: https://mitpress.mit.edu/9780262014410/designing-sound/.

[16] Dimitris Menexopoulos, Pedro Pestana and Joshua Reiss. 'The state of the art in procedural audio'. In: *Journal of the Audio Engineering Society* 71.12 (2023), pp. 826–848.

[17] D. Griffin and Jae Lim. 'Signal estimation from modified short-time Fourier transform'. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32.2 (1984), pp. 236–243. DOI: 10.1109/TASSP.1984.1164317.

[18] Diederik P. Kingma and Max Welling. 'Auto-Encoding Variational Bayes'. In: *International Conference on Learning Representations* abs/1312.6114 (2013). URL: https://api.semanticscholar.org/CorpusID:216078090.

[19] Robert L. Mott. *Sound Effects: Radio, Television and Film*. McFarland, 2014.

[20] Jim Stinson. *Real-time Sound Effects: The Foley Way*. Videomaker.com. Archived from the original on 7 September 2009. Retrieved 20 May 2010. July 1999.

[21] Chris Hopkins. *Video Game Audio: A History, 1972–2020*. McFarland, 2022.

[22] Stefan Bilbao. *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*. Wiley Publishing, 2009. ISBN: 0470510463.

[23] Claude E. Shannon. 'A Mathematical Theory of Communication'. In: *The Bell System Technical Journal* 27 (1948), pp. 379–423, 623–656.

[24] Joshua D Reiss and Andrew McPherson. *Audio Effects*. CRC Press, 2014.

[25] Julius O. Smith III. *Introduction to Digital Filters with Audio Applications*. W3K Publishing, 2007. URL: https://ccrma.stanford.edu/~jos/filters/.

[26] Curtis Roads. *The Computer Music Tutorial*. Cambridge, MA: MIT Press, 1996.

[27] Jari Kleimola et al. 'Feedback Amplitude Modulation Synthesis'. In: *EURASIP Journal on Advances in Signal Processing* 2011.1 (2011), pp. 1–13. URL: https://asp-eurasipjournals.springeropen.com/articles/10.1155/2011/434378.

[28] John M. Chowning. 'The Synthesis of Complex Audio Spectra by Means of Frequency Modulation'. In: *Journal of the Audio Engineering Society* 21.7 (1973), pp. 526–534. URL: https://web.eecs.umich.edu/~fessler/course/100/misc/chowning-73-tso.pdf.

[29] Curtis Roads. 'Introduction to granular synthesis'. In: *Computer Music Journal* 12.2 (1988), pp. 11–13.

[30] Ata Mahjoubfar et al. 'Time stretch and its applications'. In: *Nature Photonics* 11.6 (2017), pp. 341–351.

[31] Joshua Batty, Kipps Horn and Stefan Greuter. 'Audiovisual granular synthesis: micro relationships between sound and image'. In: *Proceedings of The 9th Australasian Conference on Interactive Entertainment: Matters of Life and Death*. 2013, pp. 1–7.

[32] Wick Van den Belt. *Audio Engineering-Dynamic Processing*. Vol. 1. Wick van den Belt, 2013.

[33] Matthew Yee-King and Igor Dall'Avanzi. 'Procedural Audio in Video Games'. In: *Encyclopedia of Computer Graphics and Games*. Springer, 2024, pp. 1483–1487.

[34] Marc Le Brun. 'Digital waveshaping synthesis'. In: *Journal of the Audio Engineering Society* 27.4 (1979), pp. 250–266.

[35] Daniel Arfib. 'Digital Synthesis of Complex Spectra by means of Multiplication of Non-linear Distorted Sine Waves'. In: *Journal of The Audio Engineering Society* 27 (1978), pp. 757–768. URL: https://api.semanticscholar.org/CorpusID:32725015.

[36] Xavier Serra and Julius Smith. 'Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition'. In: *Computer Music Journal* 14.4 (1990), pp. 12–24. ISSN: 01489267, 15315169. URL: http://www.jstor.org/stable/3680788 (visited on 17/09/2022).

[37] D.-Y. Wu et al. 'DDSP-Based Singing Vocoders: A New Subtractive-Based Synthesizer and a Comprehensive Evaluation'. In: *Ismir 2022 Hybrid Conference*. 2022.

[38] M. Yee-King and L. McCallum. *Studio Report: Sound Synthesis With DDSP and Network Bending Techniques*. 2021.

[39] F. Ganis et al. 'Real-Time Timbre Transfer and Sound Synthesis Using DDSP'. In: *Proc. 18th Sound and Music Computing Conference (SMC 2021)*. Sound and Music Computing Network. 2021, pp. 175–182.

[40] Franco Caspe, Andrew McPherson 0002 and Mark Sandler 0001. 'DDX7: Differentiable FM Synthesis of Musical Instrument Sounds'. In: *Proceedings of the 23rd International Society for Music Information Retrieval Conference, ISMIR 2022, Bengaluru, India, December 4-8, 2022*. Ed. by Preeti Rao et al. 2022, pp. 608–616. ISBN: 978-1-7327299-2-6. URL: https://archives.ismir.net/ismir2022/paper/000073.pdf.

[41] Richard E. Turner. 'Statistical Models for Natural Sounds'. PhD thesis. Gatsby Computational Neuroscience Unit, UCL, 2010.

[42] Tony S. Verma and Teresa H. Y. Meng. 'Extending Spectral Modeling Synthesis with Transient Modeling Synthesis'. In: *Computer Music Journal* 24.2 (June 2000), pp. 47–59. ISSN: 0148-9267. DOI: 10.1162/014892600559317. eprint: https:

//direct.mit.edu/comj/article-pdf/24/2/47/1848972/014892600559317.pdf. URL: https://doi.org/10.1162/014892600559317.

[43] David Baraff. 'Physically based modeling: Rigid body simulation'. In: *SIGGRAPH Course Notes, ACM SIGGRAPH* 2.1 (2001), pp. 2–1.

[44] Luca Turchet. 'Footstep sounds synthesis: Design, implementation, and evaluation of foot–floor interactions, surface materials, shoe types, and walkers' features'. In: *Applied Acoustics* 107 (2016), pp. 46–68. ISSN: 0003-682X. DOI: https://doi.org/10.1016/j.apacoust.2015.05.013. URL: https://www.sciencedirect.com/science/article/pii/S0003682X15001747.

[45] Huixuan Han et al. 'Modelling of lateral forces generated by pedestrians walking across footbridges'. In: *Applied Mathematical Modelling* 89.2 (2021), pp. 1775–1791. ISSN: 0307-904X. DOI: 10.1016/j.apm.2020.08.081. URL: https://www.sciencedirect.com/science/article/pii/S0307904X20305291.

[46] David Baraff. 'An introduction to physically based modeling: rigid body simulation ii—nonpenetration constraints'. In: *SIGGRAPH course notes* (1997), pp. D31–D68.

[47] Kees Van Den Doel and Dinesh K Pai. 'Modal synthesis for vibrating objects'. In: *Audio Anectodes. AK Peter, Natick, MA* (2003), pp. 1–8.

[48] Mauricio Alvarez, David Luengo and Neil D Lawrence. 'Latent force models'. In: *Artificial intelligence and statistics*. PMLR. 2009, pp. 9–16.

[49] Julius O Smith. 'Physical modeling using digital waveguides'. In: *Computer music journal* 16.4 (1992), pp. 74–91.

[50] Balázs Bank, Stefano Zambon and Federico Fontana. 'A modal-based real-time piano synthesizer'. In: *IEEE transactions on audio, speech, and language processing* 18.4 (2010), pp. 809–821.

[51] P-A Taillard et al. 'Modal analysis of the input impedance of wind instruments. Application to the sound synthesis of a clarinet'. In: *Applied acoustics* 141 (2018), pp. 271–280.

[52] Cynthia Bruyns. 'Modal Synthesis for Arbitrarily Shaped Objects'. In: *Computer Music Journal* 30.3 (2006), pp. 22–37. URL: http://www.jstor.org/stable/4617941.

[53] William J. Wilkinson, Joshua D. Reiss and Dan Stowell. 'Latent Force Models for Sound: Learning Modal Synthesis Parameters and Excitation Functions from Audio Recordings'. In: *Proceedings of the 20th International Conference on Digital Audio Effects*. Edinburgh, UK, 2017.

[54] Lucas Mengual, David Moffat and Joshua D. Reiss. 'Modal synthesis of weapon sounds'. In: *Proc. Audio Engineering Society Conference: 61st Audio Engineering Society International Conference: Audio for Games*. London, Feb. 2016.

[55] Don Morgan and Sanzheng Qiao. 'Analysis of Damped Mass-Spring Systems for Sound Synthesis'. In: *EURASIP Journal on Audio, Speech, and Music Processing* 2009 (2009), pp. 1–19. URL: https://api.semanticscholar.org/CorpusID:541351.

[56] Rodrigo Diaz et al. 'Rigid-Body Sound Synthesis with Differentiable Modal Resonators'. In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023, pp. 1–5. DOI: 10.1109/ICASSP49357.2023.10095139.

[57] Damian Murphy et al. 'Acoustic Modeling Using the Digital Waveguide Mesh'. In: *IEEE Signal Processing Magazine* 24.2 (2007), pp. 55–66. DOI: 10.1109/MSP.2007.323264.

[58] Julius O Smith III. 'Principles of digital waveguide models of musical instruments'. In: *Applications of digital signal processing to audio and acoustics*. Springer, 1998, pp. 417–466.

[59] Rod Selfridge et al. 'Creating real-time aeroacoustic sound effects using physically informed models'. In: *Journal of the Audio Engineering Society* (2018).

[60] Stefano Baldan et al. 'Physically informed car engine sound synthesis for virtual and augmented environments'. In: *2015 IEEE 2nd VR Workshop on Sonic Interactions for Virtual Environments (SIVE)*. 2015, pp. 1–6. DOI: 10.1109/SIVE.2015.7361287.

[61] Leonard J Paul. 'Granulation of sound in video games'. In: *methods* 2.3 (2011), p. 4.

[62] Leonard J Paul. 'An introduction to granular synthesis in video games'. In: *From Pac-Man to Pop Music*. Routledge, 2017, pp. 135–149.

[63] Siyuan Shan et al. ‘Differentiable wavetable synthesis’. In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 4598–4602.

[64] menexopoulos dimitris. ‘using texture maps to procedurally generate sound in virtual environments’. In: *journal of the audio engineering society* 8 (Apr. 2024).

[65] Diemo Schwarz. ‘Current research in concatenative sound synthesis’. In: *International Computer Music Conference (ICMC)*. 2005, pp. 1–1.

[66] Diemo Schwarz and Peter Knees. ‘Real-Time Corpus-Based Sound Synthesis and its Use in Creative Context’. In: *Proceedings of the 14th International Conference on Digital Audio Effects (DAFx)*. Paris, France, 2011.

[67] Vesa Välimäki et al. ‘Physical modeling of plucked string instruments with application to real-time sound synthesis’. In: *Journal of the Audio Engineering Society* 44.5 (1996), pp. 331–353.

[68] Santiago Pascual et al. ‘Full-Band General Audio Synthesis with Score-Based Diffusion’. In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2022), pp. 1–5. URL: https://api.semanticscholar.org/CorpusID:253116859.

[69] Haohe Liu et al. ‘AudioLDM: Text-to-Audio Generation with Latent Diffusion Models’. In: *International Conference on Machine Learning*. 2023. URL: https://api.semanticscholar.org/CorpusID:256390486.

[70] Jian-Xin Pan et al. ‘Maximum likelihood estimation’. In: *Growth curve models and statistical diagnostics* (2002), pp. 77–158.

[71] Douglas A Reynolds et al. ‘Gaussian mixture models.’ In: *Encyclopedia of biometrics* 741.659-663 (2009).

[72] Sean R Eddy. ‘Hidden markov models’. In: *Current opinion in structural biology* 6.3 (1996), pp. 361–365.

[73] Muhammad Huzaifah and Lonce Wyse. ‘Deep generative models for musical audio synthesis’. In: *Handbook of artificial intelligence for music: foundations, advanced approaches, and developments for creativity* (2021), pp. 639–678.

[74] Dor Bank, Noam Koenigstein and Raja Giryes. 'Autoencoders'. In: *Machine learning for data science handbook: data mining and knowledge discovery handbook* (2023), pp. 353–374.

[75] Durk P Kingma and Prafulla Dhariwal. 'Glow: Generative flow with invertible 1x1 convolutions'. In: *Advances in neural information processing systems* 31 (2018).

[76] George Papamakarios et al. 'Normalizing Flows for Probabilistic Modeling and Inference'. In: *J. Mach. Learn. Res.* 22 (2019), 57:1–57:64. URL: https://api.semanticscholar.org/CorpusID:208637478.

[77] A. van den Oord et al. 'WaveNet: A Generative Model for Raw Audio'. In: *The 9th ISCA Speech Synthesis Workshop*. Sunnyvale, CA, USA: ISCA, Sept. 2016, p. 125. URL: http://www.isca-speech.org/archive/SSW_2016/abstracts/ssw9_DS-4_van_den_Oord.html.

[78] Soroush Mehri et al. 'SampleRNN: An Unconditional End-to-End Neural Audio Generation Model'. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=SkxKPDv5xl.

[79] Ian Goodfellow et al. 'Generative adversarial networks'. In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[80] Yang Song et al. 'Maximum likelihood training of score-based diffusion models'. In: *Advances in neural information processing systems* 34 (2021), pp. 1415–1428.

[81] Aoife McDonagh et al. 'Synthesizing game audio using deep neural networks'. In: *2018 IEEE Games, Entertainment, Media Conference (GEM)*. IEEE. 2018, pp. 1–9.

[82] Jonathan Ho, Ajay Jain and Pieter Abbeel. 'Denoising diffusion probabilistic models'. In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[83] Dongchao Yang et al. 'Diffsound: Discrete Diffusion Model for Text-to-Sound Generation'. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 31 (2023), pp. 1720–1733. DOI: 10.1109/TASLP.2023.3268730.

[84] Javier Nistal, Stefan Lattner and Gaël Richard. 'DarkGAN: Exploiting Knowledge Distillation for Comprehensible Audio Synthesis with GANs'. In: *International Society for Music Information Retrieval Conference*. 2021. URL: https://api.semanticscholar.org/CorpusID:236881319.

[85] Yuexi Du et al. 'Conditional generation of audio from video via foley analogies'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 2426–2436.

[86] Balandino Di Donato and Iain McGregor. 'The digital Foley: what Foley artists say about using audio synthesis.' In: *Audio Engineering Society Conference: AES 2024 International Audio for Games Conference*. Audio Engineering Society. 2024.

[87] Purvi Agrawal and Sriram Ganapathy. 'Interpretable representation learning for speech and audio signals based on relevance weighting'. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2020), pp. 2823–2836.

[88] Qiuqiang Kong et al. 'PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition'. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2019), pp. 2880–2894. URL: https://api.semanticscholar.org/CorpusID:209444382.

[89] Yusong Wu et al. 'Large-Scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation'. In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2022), pp. 1–5. URL: https://api.semanticscholar.org/CorpusID:253510826.

[90] Cécile Picard, Nicolas Tsingos and François Faure. 'Audio texture synthesis for complex contact interactions'. In: *VRIPHYS 2008-5th Workshop On Virtual Reality Interaction and Physical Simulation*. Eurographics Association. 2008, pp. 83–88.

[91] Muhammad Huzaifah bin Md Shahrin and Lonce Wyse. 'Applying visual domain style transfer and texture synthesis techniques to audio: insights and challenges'. In: *Neural Computing and Applications* 32.4 (2020), pp. 1051–1065.

[92] Hendrik Purwins et al. 'Deep learning for audio signal processing'. In: *IEEE Journal of Selected Topics in Signal Processing* 13.2 (2019), pp. 206–219.

[93] Chao Song and Tariq A Alkhalifah. 'Wavefield reconstruction inversion via physics-informed neural networks'. In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2021), pp. 1–12.

[94] Antonio Alguacil et al. 'Predicting the propagation of acoustic waves using deep convolutional neural networks'. In: *Journal of Sound and Vibration* 512 (2021), p. 116285.

[95] Jungil Kong, Jaehyeon Kim and Jaekyoung Bae. 'HiFi-GAN: generative adversarial networks for efficient and high fidelity speech synthesis'. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS'20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.

[96] Kundan Kumar et al. 'MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis'. In: *Neural Information Processing Systems*. 2019. URL: https://api.semanticscholar.org/CorpusID:202777813.

[97] Nicolas Saint-Arnaud and Kris Popat. 'Analysis and synthesis of sound textures'. In: *Computational auditory scene analysis*. CRC Press, 2021, pp. 293–308.

[98] Naotake Masuda and Daisuke Saito. 'Quality-diversity for Synthesizer Sound Matching'. In: *Journal of Information Processing* 31 (2023), pp. 220–228.

[99] Marco Comunità, Huy Phan and Joshua D. Reiss. 'Neural Synthesis of Footsteps Sound Effects with Generative Adversarial Networks'. In: *Audio Engineering Society Convention 152*. Convention Paper 10583. Online and In-Person, May 2022: Audio Engineering Society, 2022. URL: http://www.aes.org/e-lib/browse.cfm?elib=21796.

[100] Adrian Barahona-Rios and Sandra Pauletto. 'Synthesising Knocking Sound Effects Using Conditional WaveGAN'. In: *Sound and Music Computing*. 2020. URL: https://api.semanticscholar.org/CorpusID:229198283.

[101] Sanchita Ghose and John J. Prevost. 'FoleyGAN: Visually Guided Generative Adversarial Network-Based Synchronous Sound Generation in Silent Videos'. In: *IEEE Transactions on Multimedia* 25 (2023), pp. 4508–4519. DOI: 10.1109/TMM.2022.3177894.

[102] Martin Russ. *Sound synthesis and sampling*. Routledge, 2012.

[103] Zhen Ye et al. ‘NAS-FM: neural architecture search for tunable and interpretable sound synthesis based on frequency modulation’. In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. IJCAI ’23. Macao, P.R.China, 2023. ISBN: 978-1-956792-03-4. DOI: 10.24963/ijcai.2023/651. URL: https://doi.org/10.24963/ijcai.2023/651.

[104] Marios Athineos and Daniel PW Ellis. ‘Autoregressive modeling of temporal envelopes’. In: *IEEE transactions on signal processing* 55.11 (2007), pp. 5237–5245.

[105] Xubo Liu et al. ‘Conditional sound generation using neural discrete time-frequency representation learning’. In: *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2021, pp. 1–6.

[106] Xi Chen et al. ‘PixelSNAIL: An Improved Autoregressive Generative Model’. In: *International Conference on Learning Representations*. 2017.

[107] Aäron van den Oord, Oriol Vinyals and Koray Kavukcuoglu. ‘Neural Discrete Representation Learning’. In: *Neural Information Processing Systems*. 2017. URL: https://api.semanticscholar.org/CorpusID:20282961.

[108] Felix Kreuk et al. ‘Audiogen: Textually guided audio generation’. In: *International Conference on Learning Representations*. 2022.

[109] Pengzhi Li, Yan Pei and Jianqiang Li. ‘A comprehensive survey on design and application of autoencoder in deep learning’. In: *Applied Soft Computing* 138 (2023), p. 110176.

[110] Antoine Caillon and Philippe Esling. *RAVE: A variational autoencoder for fast and high-quality neural audio synthesis*. 2021. arXiv: 2111.05011 [cs.LG].

[111] Rongjie Huang et al. ‘Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models’. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 13916–13932.

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

[113] Martín Arjovsky, Soumith Chintala and Léon Bottou. 'Wasserstein Generative Adversarial Networks'. In: *International Conference on Machine Learning*. 2017. URL: https://api.semanticscholar.org/CorpusID:2057420.

[114] Ishaan Gulrajani et al. 'Improved Training of Wasserstein GANs'. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 5769–5779. ISBN: 9781510860964.

[115] Chris Donahue, Julian McAuley and Miller Puckette. 'Adversarial Audio Synthesis'. In: *International Conference on Learning Representations*. 2018. URL: https://api.semanticscholar.org/CorpusID:52890982.

[116] Alec Radford, Luke Metz and Soumith Chintala. 'Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks'. In: *CoRR* abs/1511.06434 (2015). URL: https://api.semanticscholar.org/CorpusID:11758569.

[117] Ryan Prenger, Rafael Valle and Bryan Catanzaro. 'Waveglow: A flow-based generative network for speech synthesis'. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 3617–3621.

[118] Sungwon Kim et al. 'FloWaveNet : A Generative Flow for Raw Audio'. In: *International Conference on Machine Learning*. 2018. URL: https://api.semanticscholar.org/CorpusID:53249313.

[119] Sergi Andreu and Monica Villanueva Aylagas. 'Neural Synthesis of Sound Effects Using Flow-Based Deep Generative Models'. In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Vol. 18. 1. 2022, pp. 2–9.

[120] Wei Ping et al. 'Waveflow: A compact flow-based model for raw audio'. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7706–7716.

[121] Jascha Sohl-Dickstein et al. 'Deep unsupervised learning using nonequilibrium thermodynamics'. In: *International conference on machine learning*. PMLR. 2015, pp. 2256–2265.

[122] Robin Rombach et al. ‘High-Resolution Image Synthesis with Latent Diffusion Models’. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 10674–10685. URL: https://api.semanticscholar.org/CorpusID:245335280.

[123] Aditya Ramesh et al. ‘Hierarchical Text-Conditional Image Generation with CLIP Latents’. In: *arXiv preprint arXiv:2204.06125* (2022). URL: https://arxiv.org/abs/2204.06125.

[124] *MidJourney*. https://www.midjourney.com/. Accessed: 2025-01-31. 2023.

[125] Zhifeng Kong et al. ‘DiffWave: A Versatile Diffusion Model for Audio Synthesis’. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=a-xFK8Ymz5J.

[126] Nanxin Chen et al. ‘WaveGrad: Estimating Gradients for Waveform Generation’. In: *International Conference on Learning Representations*. 2021. URL: https://www.semanticscholar.org/paper/WaveGrad%3A-Estimating-Gradients-for-Waveform-Chen-Zhang/685af6d2bcdff7170574643b2c5ab4fb

[127] Anders Boesen Lindbo Larsen et al. ‘Autoencoding beyond pixels using a learned similarity metric’. In: *International conference on machine learning*. PMLR. 2016, pp. 1558–1566.

[128] Antoine Caillon and Philippe Esling. ‘Streamable Neural Audio Synthesis With Non-Causal Convolutions’. In: *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. 2022. URL: https://api.semanticscholar.org/CorpusID:248177757.

[129] Ninon Devis et al. ‘Continuous descriptor-based control for deep audio synthesis’. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[130] Zih-Syuan Yang and Jason Hockman. ‘A Plugin for Neural Audio Synthesis of Impact Sound Effects’. In: *Proceedings of the 18th International Audio Mostly Conference*. AM ’23. Edinburgh, United Kingdom: Association for Computing Machinery, 2023,

pp. 143–146. ISBN: 9798400708183. DOI: 10.1145/3616195.3616221. URL: https://doi.org/10.1145/3616195.3616221.

[131] Philippe Esling et al. 'Flow Synthesizer: Universal Audio Synthesizer Control with Normalizing Flows'. In: *Applied Sciences* (2019). URL: https://api.semanticscholar.org/CorpusID:210905230.

[132] Haohe Liu et al. 'Audioldm 2: Learning holistic audio generation with self-supervised pretraining'. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2024).

[133] Anton Lundberg. 'Data-Driven Procedural Audio: Procedural Engine Sounds Using Neural Audio Synthesis'. en. MA thesis. KTH School of Electrical Engineering and Computer Science (EECS), 2020.

[134] T.S. Verma and T.H.Y. Meng. 'An analysis/synthesis tool for transient signals that allows a flexible sines+transients+noise model for audio'. en. In: *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*. Vol. 6. Seattle, WA, USA: IEEE, 1998, pp. 3573–3576. ISBN: 978-0-7803-4428-0. DOI: 10.1109/ICASSP.1998.679647. URL: http://ieeexplore.ieee.org/document/679647/ (visited on 11/08/2023).

[135] Ben Hayes, Charalampos Saitis and George Fazekas. 'NEURAL WAVESHAPING SYNTHESIS'. en. In: (2021), p. 8.

[136] Adrien Bitton, Philippe Esling and Tatsuya Harada. 'Neural Granular Sound Synthesis'. en. In: (), p. 6.

[137] Aäron van den Oord et al. 'Parallel WaveNet: Fast High-Fidelity Speech Synthesis'. In: *International Conference on Machine Learning*. 2017. URL: https://api.semanticscholar.org/CorpusID:27706557.

[138] Nal Kalchbrenner et al. 'Efficient neural audio synthesis'. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2410–2419.

[139] Hendrik Purwins et al. 'Deep Learning for Audio Signal Processing'. In: *IEEE Journal of Selected Topics in Signal Processing* 13.2 (May 2019), pp. 206–219. DOI: 10.1109/jstsp.2019.2908700. URL: https://doi.org/10.1109%2Fjstsp.2019.2908700.

[140] Simon Welker, Julius Richter and Timo Gerkmann. 'Speech Enhancement with Score-Based Generative Models in the Complex STFT Domain'. In: *Interspeech*. 2022. URL: https://api.semanticscholar.org/CorpusID:247839443.

[141] Andrés Marafioti et al. 'Adversarial generation of time-frequency features with application in audio synthesis'. In: *International conference on machine learning*. PMLR. 2019, pp. 4352–4362.

[142] Zdenek Prusa, Péter Balázs and Peter Lempel Søndergaard. 'A Noniterative Method for Reconstruction of Phase From STFT Magnitude'. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25 (2017), pp. 1154–1164.

[143] Sang-gil Lee et al. 'BigVGAN: A Universal Neural Vocoder with Large-Scale Training'. In: *International Conference on Learning Representations*. 2022. URL: https://api.semanticscholar.org/CorpusID:249538510.

[144] Ryuichi Yamamoto, Eunwoo Song and Jae-Min Kim. 'Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram'. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 6199–6203.

[145] Hakan Erdogan et al. 'Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks'. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015, pp. 708–712. DOI: 10.1109/ICASSP.2015.7178061.

[146] Shichao Hu et al. 'Phase-Aware Music Super-Resolution Using Generative Adversarial Networks'. In: *Interspeech*. 2020. URL: https://api.semanticscholar.org/CorpusID:222272084.

[147] Boualem Boashash. *Time-frequency signal analysis and processing: a comprehensive reference*. Academic press, 2015.

[148] Ervin Sejdic, Igor Djurovic and LJubisa Stankovic. 'Quantitative Performance Analysis of Scalogram as Instantaneous Frequency Estimator'. In: *IEEE Transactions on Signal Processing* 56.8 (2008), pp. 3837–3845. DOI: 10.1109/TSP.2008.924856.

[149] Antonios K Alexandridis and Achilleas D Zapranis. 'Wavelet neural networks: A practical guide'. In: *Neural Networks* 42 (2013), pp. 1–27.

[150] Pedro Narváez and Winston S Percybrooks. 'Synthesis of normal heart sounds using generative adversarial networks and empirical wavelet transform'. In: *Applied Sciences* 10.19 (2020), p. 7003.

[151] Shlomo Dubnov et al. 'Synthesis of sound textures by learning and resampling of wavelet trees'. In: *IEEE Computer Graphics and Applications* 22.4 (2002), pp. 38–48.

[152] Stephane Mallat. *A wavelet tour of signal processing*. 1999.

[153] Daniel Griffin and Jae Lim. 'Signal estimation from modified short-time Fourier transform'. In: *IEEE Transactions on acoustics, speech, and signal processing* 32.2 (1984), pp. 236–243.

[154] Anastasia Natsiou and Seán O'Leary. 'Audio representations for deep learning in sound synthesis: A review'. In: *2021 IEEE/ACS 18th International Conference on Computer Systems and Applications (AICCSA)* (2021), pp. 1–8. URL: https://api.semanticscholar.org/CorpusID:245827795.

[155] Neil Zeghidour et al. 'SoundStream: An End-to-End Neural Audio Codec'. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 30 (2021), pp. 495–507. URL: https://api.semanticscholar.org/CorpusID:236149944.

[156] Yuan Gong, Yu-An Chung and James R. Glass. 'AST: Audio Spectrogram Transformer'. In: *Interspeech*. 2021. URL: https://api.semanticscholar.org/CorpusID:233024831.

[157] Romain Serizel et al. 'Acoustic Features for Environmental Sound Analysis'. In: Jan. 2018, pp. 71–101. ISBN: 978-3-319-63449-4. DOI: 10.1007/978-3-319-63450-0_4.

[158] Won Jang et al. 'UnivNet: A Neural Vocoder with Multi-Resolution Spectrogram Discriminators for High-Fidelity Waveform Generation'. In: *Interspeech*. 2021. URL: https://api.semanticscholar.org/CorpusID:235435945.

[159] Yunyi Liu and Craig Jin. 'Conditional Sound Effects Generation with Regularized WGAN'. In: *20th Sound and Music Computing Conference, Stockholm*. 2023.

[160] Jonathan Shen et al. ‘Natural tts synthesis by conditioning wavenet on mel spectrogram predictions’. In: *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2018, pp. 4779–4783.

[161] Yunyi Liu and Craig Jin. ‘Impact on quality and diversity from integrating a reconstruction loss into neural audio synthesis’. In: *Acoustical Society of America Journal* 154 (Oct. 2023), A99–A99. DOI: 10.1121/10.0022922.

[162] Shawn Hershey et al. ‘CNN architectures for large-scale audio classification’. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2016), pp. 131–135. URL: https://api.semanticscholar.org/CorpusID:8810481.

[163] Irina Higgins et al. ‘beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework’. In: *International Conference on Learning Representations*. 2016. URL: https://api.semanticscholar.org/CorpusID:46798026.

[164] Andrey Voynov and Artem Babenko. ‘Unsupervised discovery of interpretable directions in the gan latent space’. In: *International conference on machine learning*. PMLR. 2020, pp. 9786–9796.

[165] Robin Scheibler et al. ‘Foley sound synthesis with a class-conditioned latent diffusion model’. In: *2023 Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE 2023)*. 2023.

[166] Hugo Caracalla and Axel Roebel. ‘Sound texture synthesis using convolutional neural networks’. In: *International Conference on Digital Audio Effects (DAFx)*. 2019.

[167] Jong Kim et al. ‘CREPE: A Convolutional Representation for Pitch Estimation’. In: *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on* (Apr. 2018).

[168] Alain de Cheveigné and Hideki Kawahara. ‘YIN, a fundamental frequency estimator for speech and music.’ In: *The Journal of the Acoustical Society of America* 111 4 (2002), pp. 1917–30. URL: https://api.semanticscholar.org/CorpusID:1607434.

[169] Andrew Owens et al. ‘Visually Indicated Sounds’. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2405–2413. DOI: 10.1109/CVPR.2016.264.

[170] Adri’an Barahona-R’ios and Tom Collins. ‘NoiseBandNet: Controllable Time-Varying Neural Synthesis of Sound Effects Using Filterbanks’. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 32 (2023), pp. 1573–1585. URL: https://api.semanticscholar.org/CorpusID:259937076.

[171] Javier Nistal, Stefan Lattner and Gaël Richard. ‘DrumGAN: Synthesis of Drum Sounds With Timbral Feature Conditioning Using Generative Adversarial Networks’. In: *International Society for Music Information Retrieval Conference*. 2020. URL: https://api.semanticscholar.org/CorpusID:221340716.

[172] Philippe Esling, Axel Chemla-Romeu-Santos and Adrien Bitton. ‘Generative timbre spaces with variational audio synthesis’. In: *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. 2018, pp. 175–181.

[173] Guillaume Lample et al. ‘Fader Networks: Manipulating Images by Sliding Attributes’. In: *Neural Information Processing Systems*. Vol. abs/1706.00409. 2017. URL: https://api.semanticscholar.org/CorpusID:27009824.

[174] Lvmin Zhang, Anyi Rao and Maneesh Agrawala. ‘Adding Conditional Control to Text-to-Image Diffusion Models’. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 3813–3824. DOI: 10.1109/ICCV51070.2023.00355.

[175] Riccardo Fosco Gramaccioni et al. *Stable-V2A: Synthesis of Synchronized Sound Effects with Temporal and Semantic Controls*. 2025. arXiv: 2412.15023 [cs.SD]. URL: https://arxiv.org/abs/2412.15023.

[176] Heng Wang et al. ‘V2a-mapper: A lightweight solution for vision-to-audio generation by connecting foundation models’. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 14. 2024, pp. 15492–15501.

[177] Yi Ren et al. ‘FastSpeech 2: Fast and High-Quality End-to-End Text to Speech’. In: *International Conference on Learning Representations*. 2021.

[178] Vladimir Popov et al. ‘GradTTS: A Diffusion Probabilistic Model for Text-to-Speech’. In: *International Conference on Machine Learning*. 2021, pp. 8599–8608.

[179] Xu Tan et al. ‘NaturalSpeech: End-to-End Text-to-Speech Synthesis With Human-Level Quality’. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46 (2022), pp. 4234–4245. URL: https://api.semanticscholar.org/CorpusID:248572487.

[180] Andrea Agostinelli et al. ‘Musiclm: Generating music from text’. In: *arXiv preprint arXiv:2301.11325* (2023).

[181] Xiao Liu et al. ‘Self-supervised learning: Generative or contrastive’. In: *IEEE transactions on knowledge and data engineering* 35.1 (2021), pp. 857–876.

[182] Prannay Khosla et al. ‘Supervised contrastive learning’. In: *Advances in neural information processing systems* 33 (2020), pp. 18661–18673.

[183] Zalán Borsos et al. ‘Audiolm: a language modeling approach to audio generation’. In: *IEEE/ACM transactions on audio, speech, and language processing* 31 (2023), pp. 2523–2533.

[184] H. Chen et al. ‘VGGSound: A large-scale audio-visual dataset’. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 721–725.

[185] Jort F. Gemmeke et al. ‘Audio Set: An ontology and human-labeled dataset for audio events’. In: *Proc. IEEE ICASSP 2017*. New Orleans, LA, 2017.

[186] K. Drossos, S. Lipping and T. Virtanen. ‘Clotho: An audio captioning dataset’. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 736–740.

[187] Jiquan Ngiam et al. ‘Multimodal Deep Learning’. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. ICML’11. Bellevue, Washington, USA: Omnipress, 2011, pp. 689–696. ISBN: 9781450306195.

[188] Lele Chen et al. ‘Deep cross-modal audio-visual generation’. In: *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*. 2017, pp. 349–357.

[189] Wangli Hao, Zhaoxiang Zhang and He Guan. 'Cmcgan: A uniform framework for cross-modal visual-audio mutual generation'. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[190] Vladimir Iashin and Esa Rahtu. 'Taming visually guided sound generation'. In: *British Machine Vision Conference (BMVC)*. 2021.

[191] Peihao Chen et al. 'Generating visually aligned sound from videos'. In: *IEEE Transactions on Image Processing* 29 (2020), pp. 8292–8302.

[192] Roy Sheffer and Yossi Adi. 'I Hear Your True Colors: Image Guided Audio Generation'. In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2022), pp. 1–5. URL: https://api.semanticscholar.org/CorpusID:253384594.

[193] Ce Zhou et al. 'A comprehensive survey on pretrained foundation models: A history from bert to chatgpt'. In: *International Journal of Machine Learning and Cybernetics* (2024), pp. 1–65.

[194] Mao Guo, Chenghu Zhou and Jiahang Liu. 'Jointly learning of visual and auditory: A new approach for RS image and audio cross-modal retrieval'. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 12.11 (2019), pp. 4644–4654.

[195] Meng Xu et al. 'Cmjrt: Cross-modal joint representation transformer for multimodal sentiment analysis'. In: *IEEE Access* 10 (2022), pp. 131671–131679.

[196] Alec Radford et al. 'Learning Transferable Visual Models From Natural Language Supervision'. In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 8748–8763. URL: http://proceedings.mlr.press/v139/radford21a.html.

[197] Simian Luo et al. 'Diff-Foley: Synchronized Video-to-Audio Synthesis with Latent Diffusion Models'. In: *Neural Information Processing Systems*. 2023. URL: https://api.semanticscholar.org/CorpusID:259309037.

[198] Sanchita Ghose and John Jeffrey Prevost. 'Autofoley: Artificial synthesis of synchronized sound tracks for silent videos with deep learning'. In: *IEEE Transactions on Multimedia* 23 (2020), pp. 1895–1907.

[199] Lucas Theis, Aäron van den Oord and Matthias Bethge. 'A note on the evaluation of generative models'. In: *CoRR* abs/1511.01844 (2015). URL: https://api.semanticscholar.org/CorpusID:2187805.

[200] Le Zhang et al. 'Disentangling human error from ground truth in segmentation of medical images'. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 15750–15762.

[201] Dale Reed. 'Capturing perceptual expertise: a sound equalization expert system'. In: *Knowledge-Based Systems* 14.1-2 (2001), pp. 111–118.

[202] Keiji Kawai et al. 'Personal evaluation structure of environmental sounds: experiments of subjective evaluation using subjects' own terms'. In: *Journal of sound and vibration* 277.3 (2004), pp. 523–533.

[203] Ashvala Vinay and Alexander Lerch. 'Evaluating Generative Audio Systems and Their Metrics'. In: *Proceedings of the 23rd International Society for Music Information Retrieval Conference, ISMIR 2022*. Bengaluru, India: ISMIR, Dec. 2022, pp. 858–865.

[204] Muhammad Ferjad Naeem et al. 'Reliable Fidelity and Diversity Metrics for Generative Models'. In: *International Conference on Machine Learning*. 2020. URL: https://api.semanticscholar.org/CorpusID:211259260.

[205] David A. Jaffe. 'Ten Criteria for Evaluating Synthesis Techniques'. In: *Computer Music Journal* 19.1 (1995), pp. 76–87. ISSN: 01489267, 15315169. URL: http://www.jstor.org/stable/3681301 (visited on 07/02/2025).

[206] Nelly Garcia-Sihuay, Yisu Zong and Joshua Reiss. 'Discerning real from synthetic: analysis and perceptual evaluation of sound effects'. In: *Audio Engineering Society Conference: AES 2024 International Audio for Games Conference*. Audio Engineering Society. 2024.

[207] Yisu Zong, Nelly Garcia-Sihuay and Joshua Reiss. 'A Machine learning method to evaluate and improve sound effects synthesis model design'. In: *Audio Engineering*

Society Conference: AES 2024 International Audio for Games Conference. Audio Engineering Society. 2024.

[208] Dongyang Dai et al. 'Learning Discriminative Features from Spectrograms Using Center Loss for Speech Emotion Recognition'. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019), pp. 7405–7409. URL: https://api.semanticscholar.org/CorpusID:145847880.

[209] Soha A Nossier et al. 'A comparative study of time and frequency domain approaches to deep learning based speech enhancement'. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8.

[210] Kevin Kilgour et al. 'Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms'. In: *Interspeech*. 2019. URL: https://api.semanticscholar.org/CorpusID:202725406.

[211] Martin Heusel et al. 'GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium'. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6629–6640. ISBN: 9781510860964.

[212] Alexandre D'efossez et al. 'High Fidelity Neural Audio Compression'. In: *ArXiv* abs/2210.13438 (2022). URL: https://api.semanticscholar.org/CorpusID:253097788.

[213] Javier Nistal et al. 'VQCPC-GAN: Variable-length adversarial audio synthesis using vector-quantized contrastive predictive coding'. In: *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE. 2021, pp. 116–120.

[214] Yoonjin Chung et al. *KAD: No More FAD! An Effective and Efficient Evaluation Metric for Audio Generation*. 2025. arXiv: 2502.15602 [cs.SD]. URL: https://arxiv.org/abs/2502.15602.

[215] Jiaming Cheng et al. 'A Deep Adaptation Network for Speech Enhancement: Combining a Relativistic Discriminator With Multi-Kernel Maximum Mean Discrepancy'.

In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 41–53. DOI: 10.1109/TASLP.2020.3036611.

[216] Eitan Richardson and Yair Weiss. 'On GANs and GMMs'. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada: Curran Associates Inc., 2018, pp. 5852–5863.

[217] Jen-Yu Liu et al. 'Unconditional Audio Generation with Generative Adversarial Networks and Cycle Regularization'. In: *Interspeech*. 2020. URL: https://api.semanticscholar.org/CorpusID:218674026.

[218] Flávio Ribeiro et al. 'CROWDMOS: An approach for crowdsourcing mean opinion score studies'. In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2011, pp. 2416–2419. DOI: 10.1109/ICASSP.2011.5946971.

[219] International Telecommunication Union. *Recommendation ITU-R BS.1534-2: Method for the subjective assessment of intermediate quality level of audio systems*. International Telecommunication Union. 2014. URL: https://www.itu.int/rec/R-REC-BS.1534-2-201406-I/en.

[220] Thilo Thiede et al. 'PEAQ-The ITU standard for objective measurement of perceived audio quality'. In: *Journal of the Audio Engineering Society* 48.1/2 (2000), pp. 3–29.

[221] Gregory Hickok. 'The architecture of speech production and the role of the phoneme in speech processing'. In: *Language, Cognition and Neuroscience* 29.1 (2014), pp. 2–20.

[222] Li-Chia Yang, Szu-Yu Chou and Yi-Hsuan Yang. 'MidiNet: A Convolutional Generative Adversarial Network for Symbolic-Domain Music Generation'. In: *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*. Ed. by Sally Jo Cunningham et al. 2017, pp. 324–331. URL: https://ismir2017.smcnus.org/wp-content/uploads/2017/10/226%5C_Paper.pdf.

[223] Boom Library. *Boom Library - High Quality Sound Effects*. Accessed: 2025-02-10. 2025. URL: https://www.boomlibrary.com/.

[224] Films for the Humanities & Sciences. *BBC Sound Effects Library*. Sound effects library. Princeton, N.J.: Films for the Humanities & Sciences, 1991.

[225] *Epidemic Sound - Royalty Free Music and Sound Effects*. https://www.epidemicsound.com/. Accessed: 2025-02-10.

[226] Juan Salamon, Chris Jacoby and Juan P. Bello. 'A Dataset and Taxonomy for Urban Sound Research'. In: *Proceedings of the 22nd ACM International Conference on Multimedia*. Orlando, USA, Nov. 2014.

[227] Keunwoo Choi et al. 'Foley Sound Synthesis at the DCASE 2023 Challenge'. In: *In arXiv e-prints: 2304.12521* (2023).

[228] Eduardo Fonseca et al. 'Learning Sound Event Classifiers from Web Audio with Noisy Labels'. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019), pp. 21–25. URL: https://api.semanticscholar.org/CorpusID:57572929.

[229] Konstantinos Drossos, Samuel Lipping and Tuomas Virtanen. 'Clotho: an Audio Captioning Dataset'. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 736–740. DOI: 10.1109/ICASSP40776.2020.9052990.

[230] Aswathy Madhu and Suresh Kumaraswamy. 'Data Augmentation Using Generative Adversarial Network for Environmental Sound Classification'. In: *2019 27th European Signal Processing Conference (EUSIPCO)*. 2019, pp. 1–5. DOI: 10.23919/EUSIPCO.2019.8902819.

[231] Jon Fagerström, Sebastian J. Schlecht and Vesa Välimäki. 'One-to-Many Conversion for Percussive Samples'. In: *2021 24th International Conference on Digital Audio Effects (DAFx)*. 2021, pp. 129–135. DOI: 10.23919/DAFx51585.2021.9768256.

[232] Shiguang Liu, Sijia Li and Haonan Cheng. 'Towards an end-to-end visual-to-raw-audio generation with GAN'. In: *IEEE Transactions on Circuits and Systems for Video Technology* 32.3 (2021), pp. 1299–1312.

[233] Chitralekha Gupta, Purnima Kamath and Lonce Wyse. 'Signal Representations for Synthesizing Audio Textures with Generative Adversarial Networks'. In: *CoRR*

abs/2103.07390 (2021). Sound and Music Computing: 2103.07390. URL: https://arxiv.org/abs/2103.07390.

[234] Javier Nistal, Stefan Lattner and Gael Richard. 'Comparing representations for audio synthesis using generative adversarial networks'. In: *2020 28th European Signal Processing Conference (EUSIPCO)*. IEEE. 2021, pp. 161–165.

[235] David Moffat, David Ronan and Joshua Reiss. 'Unsupervised Taxonomy of Sound Effects'. In: *20th International Conference on Digital Audio Effects (DAFx-17)*. Queen Mary, University of London. Edinburgh, UK, Apr. 2017.

[236] David Gerhard. 'Audio Signal Classification'. In: *2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)*. IEEE. 2018, pp. 175–180.

[237] Dhruv Jain et al. 'A Taxonomy of Sounds in Virtual Reality'. In: *Proceedings of the 2021 ACM Designing Interactive Systems Conference*. DIS '21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 160–170. ISBN: 9781450384766. DOI: 10.1145/3461778.3462106. URL: https://doi.org/10.1145/3461778.3462106.

[238] Joseph M. Antognini, Matt Hoffman and Ron J. Weiss. 'Audio Texture Synthesis with Random Neural Networks: Improving Diversity and Quality'. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 3587–3591. DOI: 10.1109/ICASSP.2019.8682598.

[239] Sound Ideas. *Sound Ideas – Royalty Free Sound Effects and Production Music*. https://www.sound-ideas.com/. Accessed: 26 February 2025.

[240] Soochan Lee, Junsoo Ha and Gunhee Kim. 'Harmonizing Maximum Likelihood with GANs for Multimodal Conditional Generation'. In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=HJxyAjRcFX.

[241] Jun-Yan Zhu et al. 'Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks'. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2242–2251. DOI: 10.1109/ICCV.2017.244.

[242] Yunyi Liu and Craig Jin. 'Conditional Sound Effects Generation with Regularized WGAN'. In: *Proceedings of the Sound and Music Computing Conference*. KMH

Royal College of Music in Stockholm. Stockholm, 2023, pp. 34–40. DOI: 10.5281/zenodo.8136568.

[243] Jordi Pons et al. 'Upsampling Artifacts in Neural Audio Synthesis'. In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2021, pp. 3005–3009. DOI: 10.1109/ICASSP39728.2021.9414913.

[244] Yunyi Liu, Craig Jin and David Gunawan. 'DDSP-SFX: Acoustically-guided sound effects generation with differentiable digital signal processing'. In: *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. 2023, pp. 216–221.

[245] Soonil Kwon. 'Voice-Driven Sound Effect Manipulation'. en. In: *International Journal of Human-Computer Interaction* 28.6 (June 2012), pp. 373–382. ISSN: 1044-7318, 1532-7590. DOI: 10.1080/10447318.2011.595359. URL: http://www.tandfonline.com/doi/abs/10.1080/10447318.2011.595359 (visited on 11/08/2023).

[246] Yuki Okamoto et al. 'Onoma-to-wave: Environmental Sound Synthesis from Onomatopoeic Words'. In: *APSIPA Transactions on Signal and Information Processing* 11.1 (2022), pp. -. DOI: 10.1561/116.00000049. URL: http://dx.doi.org/10.1561/116.00000049.

[247] Kohei Suzuki et al. 'Speak Like a Dog: Human to Non-human creature Voice Conversion'. In: *Proceedings of 2022 APSIPA Annual Summit and Conference*. APSIPA. Chiang Mai, Thailand, Nov. 2022.

[248] Riki Takizawa and Shigeyuki Hirai. 'Synthesis of Explosion Sounds from Utterance Voice of Onomatopoeia using Transformer'. en. In: *28th International Conference on Intelligent User Interfaces*. Sydney NSW Australia: ACM, Mar. 2023, pp. 87–90. ISBN: 9798400701078. DOI: 10.1145/3581754.3584139. URL: https://dl.acm.org/doi/10.1145/3581754.3584139 (visited on 11/08/2023).

[249] Adrián Barahona-Ríos and Tom Collins. 'NoiseBandNet: Controllable Time-Varying Neural Synthesis of Sound Effects Using Filterbanks'. In: *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 32 (Feb. 2024), pp. 1573–1585. ISSN: 2329-9290. DOI:

10.1109/TASLP.2024.3364616. URL: https://doi.org/10.1109/TASLP.2024.3364616.

[250] T.Q. Nguyen. 'Near-perfect-reconstruction pseudo-QMF banks'. In: *IEEE Transactions on Signal Processing* 42.1 (1994), pp. 65–76. DOI: 10.1109/78.258122.

[251] Jordie Shier et al. *Real-time Timbre Remapping with Differentiable DSP*. 2024. arXiv: 2407.04547 [cs.SD]. URL: https://arxiv.org/abs/2407.04547.

[252] Yotam Nitzan et al. 'Face identity disentanglement via latent space mapping'. In: *ACM Trans. Graph.* 39.6 (Nov. 2020). ISSN: 0730-0301. DOI: 10.1145/3414685.3417826. URL: https://doi.org/10.1145/3414685.3417826.

[253] Marius-Constantin Popescu et al. 'Multilayer perceptron and neural networks'. In: *WSEAS Transactions on Circuits and Systems* 8.7 (2009), pp. 579–588.

[254] Rahul Dey and Fathi M Salem. 'Gate-variants of gated recurrent unit (GRU) neural networks'. In: *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*. IEEE. 2017, pp. 1597–1600.

[255] Leonardo Fierro and Vesa Valimaki. 'Sitrano: A Matlab App for Sines-Transients-Noise Decomposition of Audio Signals'. In: *2021 24th International Conference on Digital Audio Effects (DAFx)*. 2021, pp. 73–80. DOI: 10.23919/DAFx51585.2021.9768212.

[256] Jonathan Driedger, Meinard Müller and Sascha Disch. 'Extending Harmonic-Percussive Separation of Audio Signals'. In: *International Society for Music Information Retrieval Conference*. 2014. URL: https://api.semanticscholar.org/CorpusID:9367555.

[257] Diederik P. Kingma and Jimmy Ba. 'Adam: A Method for Stochastic Optimization'. In: *International Conference on Learning Representations* abs/1412.6980 (2014). URL: https://api.semanticscholar.org/CorpusID:6628106.

[258] C. Gupta et al. 'Towards Controllable Audio Texture Morphing'. In: *Proceedings of the 48th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2023)* (4th–10th June 2023). Co-first authors marked with * in the source: C. Gupta and P. Kamath. Rhodes Island, Greece: IEEE, 2023.

[259] Aäron van den Oord et al. 'Conditional image generation with PixelCNN decoders'. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 4797–4805. ISBN: 9781510838819.

[260] Zhiting Hu et al. 'Deep generative models with learnable knowledge constraints'. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada: Curran Associates Inc., 2018, pp. 10522–10533.

[261] Eszter Vértes and Maneesh Sahani. 'Flexible and accurate inference and learning for deep generative models'. In: *Neural Information Processing Systems*. 2018. URL: https://api.semanticscholar.org/CorpusID:44086783.

[262] Yunchuan Guan et al. 'Hierarchical Meta-Learning with Hyper-Tasks for Few-Shot Learning'. In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. CIKM '23. <conf-loc>, <city>Birmingham</city>, <country>United Kingdom</country>, </conf-loc>: Association for Computing Machinery, 2023, pp. 587–596. ISBN: 9798400701245. DOI: 10.1145/3583780.3614911. URL: https://doi.org/10.1145/3583780.3614911.

[263] Heewoo Jun et al. 'Distribution augmentation for generative modeling'. In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org, 2020.

[264] Xin Ding et al. 'Continuous Conditional Generative Adversarial Networks: Novel Empirical Losses and Label Input Mechanisms'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45 (2020), pp. 8143–8158. URL: https://api.semanticscholar.org/CorpusID:237355091.

[265] Amin Heyrani Nobari, Wei Chen and Faez Ahmed. 'PcDGAN: A Continuous Conditional Diverse Generative Adversarial Network For Inverse Design'. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. KDD '21. Virtual Event, Singapore: Association for Computing Machinery, 2021, pp. 606–616. ISBN: 9781450383325. DOI: 10.1145/3447548.3467414. URL: https://doi.org/10.1145/3447548.3467414.

[266] Chang-Bin Zhang et al. ‘Delving Deep Into Label Smoothing’. In: *IEEE Transactions on Image Processing* 30 (2020), pp. 5984–5996. URL: https://api.semanticscholar.org/CorpusID:227162602.

[267] Ethan Perez et al. ‘FiLM: visual reasoning with a general conditioning layer’. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI’18/IAAI’18/EAAI’18. New Orleans, Louisiana, USA: AAAI Press, 2018. ISBN: 978-1-57735-800-8.

[268] Adrian Barahona Rios and Tom Collins. ‘SpecSinGAN: Sound Effect Variation Synthesis Using Single-Image GANs’. In: *Proceedings of the Sound and Music Computing Conference*. 2022.

[269] Y. Lecun et al. ‘Gradient-based learning applied to document recognition’. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[270] Kıvanç Tatar, Daniel Bisig and Philippe Pasquier. ‘Latent Timbre Synthesis’. In: *Neural Computing and Applications* 33 (Jan. 2021), pp. 1–18. DOI: 10.1007/s00521-020-05424-2.

[271] Christian Szegedy et al. ‘Rethinking the Inception Architecture for Computer Vision’. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 2818–2826. URL: https://api.semanticscholar.org/CorpusID:206593880.

[272] J.F. Schouten. ‘The perception of timbre’. English. In: *The 6th international congress on acoustics, Tokyo, Japan, August 21-28, 1968*. 1968, pp. 35–44.

[273] David Moffat, Rod Selfridge and Joshua D Reiss. ‘Sound effect synthesis’. In: *Foundations in Sound Design for Interactive Media*. Routledge, 2019, pp. 274–299.

[274] Luca Turchet et al. ‘Physically Based Sound Synthesis and Control of Footsteps Sounds’. In: *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. 2010. URL: https://api.semanticscholar.org/CorpusID:35113931.

[275] Aaron Valero Puche and Sukhan Lee. ‘Caesynth: Real-Time Timbre Interpolation and Pitch Control with Conditional Autoencoders’. In: *2021 IEEE 31st International*

Workshop on Machine Learning for Signal Processing (MLSP). 2021, pp. 1–6. DOI: 10.1109/MLSP52302.2021.9596414.

[276] Jan Zuiderveld, Marco Federici and Erik J Bekkers. 'Towards Lightweight Controllable Audio Synthesis with Conditional Implicit Neural Representations'. In: *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*. 2021. URL: https://openreview.net/forum?id=-e7kA-HhM3.

[277] Nicolas Jonason. *The control-synthesis approach for making expressive and controllable neural music synthesizers*. 2020.

[278] Kun Su et al. 'Physics-Driven Diffusion Models for Impact Sound Synthesis from Videos'. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 9749–9759. DOI: 10.1109/CVPR52729.2023.00940.

[279] Eric Grinstein et al. 'Audio Style Transfer'. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2017), pp. 586–590. URL: https://api.semanticscholar.org/CorpusID:23919379.

[280] Christian Steinmetz, Nicholas Bryan and Joshua Reiss. 'Style Transfer of Audio Effects with Differentiable Signal Processing'. In: *Journal of the Audio Engineering Society* 70 (Nov. 2022), pp. 708–721. DOI: 10.17743/jaes.2022.0025.

[281] Tero Karras, Samuli Laine and Timo Aila. 'A Style-Based Generator Architecture for Generative Adversarial Networks'. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), pp. 4396–4405. URL: https://api.semanticscholar.org/CorpusID:54482423.

[282] Antoine Lavault, Axel Roebel and Matthieu Voiry. 'StyleWaveGAN: Style-based synthesis of drum sounds using generative adversarial networks for higher audio quality'. In: *2022 30th European Signal Processing Conference (EUSIPCO)*. 2022, pp. 234–238. DOI: 10.23919/EUSIPCO55093.2022.9909880.

[283] Yunyi Liu and Craig Jin. 'ICGAN: An implicit conditioning method for interpretable feature control of neural audio synthesis'. In: *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. 2024.

[284] Danzel Serrano. 'A Neural Analysis–Synthesis Approach to Learning Procedural Audio Models'. MA thesis. New Jersey Institute of Technology, Department of Computer Science, 2022.

[285] Haokun Tian, Stefan Lattner and Charalampos Saitis. *Assessing the Alignment of Audio Representations with Timbre Similarity Ratings*. 2025. arXiv: 2507.07764 [cs.SD]. URL: https://arxiv.org/abs/2507.07764.

[286] P. C. Mahalanobis. 'On the Generalised Distance in Statistics'. In: *Sankhya A* 80.1 (2018). Reprint of: Mahalanobis, P.C. (1936), pp. 1–7. ISSN: 0976-8378. DOI: 10.1007/s13171-019-00164-5.

[287] Riccardo Simionato and Stefano Fasciani. *Sine, Transient, Noise Neural Modeling of Piano Notes*. 2024. arXiv: 2409.06513 [cs.SD]. URL: https://arxiv.org/abs/2409.06513.

[288] Frederic Font, Gerard Roma and Xavier Serra. 'Freesound technical demo'. In: *Proceedings of the 21st ACM international conference on Multimedia* (2013). URL: https://api.semanticscholar.org/CorpusID:28550242.

[289] Douglas C. Montgomery, Elizabeth A. Peck and G. Geoffrey Vining. *Introduction to Linear Regression Analysis*. 5th ed. Hoboken, NJ: Wiley, 2012. ISBN: 978-0-470-54281-1.

[290] Bowen Zhang et al. 'Audio Deepfake Detection: What Has Been Achieved and What Lies Ahead'. In: *Sensors* 25.7 (2025). ISSN: 1424-8220. DOI: 10.3390/s25071989. URL: https://www.mdpi.com/1424-8220/25/7/1989.

[291] Yipin Zhou et al. 'Visual to Sound: Generating Natural Sound for Videos in the Wild'. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2017), pp. 3550–3558. URL: https://api.semanticscholar.org/CorpusID:32701102.

[292] Adobe. *Generate sound effects using text prompts (Firefly, beta)*. https://helpx.adobe.com/firefly/work-with-audio-and-video/work-with-audio/text-to-sound-effects.html. 2025.

[293] Peiwen Sun et al. 'Both Ears Wide Open: Towards Language-Driven Spatial Audio Generation'. In: *International Conference on Learning Representations (ICLR)*. ICLR

2025 (spotlight). 2025. arXiv: 2410.10676. URL: https://arxiv.org/abs/2410.10676.

[294] Bhavika Devnani et al. 'Learning Spatially-Aware Language and Audio Embeddings'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. NeurIPS 2024. 2024. arXiv: 2409.11369. URL: https://arxiv.org/abs/2409.11369.

[295] Mojtaba Heydari et al. 'ImmerseDiffusion: A Generative Spatial Audio Latent Diffusion Model'. In: *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Also published at ICASSP 2025. 2025. arXiv: 2410.14945. URL: https://arxiv.org/abs/2410.14945.

[296] Earl B Hunt. *Artificial intelligence*. Academic Press, 2014.

[297] Ethem Alpaydin. *Machine learning*. MIT press, 2021.

[298] David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams. 'Learning representations by back-propagating errors'. In: *Nature* 323 (1986), pp. 533–536. URL: https://api.semanticscholar.org/CorpusID:205001834.

[299] Yann LeCun, Yoshua Bengio and Geoffrey Hinton. 'Deep learning'. In: *nature* 521.7553 (2015), pp. 436–444.

[300] Maria MP Petrou and Costas Petrou. *Image processing: the fundamentals*. John Wiley & Sons, 2010.

[301] Jian Wu et al. 'On Decoder-Only Architecture For Speech-to-Text and Large Language Model Integration'. In: *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)* (2023), pp. 1–8. URL: https://api.semanticscholar.org/CorpusID:259501685.

[302] Jade Copet et al. 'Simple and controllable music generation'. In: *Advances in Neural Information Processing Systems* 36 (2024).

[303] Kurt Hornik, Maxwell B. Stinchcombe and Halbert L. White. 'Multilayer feedforward networks are universal approximators'. In: *Neural Networks* 2 (1989), pp. 359–366. URL: https://api.semanticscholar.org/CorpusID:2757547.

[304] Yoonjin Chung, Junwon Lee and Juhan Nam. 'T-Foley: A Controllable Waveform-Domain Diffusion Model for Temporal-Event-Guided Foley Sound Synthesis'. In: *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal*

Processing (ICASSP). 2024, pp. 6820–6824. DOI: 10.1109/ICASSP48485.2024.10447380.

[305] Yann LeCun et al. 'Gradient-based learning applied to document recognition'. In: *Proc. IEEE* 86 (1998), pp. 2278–2324. URL: https://api.semanticscholar.org/CorpusID:14542261.

[306] S. K. Verbitskiy and Viacheslav Vyshegorodtsev. 'ERANNs: Efficient Residual Audio Neural Networks for Audio Pattern Recognition'. In: *Pattern Recognit. Lett.* 161 (2021), pp. 38–44. URL: https://api.semanticscholar.org/CorpusID:235313953.

[307] Shuo-yiin Chang and Nelson Morgan. 'Robust CNN-based speech recognition with Gabor filter kernels'. In: *Interspeech*. 2014. URL: https://api.semanticscholar.org/CorpusID:9498408.

[308] Dimitri Palaz, Mathew Magimai-Doss and Ronan Collobert. 'Analysis of CNN-based speech recognition system using raw speech as input'. In: *Interspeech*. 2015. URL: https://api.semanticscholar.org/CorpusID:2610591.

[309] Tursunov Anvarjon, Mustaqeem and Soonil Kwon. 'Deep-Net: A Lightweight CNN-Based Speech Emotion Recognition System Using Deep Frequency Features'. In: *Sensors (Basel, Switzerland)* 20 (2020). URL: https://api.semanticscholar.org/CorpusID:221748924.

[310] Alex Graves and Alex Graves. 'Long short-term memory'. In: *Supervised sequence labelling with recurrent neural networks* (2012), pp. 37–45.

[311] Sercan Ö. Arik et al. 'Deep voice: real-time neural text-to-speech'. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 195–204.

[312] Ashish Vaswani et al. 'Attention is All you Need'. In: *Neural Information Processing Systems*. 2017. URL: https://api.semanticscholar.org/CorpusID:13756489.

[313] Shenggui Li et al. 'Sequence Parallelism: Long Sequence Training from System Perspective'. In: *Annual Meeting of the Association for Computational Linguistics*. 2021. URL: https://api.semanticscholar.org/CorpusID:246017095.

[314] Federico Piovesan. 'EXPLORING TRANSFORMER MODEL FOR ACOUSTIC SCENE CLASSIFICATION'. PhD thesis. Politecnico di Torino, 2024.

[315] Naihan Li et al. 'Neural Speech Synthesis with Transformer Network'. In: *AAAI Conference on Artificial Intelligence*. 2018. URL: https://api.semanticscholar.org/CorpusID:59413863.

[316] Yibin Zheng et al. 'Improving End-to-End Speech Synthesis with Local Recurrent Neural Network Enhanced Transformer'. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 6734–6738. DOI: 10.1109/ICASSP40776.2020.9054148.

[317] Hyeong Rae Ihm et al. 'Reformer-TTS: Neural Speech Synthesis with Reformer Network'. In: *Interspeech*. 2020. URL: https://api.semanticscholar.org/CorpusID:226205647.

[318] Cheng-Zhi Anna Huang et al. 'Music Transformer: Generating Music with Long-Term Structure'. In: *International Conference on Learning Representations*. 2018. URL: https://api.semanticscholar.org/CorpusID:54477714.

[319] Masanori Morise, Fumiya Yokomori and Kenji Ozawa. 'WORLD: A Vocoder-Based High-Quality Speech Synthesis System for Real-Time Applications'. In: *IEICE Trans. Inf. Syst.* 99-D (2016), pp. 1877–1884. URL: https://api.semanticscholar.org/CorpusID:3570465.

[320] Alexey Dosovitskiy et al. 'An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale'. In: *International Conference on Learning Representations*. 2021.

[321] Pierre-François Verhulst. 'Notice sur la loi que la population poursuit dans son accroissement'. In: *Correspondance mathématique et physique* 10 (1838), pp. 113–121.

[322] AF Agarap. 'Deep learning using rectified linear units (relu)'. In: *arXiv preprint arXiv:1803.08375* (2018).

[323] Andrew L Maas, Awni Y Hannun, Andrew Y Ng et al. 'Rectifier nonlinearities improve neural network acoustic models'. In: *Proc. icml*. Vol. 30. 1. Atlanta, GA. 2013, p. 3.

[324] Ludwig Boltzmann. ‘Studien über das Gleichgewicht der lebendigen Kraft zwischen bewegten materiellen Punkten’. In: *Wiener Berichte* 58 (1868), pp. 517–560.

[325] Josiah Willard Gibbs. *Elementary Principles in Statistical Mechanics*. Yale University Press, 1902.

[326] Chae Young Lee et al. ‘Conditional WaveGAN’. In: *ArXiv* abs/1809.10636 (2018). URL: https://api.semanticscholar.org/CorpusID:52880651.

[327] Sebastian Ruder. ‘An overview of gradient descent optimization algorithms’. In: *arXiv preprint arXiv:1609.04747* (2016).

[328] Geoffrey Hinton. *Neural Networks for Machine Learning, Lecture 6a: Overview of mini-batch gradient descent*. Coursera. 2012.

[329] Kaiming He et al. ‘Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification’. In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 1026–1034. URL: https://api.semanticscholar.org/CorpusID:13740328.

[330] Xavier Glorot and Yoshua Bengio. ‘Understanding the difficulty of training deep feedforward neural networks’. In: *International Conference on Artificial Intelligence and Statistics*. 2010. URL: https://api.semanticscholar.org/CorpusID:5575601.

1 Fundamentals of Machine Learning

1.1 Background

Artificial Intelligence (AI) [296] encompasses a broad range of computational techniques and theories aimed at mimicking human cognitive functions. Machine Learning (ML) [297] is a critical subset of AI, focused on the development of algorithms and statistical models that systems use to perform specific tasks without explicit instructions. Instead, these models improve their performance based on exposure to data. Machine learning is fundamentally about the creation and study of algorithms that can learn from and make predictions or decisions based on data. Such algorithms operate by building a model from example inputs and using the structured model to make data-driven predictions or decisions, rather than following strict static program instructions.

Central to these algorithms are neural networks, which consist of layers of interconnected nodes or neurons, where each connection, or edge, has an associated weight and bias that adjusts as learning proceeds via processes such as backpropagation [298]. These weights and biases are fundamental components of neural networks, representing the knowledge that the network has learned during training. The concept of learning in neural networks is facilitated through the adjustment of weights and biases during the training process. These parameters are adjusted through backpropagation [298], which iteratively modifies the weights and biases to minimize the difference between the predicted output and the actual output, a process often coupled with a gradient descent optimization method to find the optimal parameters.

As the complexity of tasks increases, the limitations of shallow neural networks become apparent, necessitating the evolution into more sophisticated architectures. This is where deep learning (DL) [299] comes into play, utilizing architectures with multiple hidden layers that enable higher levels of abstraction and representation learning. Deep learning, a subset of machine learning, extends these foundational principles by employing deep neural networks—structures with multiple layers of neurons that enable the capture and representation of complex and high-level data features.

The layers in deep neural networks allow for the learning of features at multiple levels of abstraction, which is critical for both visual and audio applications. In image recognition [300], for instance, the first few layers might identify edges and textures, intermediate layers discern patterns and parts of objects, and deeper layers interpret these features to recognize whole objects and scenes. Similarly, in audio processing, early layers may learn to detect basic sound features such as pitch and volume, while intermediate layers might recognize complex sound patterns like spectral distributions, and the deepest layers could interpret entire soundscapes or understand linguistic content. This layering mimics a form of hierarchical learning that is somewhat analogous to cognitive processes in the human brain, enabling the handling of increasingly complex data structures from raw input to abstract concepts.

The capabilities of deep learning to automatically learn rich and representative features have revolutionized audio generation and sound synthesis. Deep learning models are particularly adept at capturing the complex patterns necessary for generating realistic and diverse audio content, from music composition to speech synthesis. However, the success of these models often hinges on the availability of large-scale audio datasets to train these sophisticated networks. For example, modern sound generative models [301, 302] require extensive data to produce high-fidelity audio, as they learn to predict audio samples directly from raw waveforms. This reliance on vast amounts of data not only demands significant computational resources but also necessitates expertise in tuning network architectures and training procedures. These requirements underscore a trade-off between the sophistication of audio synthesis capabilities and their accessibility, as developing and training such powerful models are typically resource-intensive tasks.

In the following sections, I will delve deeper into the specific mechanisms of neural operations, explore various activation functions that contribute to the non-linearity of decision functions in neural networks, and discuss the critical role of loss functions and optimization techniques in refining network performance. The exploration of deep learning will continue with a focus on how these methods have transformed the landscape of artificial intelligence by enabling models that learn from and generate data in ways that were previously unattainable.

1.2 Neural Network Layers

1.2.1 Linear Layers

Linear layers, or fully connected layers, are the simplest form of neural network layers. Each neuron in a linear layer is connected to all neurons in the previous layer. The mathematical operation in a linear layer is defined by:

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{.1}$$

where $\mathbf{x}$ is the input vector, $\mathbf{W}$ represents the weight matrix, $\mathbf{b}$ is the bias vector, and $\mathbf{y}$ is the output vector. Linear layers are fundamental for transforming input data by learning optimal weights and biases through training. As they are linear operations, linear layers are typically combined with non-linear activation functions (introduced in the next section) to add complexity. They are versatile and can be used in any neural network model that requires a dense connection between layers [298]. In audio generation tasks, linear layers can often be found in the final decision layer of a neural network architecture. They map the high-level features extracted by previous layers to the final output, such as class scores or predicted audio samples.

1.2.2 Multi-Layer Perceptrons

Multi-Layer Perceptrons [253], or MLPs, can be seen as a stack of linear layers consisting of multiple layers of neurons, each fully connected to all neurons in the previous layer. MLPs are capable of learning non-linear functions by using non-linear activation functions between linear layers. They are widely used for tasks requiring complex pattern recognition such as classification and regression, including simpler forms of audio classification where temporal dynamics are less significant [303]. MLPs are considered straightforward and easy-to-implement networks capable of learning spatial [78] and temporal [304] relationships when the input data is appropriately structured. This makes them versatile tools not only in fields like classic machine learning tasks involving classification and regression but also in processing sequences and spatial patterns to a certain extent.

1.2.3 Convolutional Neural Networks

Convolutional neural netwokrs, or CNNs [305] are specialized for processing data with a grid-like topology, such as images. A CNN layer applies a set of learnable filters to the input. Each filter performs a convolution operation across the input, producing a feature map that captures spatial hierarchies in data:

$$\mathbf{y}_{ij} = \sum_m \sum_n \mathbf{W}_{mn} \mathbf{x}_{i+m,j+n} + \mathbf{b} \tag{.2}$$

where $\mathbf{W}$ is the filter matrix, $\mathbf{x}$ is the input matrix, and $\mathbf{b}$ is the bias. One of the main strengths of CNNs in audio synthesis is their ability to capture local dependencies and learn spatial hierarchies in data. This capability is crucial for understanding and generating audio content where the relevance of certain frequencies changes over time. CNNs are also inherently parallelizable, making them efficient for training on large datasets with extensive computational resources. CNNs excel in extracting hierarchical features from spectrograms, which is beneficial for tasks such as speech synthesis, music generation, and environmental sound synthesis. By applying multiple layers of convolutional filters, CNNs can capture intricate patterns in audio data [306], from basic textures and tones to complex soundscapes [88] and vocal inflections [307, 308, 309].

1.2.4 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a class of neural networks designed to handle sequential data, making them well-suited for tasks such as audio generation. RNNs process sequences by maintaining a hidden state that captures information from earlier inputs, allowing them to exhibit dynamic temporal behavior. Unlike feedforward neural networks, RNNs utilize their internal state to process sequences of inputs. The basic formula for an RNN is:

$$\mathbf{h}_t = f(\mathbf{W}[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}) \tag{.3}$$

where $\mathbf{h}_t$ is the hidden state at time t, $\mathbf{x}_t$ is the input at time t, f represents a non-linear activation function, and $\mathbf{W}$ and $\mathbf{b}$ are the network parameters. RNNs struggle with long-term

dependencies due to issues like vanishing or exploding gradients. To mitigate this, advanced RNN structures like Long Short-Term Memory (LSTM) [310] units and Gated Recurrent Units (GRU) [254] were developed. LSTM [310] units include mechanisms called gates that regulate the flow of information. These gates allow LSTMs to retain or discard information across long sequences, effectively addressing the problem of vanishing gradients by learning which data to keep for long-term memory and which to forget. Similarly, GRUs [254] simplify the gating mechanism of LSTMs, combining the forget and input gates into a single update gate, making them more efficient and faster to train while still capable of capturing dependencies over time. RNN-based models are particularly beneficial where the continuity and temporal structure of audio are crucial, which makes them an ideal candidate for modelling continuous audio waveforms. Therefore, RNNs have become popular neural network layers for many audio synthesis architectures [78, 14, 311]. However, RNNs typically require considerable computational resources and expertise in network tuning and training and usually struggle with tasks requiring long sequence memory and rapid real-time processing.

1.2.5 Transformers

Transformer model [312] was introduced by Vaswani et al. originally tasked for text processing and understanding. Unlike RNNs, which process data sequentially and therefore cannot be fully parallelized, transformers process entire sequences simultaneously [313]. This allows for much faster training times as computations can be efficiently distributed across multiple processing units. The core idea behind transformers is the self-attention mechanism, which allows the model to weigh the significance of different parts of the input data independently of their position in the sequence. This enables transformers to handle long-term dependencies by directly modelling the relationships between all parts of a sequence, regardless of distance.

Transformers have been successfully applied to audio generation tasks, where their ability to model long-range dependencies and generate contextually rich audio sequences shines. In the audio domain, transformers have been explored extensively in audio processing tasks, such as acoustic scene classification [314], speech synthesis [315, 316, 317], and music generation [318]. In addition to modelling one-dimensional raw audio waveforms by autoregressive

transformers, transformers can be adapted to model two-dimensional spectrograms, which could be converted back to waveforms via vocoders [319]. Additionally, vision transformer (ViT), initially developed for image processing tasks [320], has been effectively adapted to analyze Mel spectrograms in the audio domain. This adaptation demonstrates ViT's capability to discern complex patterns and relationships in audio data across time and frequency domains, offering insights that might elude traditional models.

1.3 Activation Functions

Activation functions are crucial in neural networks as they introduce non-linear properties to the model, which allows them to learn more complex patterns in the data. One of the most classic activation functions is Sigmoid [321]: $\sigma(x) = \frac{1}{1+e^{-x}}$, historically used widely because it maps input values into a (0,1) range, acting like a gate. However, it is less used today due to issues like vanishing gradients, where gradients become very small, effectively preventing the network from learning.

Tanh, or the hyperbolic tangent function, outputs values between -1 and 1: $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. It is similar to the sigmoid but provides better training performance for zero-centered data. In GAN-based audio generative models, the tanh function is commonly used in the last layer of GANs [141, 115, 100] because it ensures the output is smoothly scaled between -1 and 1, matching the amplitude range needed for audio signals. However, Tanh is also prone to the vanishing gradient problem but to a lesser extent than the sigmoid function.

Subsequently, Rectified Linear Unit [322] (ReLU$(x) = \max(0, x)$) has become the most widely used activation function in deep learning, especially for large networks. ReLU helps to alleviate the vanishing gradient problem and allows models to converge faster. However, it can suffer from "dying ReLU" problem where neurons effectively die during training and only output zero.

To address the dying neuron issue of the ReLU, the Leaky ReLU [323] was proposed: Leaky ReLU$(x) = \max(0.01x, x)$. This function has a small positive slope in the negative area, so it does allow for backpropagation of errors through the network even for negative

input values. Nowadays, Leaky ReLUs can be easily found in generative audio models [96, 110], especially paired with CNNs to learn complex data structure.

Lastly, Softmax [324, 325] is used primarily in the output layer of a classifier, where it maps logits, the outputs of the last linear layer of a multi-class classification neural network, to probabilities: $\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$ for each output i of the network, and the sum is overall possible classes j. Softmax is usually incorporated in conditional GAN-based architectures [100, 326], where the discriminator is tasked to assess whether the generated audio matches a given condition (e.g., a specific type of sound or audio quality). Softmax helps in evaluating the confidence level across different classes of audio outputs. This supports more nuanced and controlled audio generation, enhancing the discriminator's ability to guide the generator towards producing more realistic sounds.

1.3.1 Machine Learning Techniques

As introduced above, machine learning models use neural networks and activation functions to learn data structures by stacking up non-linear functions. The learning process is achieved by gradient descent, where the model's error, quantified by a loss function, is minimized across many iterations. The loss function measures the difference between the predicted output of the model and the actual output. Common loss functions include L1 or L2 loss for regression tasks and cross-entropy loss for classification tasks. During training, the model's parameters (weights and biases) are adjusted to minimize the loss. This adjustment is done through backpropagation [298], a method where the gradient of the loss function is calculated with respect to each parameter. The model then updates its parameters in the direction that reduces the loss, using an optimization algorithm such as stochastic gradient descent [327], Adam [257], or RMSprop [328]. These optimizers refine the learning rate and other aspects of the training process to make learning more efficient.

Additionally, weight initialization also plays a crucial role in the success of training deep neural networks. Poor initialization can lead to vanishing or exploding gradients, where the weights either shrink to zero or grow too large, impeding the learning process. Strategies like He [329] initialization or Xavier initialization [330] are designed to maintain the scale

of the gradients throughout the network, ensuring stable learning. Optimization techniques further enhance model training by adapting learning rates or using momentum to escape local minima and accelerate convergence to the global minimum of the loss function. Techniques like batch normalization and dropout are employed to improve model generalization and prevent overfitting, which is crucial when models are exposed to new, unseen data.

Through these processes, machine learning models iteratively learn optimal parameters that best map the input data to the desired output. This learning framework underpins many practical applications, from natural language processing and image recognition to complex decision-making tasks. Each component, from activation functions to optimization techniques, is integral to crafting models that can effectively learn from data and perform well on a variety of tasks.

2 Questionnaires

Participant Consent Form:

I agree to take part in this research study. In giving my consent, I confirm that that:

• The details of my involvement have been explained to me, and I have been provided with a written Participant Information Statement to keep.
• I understand the purpose of the study is to investigate neural audio synthesis for sound effects creation.
• I acknowledge that the risks and benefits of participating in this study have been explained to me to my satisfaction.
• I understand that in this study I will be required to rate the performance of the synthesis model in terms of quality and realness/naturalness.
• I understand that being in this study is completely voluntary.
• I understand that I am free to withdraw from this study at any time and that I can choose to withdraw any information I have already provided (unless the data has already been de-identified or published).
• I have been informed that the confidentiality of the information I provide will be protected and will only be used for purposes that I have agreed to. I understand that information identifying me will only be told to others with my permission, except as required by law.
• I understand that the results of this study may be published, and that publications will not contain my name or any identifiable information about me.

I agree to take part in this research study.

Yes

No

FIGURE .1. Participant content form.

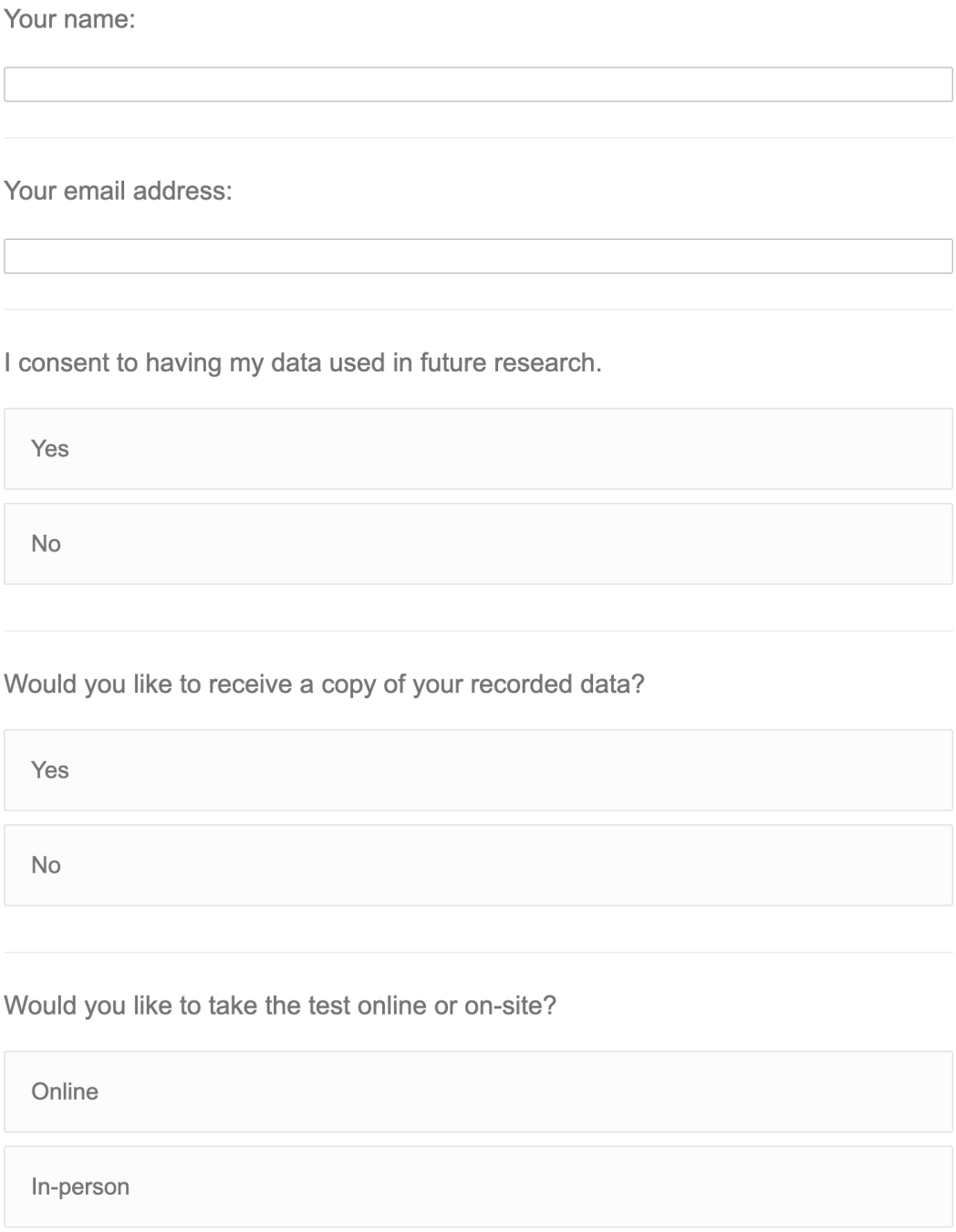


FIGURE .2. Demographic form for participants.

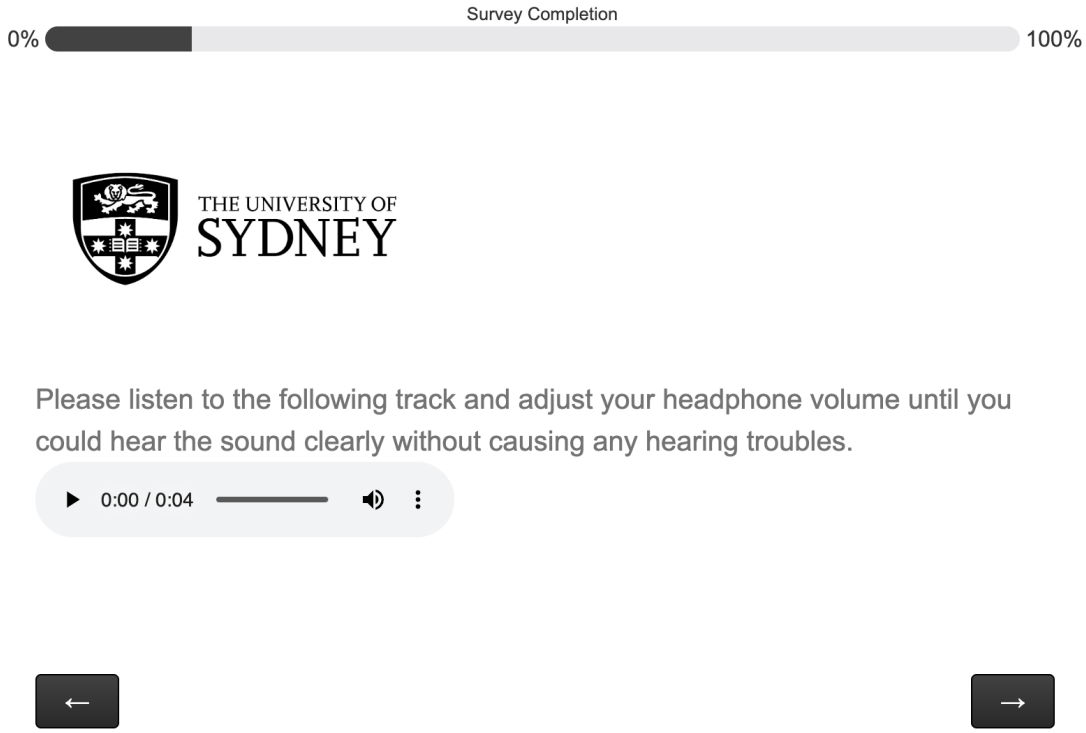


FIGURE .3. Listening trial for the listening test. Participants are encouraged to wear headphones during the test and adjust their volume so that it does not hurt the participant.

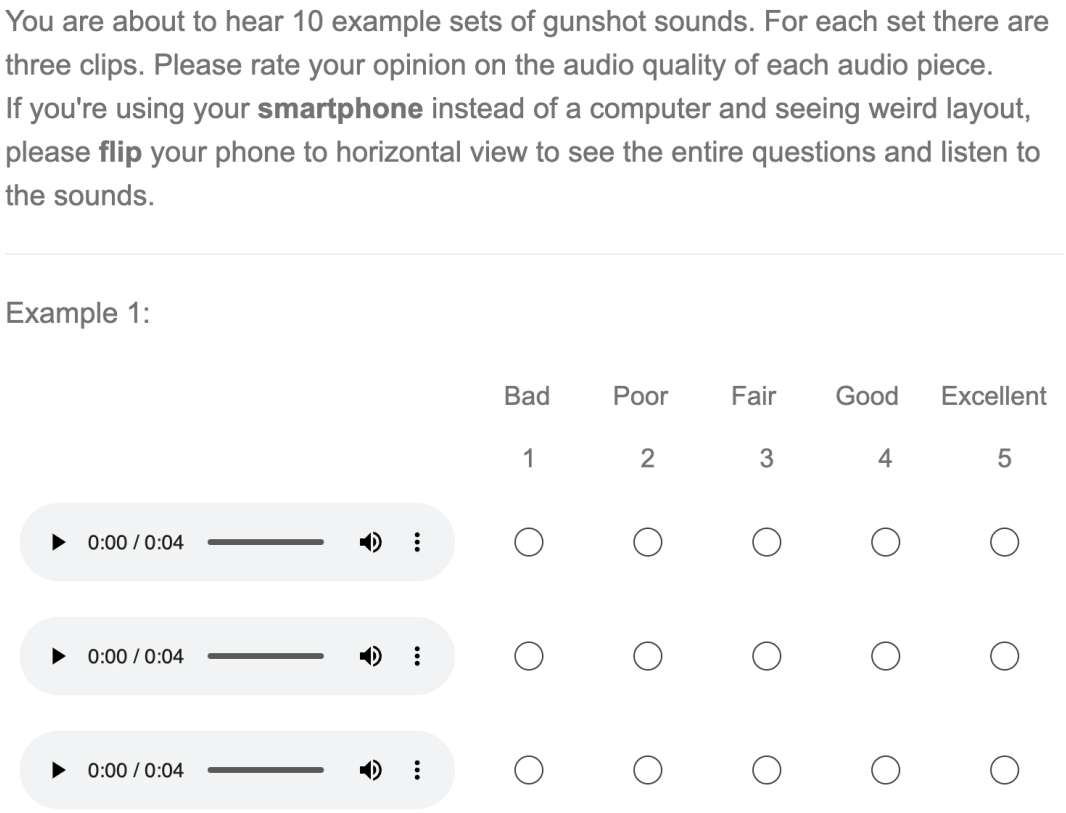


FIGURE .4. Listening test sample questions.

2.1 Listening test results

Methods	Categories																																Mean	Variance
Original	footstep	4	5	5	5	4	5	5	5	4	5	5	4	4	5	5	3	4	5	5	4	5	5	5	5	4	5	5	5	4	5	3	4	4.5625	0.37109
	birds	5	5	5	4	5	4	4	5	3	5	4	4	2	3	4	5	5	5	4	4	5	3	2	5	5	3	4	5	5	4	4	5	4.21875	0.79590
	gunshots	4	4	5	4	4	5	5	5	4	4	5	4	4	5	3	5	4	4	3	4	4	5	5	5	4	4	5	4	4	5	3	4	4.28125	0.38965
	rain	4	4	5	4	5	4	5	4	4	4	2	4	5	4	2	4	4	4	5	4	5	4	5	4	4	4	5	4	5	4	5	4	4.1875	0.52734
	engine	5	5	5	5	3	3	4	4	5	5	4	5	4	5	4	5	5	5	5	3	5	5	4	5	5	5	5	5	5	5	4	5	4.59375	0.42872
Baseline	footstep	2	3	4	4	4	4	4	5	4	5	5	3	2	3	4	4	5	3	5	3	4	3	5	3	5	4	5	4	3	4	5	5	3.9375	0.80859
	birds	3	2	2	3	4	3	2	1	2	3	4	3	4	3	3	5	4	4	4	4	5	4	3	4	4	4	5	2	4	5	4	4	3.46875	0.99902
	gunshots	2	5	4	5	4	3	4	4	3	5	4	5	4	5	4	5	3	3	5	3	3	4	5	5	4	3	2	3	5	3	4	3	3.875	0.85938
	rain	3	3	3	2	4	4	3	5	3	4	3	4	5	3	5	3	4	4	3	3	2	5	3	5	3	4	4	3	1	2	3	4	3.4375	0.93360
	engine	5	4	3	4	3	5	4	5	5	3	4	4	2	4	5	4	5	4	4	3	5	3	4	5	4	5	4	4	5	5	4	5	4.15625	0.63186
Our method	footstep	5	5	5	2	3	5	4	5	5	4	5	5	3	4	4	2	4	3	4	5	5	4	4	5	5	3	4	4	5	5	5	4	4.21875	0.79590
	birds	4	5	3	2	5	3	5	3	4	4	3	4	4	5	3	5	3	4	4	5	4	4	3	4	3	4	3	5	4	5	4	4	3.90625	0.64746
	gunshots	5	4	5	4	3	5	5	4	4	5	5	5	4	3	5	4	4	5	2	3	4	3	4	4	5	5	5	4	4	5	2	4	4.15625	0.75683
	rain	4	4	4	3	4	3	4	5	5	3	4	4	4	5	5	4	4	4	4	3	5	4	4	3	5	2	5	4	4	5	4	4	4.03125	0.53027
	engine	5	3	5	5	5	3	4	5	5	5	5	4	4	4	5	5	4	5	5	5	3	4	5	5	3	4	4	5	5	5	4	5	4.46875	0.49902
Wavegan	footstep	4	2	3	3	4	3	5	3	4	3	5	2	3	5	5	3	4	3	3	2	5	3	4	4	4	5	3	5	4	4	3	4	3.65625	0.85059
	birds	2	1	2	4	4	3	3	4	1	3	4	2	4	2	4	1	2	3	1	2	4	2	3	3	3	4	2	4	4	3	2	2	2.75	1.0625
	gunshots	3	4	5	4	4	3	4	2	4	4	3	4	4	4	1	2	4	4	3	4	4	5	3	5	4	1	4	4	2	5	3	4	3.5625	1.05859
	rain	2	1	1	2	3	1	2	2	3	4	3	4	3	4	1	4	1	2	5	3	2	4	3	1	4	3	5	4	1	2	3	4	2.71875	1.51464
	engine	3	1	2	4	4	1	5	2	1	3	2	3	2	1	4	3	5	1	2	3	5	5	4	4	3	4	3	1	3	2	4	3	2.90625	1.64746

TABLE .1. Detailed MOS evaluation results

3 Figures

3.1 Chapter 4

3.1.1 Baseline Model Architecture

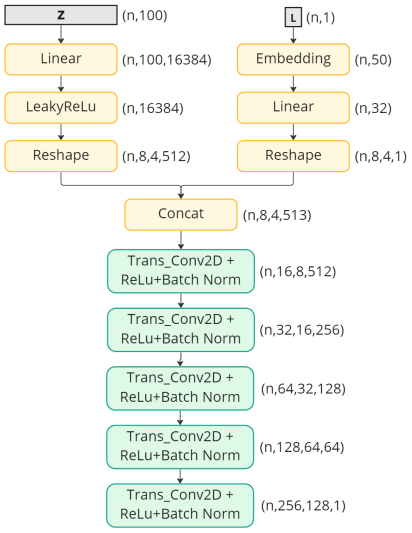


FIGURE .5. Generator Architecture

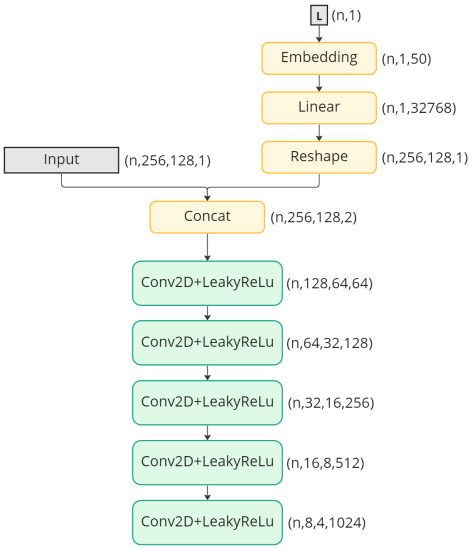
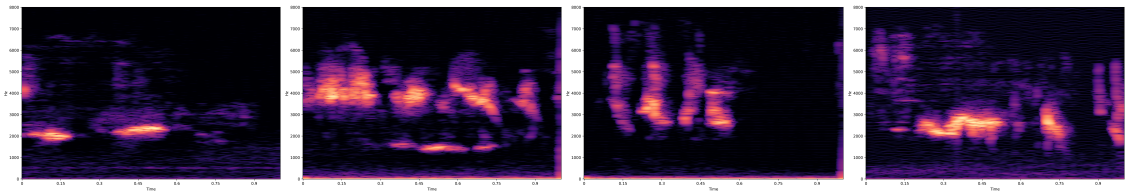


FIGURE .6. Discriminator Architecture

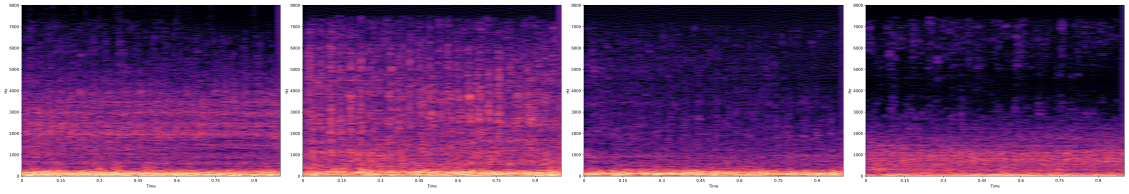
3.1.2 Location of Listening Tests

FIGURE .7. A photo of the place taken for the listening test. We took our subjective listening tests in the Carlab (Computed Audio Research Lab) at Electrical and Information Engineering at the University of Sydney. The participants were asked to listen to sit on a chair and put on a headphone (AKG K52) before starting the test. They were encouraged to adjust the volume to ensure the sound levels were appropriate to their hearing. The computer played each sound at a time, and the participants were asked to rate their preference for the sound.

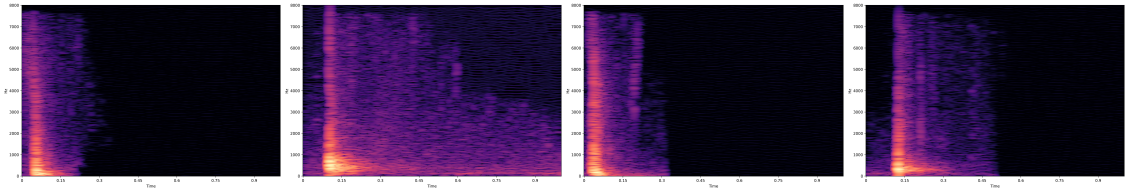
3.1.3 Randomly Selected Sound Examples



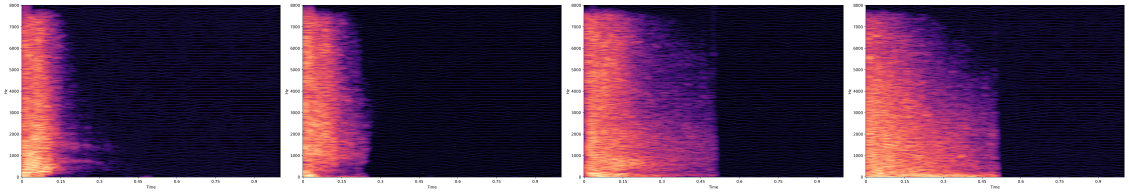
(A) Birds 1 (B) Birds 2 (C) Birds 3 (D) Birds 4



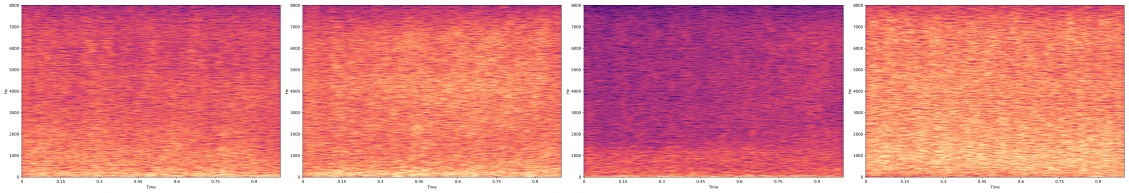
(E) Engine 1 (F) Engine 2 (G) Engine 3 (H) Engine 4



(I) Footstep 1 (J) Footstep 2 (K) Footstep 3 (L) Footstep 4



(M) Guns 1 (N) Guns 2 (O) Guns 3 (P) Guns 4



(Q) Rain 1 (R) Rain 2 (S) Rain 3 (T) Rain 4

FIGURE .8. Randomly selected sound examples generated from the proposed method using WGAN. We can see the generator successfully learns patterns of sounds from different categories. Stochastic patterns such as raining could usually be easily modelled with generative models, as they typically start with vectors sampled from Gaussian noise. Impulsive signals like footsteps and guns contain much less noise, which could be seen from the silent backgrounds. From the engines we are able to seen the evenly distributed spectral curves. For bird category, the sounds are very clear and the pitch up or down could easily be heard.

3.2 Chapter 5

3.2.1 Sound Examples

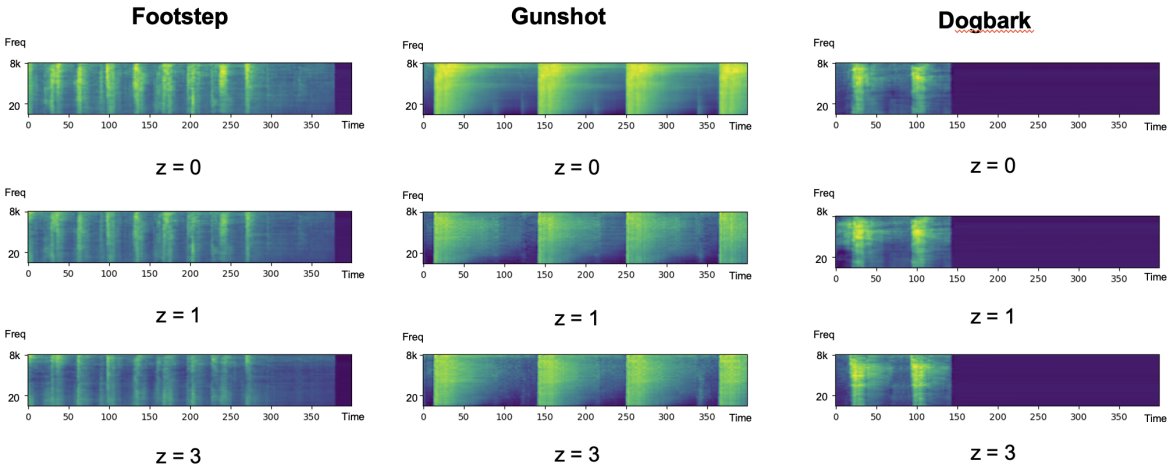
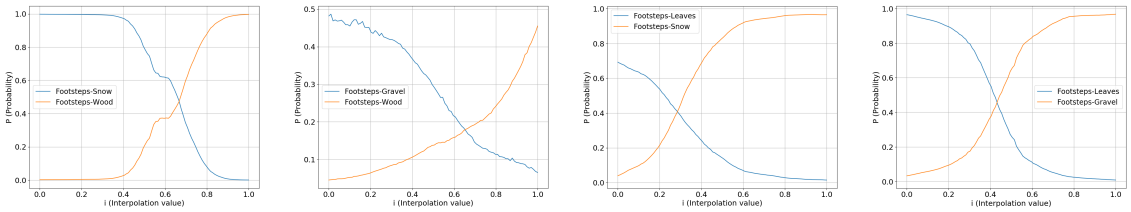


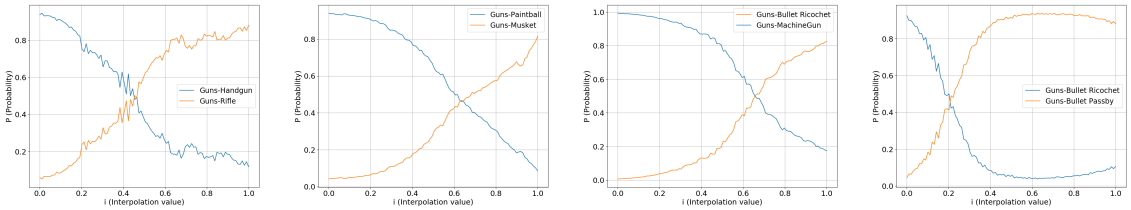
FIGURE .9. Sound examples generated from our proposed model architecture. The horizontal time axis are in frames. Each frame represents 160 samples. We use a reference sound to guide the sound generation. Then we manually set latent variable to 0 for all channels to generate the sounds. From the graph we are able to see the drastic change in spectral distributions after we changed z.

3.3 Chapter 7

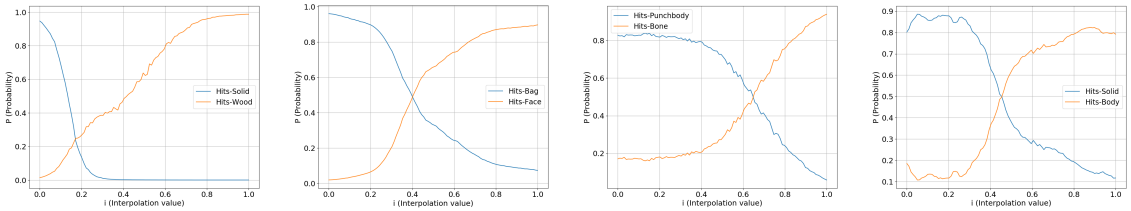
3.3.1 Interpolation along the conditioning space



(A) *Interpolation for the footsteps category*



(B) *Interpolation for the gunshots category*



(C) *Interpolation for the hits category*

FIGURE .10. *Effect of interpolating points in the conditioning space. We randomly selected five combinations of interpolation between two classes in each sound category. The horizontal axis represents the interpolation points between two targets, with one increasing from 0 to 1 and the other from 1 to 0. The vertical axis represents the probability for that class returned by a trained classifier. As the conditioning label as input to the generator is randomly sampled from the Gaussian distribution parameterized by the interpolated mean values, we sampled one hundred times to obtain the probability curves. This process smooths the curve and provides a more accurate visualization of the trend between the conditioning label and the returned probability.*

3.4 Chapter 8

3.4.1 Regression test results

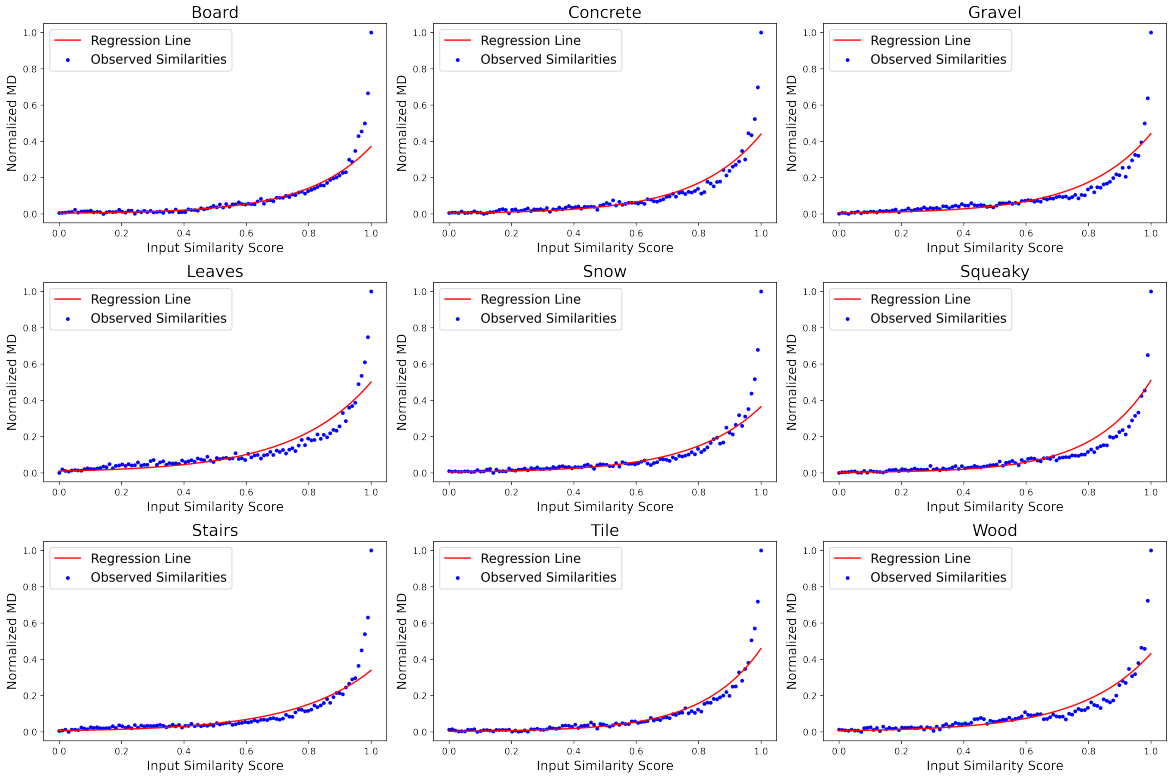


FIGURE .11. Regression Analysis for our model trained on Footstep-set.

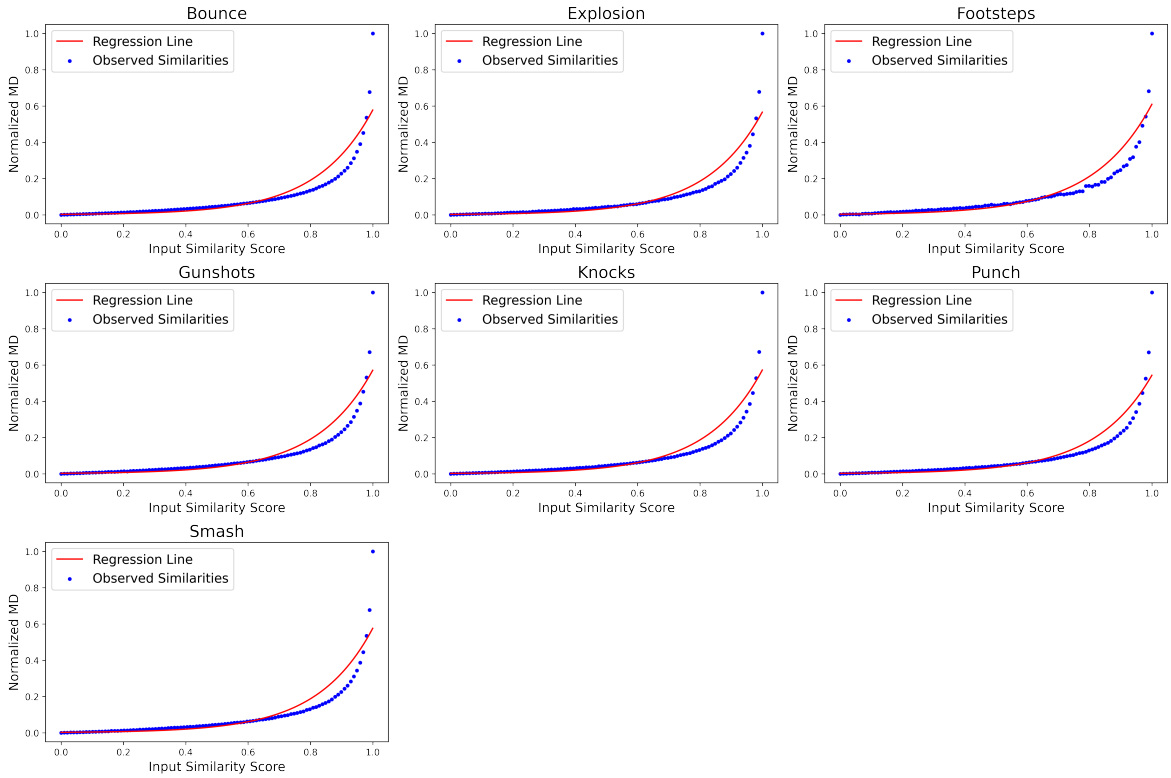


FIGURE .12. Regression Analysis for our model trained on Impact-set.

3.4.2 Pseudo codes for sound generation

```python
def process_directory(input_dir, output_dir, model, config, min_value, max_value):
    # Ensure the output directory structure mirrors the input
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    # Walk through the input directory
    for root, dirs, files in os.walk(input_dir):
        for filename in files:
            if filename.lower().endswith(('.wav', '.mp3', '.flac', '.aiff')):
                file_path = path.join(root, filename)
                output_subdir = path.join(output_dir, path.relpath(root, input_dir))
                if not os.path.exists(output_subdir):
                    os.makedirs(output_subdir)

                # Process each file
                synthesized_audio = synthesize_audio(file_path, model, config, min_value, max_value)
                output_file_path = path.join(output_subdir, filename)
                sf.write(output_file_path, synthesized_audio, config["preprocess"]["sampling_rate"])
                print(f"Saved synthesized audio to {output_file_path}")

def synthesize_audio(audio_path, model, config, min_value, max_value):
    _, spec_centroid, loudness, mh_distances, _ = preprocess(
        audio_path,
        sampling_rate=config["preprocess"]["sampling_rate"],
        block_size=config["preprocess"]["block_size"],
        signal_length=config["preprocess"]["signal_length"],
        oneshot='True')

    spec_centroid = torch.tensor(spec_centroid, dtype=torch.float32).unsqueeze(-1)
    loudness = torch.tensor(loudness, dtype=torch.float32).unsqueeze(-1)
    mh_distances = torch.tensor(mh_distances, dtype=torch.float32).unsqueeze(0)

    normalized_mh_distances = (mh_distances - min_value) / (max_value - min_value)
    loudness_normalized = (loudness - config["data"]["mean_loudness"]) / config["data"]["std_loudness"]

    signal, noise, transient = model(spec_centroid, loudness_normalized, normalized_mh_distances)
    waveform = signal.detach().numpy().squeeze()
    return waveform

# Load model and configuration
model_path = "/home/rein/Documents/ICDDSP/runs/impacts-44100_13-11-2024_23"
with open(path.join(model_path, "config.yaml"), "r") as file:
    config = yaml.safe_load(file)

ddsp = DDSP(**config["model"])
state = ddsp.state_dict()
pretrained = torch.load(path.join(model_path, "state.pth"), map_location="cpu")
state.update(pretrained)
ddsp.load_state_dict(state)
ddsp.eval()

min_value = np.load('preprocessed/min_value.npy')
max_value = np.load('preprocessed/max_value.npy')

# Define input and output directories
input_directory = '/home/rein/Downloads/Sound_datasets/Impacts/Impact-set/test'
output_directory = '/home/rein/Documents/ICDDSP/ICDDSP-synth-Impact'

# Process all subdirectories
process_directory(input_directory, output_directory, ddsp, config, min_value, max_value)
```

FIGURE .13. Codes for generating sounds with the proposed model based on extracted acoustic features as well as similarity scores obtained from reference audio waveforms. Notice that the similarity scores were already extracted using pre-trained CLAP embedding models and saved as control vectors within the range [0, 1]. Since we've normalized similarity scores (MH distances) for all training sounds, during the inference stage, we still need to normalize the MH distances of any reference sound using the saved min and max values to keep them in a similar range. The generative model is trained, and during the inference mode, its weights are frozen as static.

3.4.3 Pseudo codes for evaluating timbre control

```
def generate_sound(guiding_vector):
    guiding_vector = torch.unsqueeze(guiding_vector, 0).to(device)
    sound, _, _= ddsp(spec_centroid, loudness_normalized, guiding_vector)
    return sound

sorted_paths = sorted(os.listdir(embedding_paths))
num_classes = len(sorted_paths)

def interpolate_guiding_vector(channel_index):
    guiding_vectors = torch.ones((100, num_classes))
    guiding_vectors[:, channel_index] = torch.linspace(1, 0, 100)
    return guiding_vectors

# Function to load mean and covariance for a given class
def load_mean_cov(class_index):
    folder_path = os.path.join(embedding_paths, sorted_paths[class_index])
    mean_cov = np.load(folder_path)
    mean = mean_cov[0, :]
    cov = mean_cov[1:, :]
    return mean, cov

all_similarities = []

# Loop through each class and compute similarity
for i in range(num_classes):
    guiding_vectors = interpolate_guiding_vector(i)
    mean, cov = load_mean_cov(i)
    similarities = []

    for vector in guiding_vectors:
        sound = generate_sound(vector)
        sound = sound.squeeze(-1)  # Ensure sound is properly shaped
        embedding = frechet.get_embeddings(x=sound.detach().cpu().numpy(), sr=config["preprocess"]["sampling_rate"]).squeeze(0)
        embedding = np.array(embedding)
        dist = mahalanobis_distance(embedding, mean, cov)
        similarities.append(dist)

    similarities = np.array(similarities)
    all_similarities.append(similarities)  # Append class similarities to the list

all_similarities = np.array(all_similarities)  # Convert list of arrays into a 2D numpy array
np.save('similarities_unnormalized_impact.npy', all_similarities)
```

FIGURE .14. Pseudo codes for controlling sound generation with similarity scores. As shown, the function 'interpolate guiding vector' outputs 100 guiding vectors interpolated between 0 and 1. Such interpolated vectors are used as similarity scores which are passed into the trained DDSP model. By varying the values between 1 and 0, we are able to manipulate and generate variations of sounds. The returned results are the similarities of a given audio class. In Figure .11 and Figure .12, we show the regression analysis based on this evaluation metric.