ABSTRACT

A NEURAL ANALYSIS-SYNTHESIS APPROACH TO LEARNING PROCEDURAL AUDIO MODELS

by
Danzel Serrano

The effective sound design of environmental sounds is crucial to demonstrating an immersive experience. Classical Procedural Audio (PA) models have been developed to give the sound designer a fast way to synthesize a specific class of environmental sounds in a physically accurate and computationally efficient manner. These models are controllable due to the choice of parameters from analyzing a class of sound. However, the resulting synthesis lacks the fidelity for the preferred immersive experience; thus, the sound designer would rather search through an extensive database for real recordings of a target sound class. This thesis proposes the Procedural audio Variational autoEncoder (ProVE), a general framework for developing a high-fidelity PA model through data-driven neural audio synthesis methods to address the lack of realism in classical PA models. The two-step procedure of training ProVE models is explained through examples of sound classes of footstep sounds and the sound of pouring water.

Furthermore, the thesis demonstrates a web application where users can generate footstep sounds by defining control variables for a pretrained ProVE model to show its capacity for interactive use in sound design workflows. The increase in fidelity from ProVE models is explored through objective evaluations of audio and subjective evaluations against classical PA methods. These results show that these learned neural PA models are feasible for sound design projects. The thesis concludes with a discussion of applications and future research directions.

A NEURAL ANALYSIS-SYNTHESIS APPROACH TO LEARNING PROCEDURAL AUDIO MODELS

by
Danzel Serrano

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Data Science

Department of Computer Science

December 2022

APPROVAL PAGE

A NEURAL ANALYSIS-SYNTHESIS APPROACH TO LEARNING PROCEDURAL AUDIO MODELS

Danzel Serrano

Dr. James Geller, Dissertation Advisor   Date
Professor of Computer Science and Data Science, NJIT

Dr. Mark Cartwright, Committee Member   Date
Assistant Professor of Informatics, NJIT

Dr. Przemyslaw Musialski, Committee Member   Date
Associate Professor of Computer Science, NJIT

BIOGRAPHICAL SKETCH

Author:	Danzel Serrano
Degree:	Master of Science
Date:	December 2022
Date of Birth:	
Place of Birth:	

Undergraduate and Graduate Education:

- Master of Science in Data Science,
  New Jersey Institute of Technology, Newark, NJ, 2022
- Bachelor of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2021

Major:	Computer Science

To my loving parents, who have constantly supported me, prayed for me, and never gave up on me. I wouldn't be where I am without their guidance in technology and in music.

To my little brother, who, despite having cerebral palsy, never fails to make everyone smile with his own smile. You are a big influence on my decision to pursue a career in academia and research.

To my church family at Word Christian Fellowship International (WCFI). Thank you for your constant prayers and encouragement.

Most importantly, to God, the one I fully trust my research, my work, and my whole life to because of His divine providence and amazing love.

ACKNOWLEDGMENT

I would like to thank Dr. Mark Cartwright for guiding me through this project, introducing the audio processing realm to me, and teaching me so much in the domain of machine listening and DSP.

Without Dr. Geller, I wouldn't be able to pursue this research topic, so I would like to thank him for allowing me this opportunity.

I'd also like to thank Dr. Przemyslaw Musialski, as my participation in his lab has also influenced the methodology described in this thesis.

Finally, I'd like to thank the members of the Sound Interaction and Computing Lab (SInC) at NJIT for being a motivating medium for brainstorming ideas and recommending papers in the fields of Machine Listening, Representation Learning, and HCI.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In immersive mediums, such as film, video games, and virtual environments, the sound designer's job is to record, synthesize, and process audio stimuli in a way that best accompanies a set of visual or other modes of stimuli. The designer is usually given a set of expectations or deliverables influencing this audio-visual relationship, such as genre, and the overall atmosphere of a scene. The sound designer's job is defined in a similar way in engineering fields: the processing of sound signals in such a way that it helps solve an engineering problem, like reducing noise in industrial machines [27].

For many sounds, especially realistic sounds synchronized to on-screen events, sound designers must search recorded sound databases for particular classes of sounds and edit and process them to meet their project's expectations. Traditional sound design has been done this way for many years: crafting an auditory scene comprises recording sets of sound classes and then putting them together in context [35]. However, the problem is not just finding a recorded sound that matches. Typically the designer would find a sound that is close to what they want through public databases of recorded sounds. They would then edit and process this initial sound until it matches the requirements of the situation. In interactive media, like video games and virtual environments, this is much harder to do due to the characteristics of a sound event being dependent on the player's current context (movement speed, distance to sound sources, etc). Another problem is that these datasets of recorded audio for a sound class can be massive, and sound designers must scan through them to collect enough sounds for a specific sound class they want to model for their project. The time it takes to scan through a database depends on the level of variability required for a sound class and the number of sound classes the designer needs. Most projects require more than one sound class, so this task is usually lengthy. If the sounds collected by the designer are not able to match its corresponding visual event because a suitable sound could not be found from the database or it could not be processed appropriately, the resulting auditory scene may lose its realism, especially in interactive media where a consumer is expected to be exposed to the auditory scene for an extended period. The second problem of storage limitation is also dependent on the sound class. The designer must collect many distinct class recordings if a sound class requires high variability, which takes up storage space, especially in a high-quality project.

Procedural Audio (PA) is a different approach to capturing a specific sound class that aims to solve these problems. As an analogy, traditional sound design is to a product development lifecycle, as PA is to mathematical modeling: the former creates a static product; the latter models a variable process [17]. In other words, PA models an approximation to the sound class. The constraint on storage is mitigated, because only the process is kept in memory and used when needed, which is great for interactive media, where the data storage of visual stimuli should be the focus. Since PA models the sound class itself, the property of sound class variation is embedded in an ideal model, which is highly controllable and expressive. This avoids the need to search extensive databases, reducing the time (and money) it takes for a sound designer to capture a sound class. Current approaches to PA are computationally efficient and have control inputs and synthesis outputs that are physically inspired.

Although efficient and powerful, PA is still not widely used for sound design and game audio for several reasons. One of them being the lack of reliable tools in the immersive media industry [4]. However, the main reason PA is not being used is simply that the sound quality of the resulting synthesized sound is lower than real samples of the target sound class. This is partly due to the lack of research in bringing PA to be a standard in immersive media, where the primary research focus is on improving upon presenting visual stimuli. Current state-of-the-art approaches to enhancing the quality of sound synthesis have been data-driven through deep learning. Neural audio synthesis has been extensively researched for applications like speech synthesis [52, 48, 44] and music generation [38, 13], but neural sound synthesis for specific environmental sounds and foley[1] synthesis have only recently been illuminated [10]. Furthermore, the research on controllable neural synthesis models has progressed mainly in the music [53] and speech [9] domains, while research into controllable neural audio synthesis models for environmental sounds have been limited.

This thesis concentrates on the problem of developing a general data-driven framework for learning controllable PA models for environmental sounds with improved sound quality through neural audio synthesis. The framework presented, the Procedural (audio) Variational autoEncoder (ProVE), can be seen as a 2-stage (analysis-synthesis) abstraction of the structured approach to sound design presented in [19]. The primary neural audio synthesis model, however, is an autoencoder model from a recent paper called Differentiable Digital Signal Processing (DDSP) [16], which proposed a differentiable version of the harmonic + noise sound synthesis method [47]. Emphasis must be placed on the fact that, despite most PA models being efficient enough to be run in real-time, this thesis does not contain results of PA models developed through ProVE running in real-time. This is because the neural audio synthesis model used for the examples illustrated in this thesis does not have a fast enough inference time. One of the main objectives of this thesis is to demonstrate that ProVE models surpass classical PA models in fidelity, not in feasibility for use in real-time interactive media. Nevertheless, ProVE is agnostic to neural audio synthesis approaches, so the benefit of utilizing recent advances in

[1]Foley, in this case, refers to the act of physically recording additional sound effects to be added post-production of immersive medias.

real-time neural audio synthesis [5] as a next step is discussed in the future works section.

1.1 Problem Statement

Ultimately, the problem with PA models is low fidelity. Although these models are controllable with physics-inspired parameter choices and synthesis methods, the lack of use in immersive media is due to low audio quality. Subsequently, sound designers would rather search an extensive database to find recorded audio that best matches the sound class they need [17]. The ProVE framework aims to develop a controllable PA model that synthesizes higher fidelity environmental sounds than traditional PA models through the use of neural audio analysis-synthesis. The definition of fidelity in this case is the measure of how well a synthesis method produces sounds of a sound class and must not be mistaken for how many discrete samples are used for 1 second of audio, or the sample rate.

The problem formulation in terms of generative modeling is as follows:

An environmental sound class $\boldsymbol{X}_{class}$ can be interpreted as being a conditional distribution $p(x|\gamma, u)$ with $x \in \boldsymbol{X}_{class}$ being a sound belonging to class $\boldsymbol{X}_{class}$, $\gamma \in \boldsymbol{\Gamma}_{class}$ being a control variable sequence, and $u \in \mathcal{U}([-1, 1])$ being a random uniform sequence. The intuition here is that the definition of γ influences the characteristics of sound x, and sampling u captures the variation of possible sounds conditioned by γ. With these assumptions, the goals of the ProVE framework are:

1. With an autoencoder ($f_\theta(x)$,$g_\theta(z)$), learn an encoded representation of audio x, $z = f_\theta(x)$. Here, the encoder f_θ and the decoder g_θ are neural networks and θ denotes their parameters that get updated through gradient-based optimization. One can interpret z as a "machine-interpretable" control variable, because although modifying z results in controlling the sound from the decoder, g_θ, a human would not understand the learned semantics of z.

2. Learn a mapping from "human-interpretable" control to "machine-interpretable" control. That is, optimize a mapping from $\mathbf{\Gamma}_{class} \mapsto \boldsymbol{Z}_{class}$, where $\boldsymbol{Z}_{class}$ is the latent space learned from training the autoencoder. To do so, γ is assumed to correlate with z via some random sample $u \in \mathcal{U}([-1,1])$. The control mapping is then $h_\phi : \mathbf{\Gamma}_{class} \times \mathcal{U}([-1,1]) \mapsto \boldsymbol{Z}_{class}$, where ϕ are to denote the parameters for neural network h_ϕ.

3. The controllable data-driven procedural audio model is the composition of the "human-interpretable" control mapping $\hat{z} = h_\phi(\gamma, u)$ and the audio decoder $\hat{x} = g_\theta(\hat{z})$:
$$\hat{x} = (g_\theta \circ h_\phi)(\gamma, u) \sim p(x|\gamma, u)$$

It is important to keep the generative modeling formulation in mind. Most modern generative models sample a latent distribution in order to sample an approximation of the real distribution. Many classical approaches to developing procedural models for environmental sounds can be defined in the same way: the input to a synthesis module is usually a sequence of uniformly distributed white noise ($u \in \mathcal{U}([-1,1])$) and an accompanying control variable (γ). The control variable governs what the synthesis module does with the uniform sequence to output sound. In other words, one can sample the space of uniformly distributed noise to sample the space of sounds that approximates a sound class. This formulation of developing PA models as data-driven generative models greatly influences the use of neural audio synthesis techniques in ProVE.

It is also important to note that the set of models learned in ProVE reported in this thesis is targeted towards the class of various environmental sounds. Although research towards controllable synthesis of speech and music has leaned towards a language-oriented approach, it still shares similarities to the analysis-synthesis approach in ProVE.

1.2 Thesis Structure

The remainder of the thesis seeks to answer the following questions:

1. Do PA models trained in ProVE obtain a higher fidelity than traditional PA models of the same sound class?

2. Since the learned latent sequence space of an audio autoencoder $f_\theta : \boldsymbol{X}_{class} \mapsto \boldsymbol{Z}_{class}$ is not understood by a human, how can we define a "human-interpretable" control sequence $\gamma \in \boldsymbol{\Gamma}_{class}$ from $\boldsymbol{X}_{class}$ that allows a human to control the sound synthesis of data-driven neural audio synthesis models?

3. Are ProVE models feasible for use in the current sound design workflow?

The following chapter introduces the background of the classical analysis-synthesis approach to designing a procedural model and current approaches to neural audio synthesis and neural audio analysis. The *approach* section explains the data collection and processing methods for an arbitrary target sound class and the 2-stage "connectionist" analysis-synthesis approach in ProVE. In the evaluation section, subjective and objective evaluations are done on the models learned in ProVE, followed by a discussion of their results. Finally, the thesis concludes with applications and limitations of ProVE, the implications of ProVE in the workflow of sound design for immersive media, and future research directions in neural PA models.

CHAPTER 2

BACKGROUND

Andy Farnell's *Designing Sound* [19] takes a systematic approach to introducing the theory and technique for developing classical procedural audio models. The theory ranging from the basics of understanding the physics of the propagation of sound waves to psychoacoustics, the study of sound perception. Farnell also provides the reader with a plethora of example PA models that synthesize a variety of sound classes. In the book, a multi-stage structured approach to the development of procedural audio models is given.

With regard to sound design, an auditory scene may be described, but the designer still must understand what sound classes are required to make up the full context. This step of planning out a full auditory context is called Requirements Analysis. There are then intermediate steps to take in iteratively improving upon that product with the responses from the penultimate testing step, the first being the Research step. Once the required components of an auditory scene are planned out, the designer must delve deeper into what makes up the components. As an example, if a designer wishes to create a PA model for a rainforest soundscape, they would have in their requirements the rain, as well as a few other environmental sounds. The designer must then research to understand the physics behind droplets of rain on several types of surfaces, as well as similar analyses on the other required sound classes.

The next steps are Model Making and Method Selection. With the information retrieved from the research step, the designer sketches out an abstract version of the model in an object-oriented way, listing out technical requirements like object structures, audio streams, and possible synthesis methods to use. It is here where

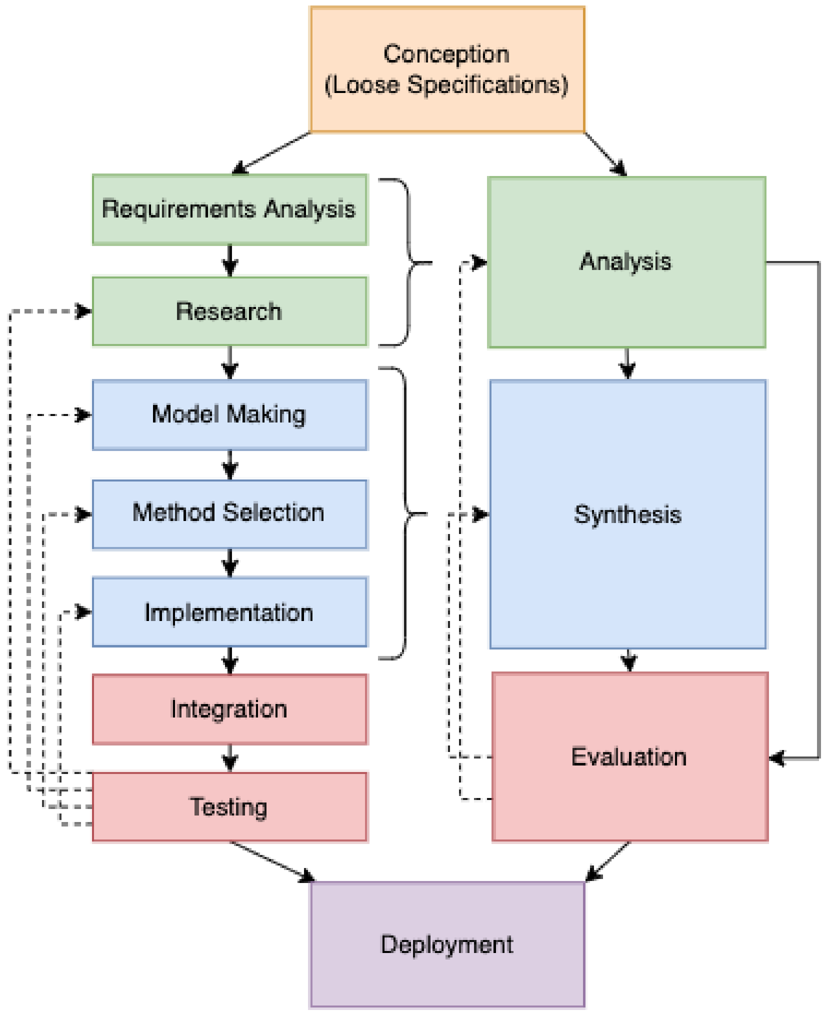


Figure 2.1 (Left) The full structured approach cycle to developing PA models as explained in Andy Farnell's *Designing Sound* [19]; (Right) an abstraction of this process into analysis and synthesis, where the output of the analysis stages are also evaluated and improved in an iterative manner.

most of the control variables are planned out from the data in the research and analysis steps. Along with creating a blueprint for the model, the designer must also select a synthesis algorithm to use that best suits the analysis of the specific sound classes.

The next step is Implementation, or the actual building of the model itself with a framework for audio synthesis. Lastly are the Integration and Testing stages, where the synthesis samples from the model are placed into a project or the implementation of the model is integrated into a system to be evaluated against the initial stated deliverables in the analysis stages. When a feature is not up to par with the expectations of the initial vision or with the information retrieved from the research stage, then the sound designer takes an iterative step backward to the necessary step to fill in the gaps. Usually, this means modifying or adding to the current configuration in the implementation step, however, the sound designer may have to go back to the research step to get more information about what's missing.

2.1 Analysis-Synthesis Abstraction to Designing Procedural Audio Models

The full "waterfall-esque" iterative process of the aforementioned structured approach to developing PA models can be abstracted into two stages: Analysis and Synthesis. One can list requirements for an auditory scene and research those requirements with various digital signal processing (DSP) techniques for analysis and simple attentive listening. As a starting point, studying the waveform in the time domain can tell a sound designer a few things, like the overall envelope of a sound event or the general activity of an entire auditory scene. In the context of impulse-based sound classes, the envelope of a sound can represent the amount of physical force exerted against a surface. If the designer is examining sounds with prevalent harmonics (musical instruments, speech, etc.), they can even examine the shape of periodic features in the time domain; however, that's usually done through spectral analysis in the frequency domain. Analysis of sound in the time domain and frequency domain may lead questions about what happens in the physical world to make the waveform and spectra appear or evolve in a certain way. In that case the designer may also want to do a physical analysis of a sound class, which are what makes the control parameters of PA models of environmental sounds physically inspired.

From the research and analysis done for the requirements of the target sound class, the designer lists possible control parameters for the synthesis module. For example, possible control parameters for a model of synthesizing footsteps would be the selection of ground material and the ground reaction force of a human foot during a step phase. The size of this list is usually reduced by correlating one parameter to others in a meaningful way to users of the model. With this list of control parameters, the designer has to choose synthesis methods that will benefit best in matching the information from the analysis step. Examples of synthesis methods used for PA models of environmental sounds include pure additive synthesis [39, 46], wavetable synthesis [26], subtractive noise, granular synthesis [45], piecewise modeling, and physical modeling.

The difference between the proposed approach in this thesis and the set of models using physical modeling synthesis [49] must be emphasized. These models render sound through simulation methods in computational physics, usually through formulating state space models. Although with exciting high-fidelity results, the physical modeling synthesis approach has disadvantages. One of them is that some of these models are computationally expensive in that prerendering sound consists of solving a system of differential equations using numerical methods. It is still an open problem to design a solution for the real-time rendering of computer-generated and physically-modeled sound, especially in interactive media like video games and virtual environments [28]. Another disadvantage of formulating sound synthesis models with the physics-based approach is that one would require immense knowledge in the domain of physics or mechanics. The most prominent are fluid dynamics [34, 7], rigid-body physics [33, 8], and research in acoustics simulations like wave propagation [3] and sound spatialization. Unfortunately, sound designers without this domain knowledge must wait for the research towards a specific sound class they require. This thesis is a step towards developing sound synthesis models that are data-driven while still borrowing some domain knowledge from physics, as opposed to being entirely driven by a model of the sound class in the physical world.

2.2 Neural Audio Synthesis

The deep learning subset of data-driven generative modeling methods for generative audio are coined neural audio synthesis methods. The main neural audio synthesis method used in this thesis is from Engel et al. [16], which presented Differentiable Digital Signal Processing (DDSP); a library of differentiable DSP modules formulated in a way that allows them to be used in automatic differentiation frameworks (Tensorflow, PyTorch, etc). The goal of DDSP was to move towards the vocoder approach to generating audio, which generates audio with oscillators and filters using the analysis-synthesis paradigm. This is opposed to the models that predicts Fourier coefficients in the spectral domain [52, 15], as well as autoregressive approaches that predict samples one-by-one in the time domain [29, 43], which take longer to train and have a larger dataset size.

Adversarial approaches like HiFi-GAN[32] and WaveGAN [14] have also been used for generating high quality synthesis of speech and music. Recent papers have used HiFi-GAN for the neural audio synthesis of footstep sound effects [11], and drum sounds [41]. Although the resulting sample space successfully generates high-fidelity footstep sounds, the model is only controllable up to a discrete label for material selection. Also, the audio generated are one-shots of a single footstep, not taking into account the speed or amount of force on the floor. Furthermore, adversarial approaches are typically unstable to train, and DDSP lists adversarial losses as another goal avoidance.

The experiments done in the DDSP paper were focused on the neural audio synthesis of sound classes that are rich in harmonics, like musical instruments and speech, using an audio autoencoder model. The encoder side of the model used a pre-trained CREPE model [31], a state-of-the-art pitch estimation model, to encode fundamental frequency $f_0[n]$ sequence. Loudness $l[n]$ was also encoded using simple A-weighting [23]. Finally, an optional latent encoder was done on the target audio, with the intuition that the residual information from $z[n]$ captured musical timbre. The latent space is thus a tuple $(f_0[n], l[n], z[n])$, which is used as input to a recurrent decoder. The output of this decoder parameterizes the differentiable synthesis modules, which, when combined together, render synthesized audio.

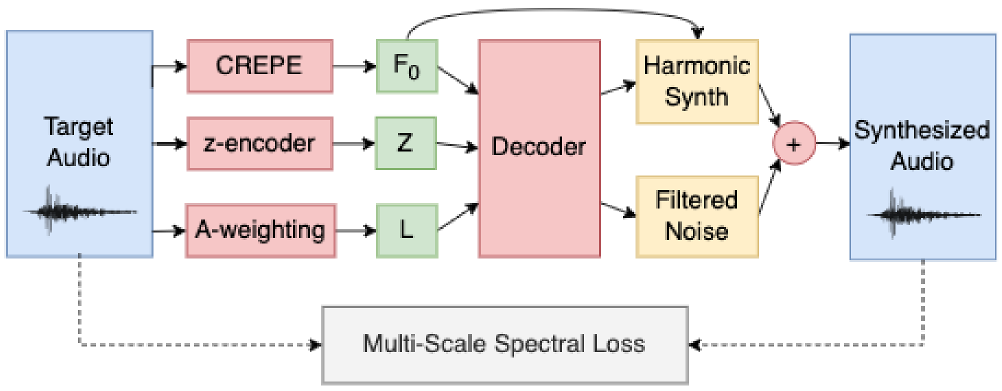


Figure 2.2 A diagram of the DDSP Autoencoder Architecture.

The synthesis method in DDSP uses spectral modeling synthesis (SMS) with a harmonic synthesizer component and a filtered noise component. The harmonic component has as input a bank of harmonic amplitudes (from the decoder model) and the fundamental frequency $f_0[n]$. It sums up a bank of oscillators with frequencies $\omega_k = k \cdot f_0$ being integer multiples of the fundamental. The harmonic signal $s[n]$ is formulated as:

$$s[n] = \sum_{k=1}^{K} A_k[n] \sin(2\pi\phi_k[n])$$

where $A_k[n]$ is a time-varying amplitude for the k-th harmonic and instantaneous phase $\phi_k[n]$ is obtained by summing $\omega_k[n]$ with some initial phase $\phi_{0,k}$:

$$\phi_k[n] = \sum_{m=0}^{n} \omega_k[m] + \phi_{0,k}$$

The DDSP-filtered noise component applies a time-varying FIR filter to random uniform noise. With the convolution theorem for the Fourier transforms, the component takes an input of vectors $\boldsymbol{H}_l$, interpreted as the frequency-domain transfer function, and multiplies them with frames of the discrete Fourier transform (DFT) of uniform noise $\boldsymbol{Y}_l = \boldsymbol{H}_l \boldsymbol{U}_l$ ($u[l] \sim \mathcal{U}_{[-1,1]^L}$, where L is the length of the output signal). The frames of $y[n] \longleftrightarrow \boldsymbol{Y}_l$ are combined together with overlap-add.

Recently, research in the use of DDSP for the procedural generation of car engine sounds has been done with exciting results [36]. The approach was in a similar manner as to ProVE; however, the collection of control variables (denoted γ in this thesis) paired with corresponding audio (denoted x) required recording the RPM of an actual car's onboard diagnostics system. This approach to collecting (γ, x) pairs is, although physically inspired, a lengthy process and difficult to do for small-scale projects. This thesis focuses on environmental sounds obtainable through online sound packs and simpler recording processes one can do from home.

2.3 Neural Audio Analysis - Machine Listening

As stated by Andy Farnell in [19],

"Good sound design is more analysis than synthesis. Most of it is component analytical, reduced, critical, and semantic listening."

Machine Listening aims to develop algorithms for a machine to perceive audio in a similar manner that humans do, encoding semantic information about a sound class and using that semantic information to solve problems like recognition or derivative generation. Unfortunately, little research is found where the motivation of controllable neural audio synthesis methods is on how a human audiates different sounds they have experienced and encoded. This focus on “machine audiation” would connect machine listening, a machine’s perception of audio, with neural audio synthesis, a machine’s audiation from the semantic information retrieved from machine listening. More specifically, a connection between neural audio synthesis methods with approaches to sound event detection (SED) is motivated by the way humans internally audiate sounds. The current approaches to SED are data-driven, trained to classify multiple classes per temporal window [1, 37]. Models used in the state-of-the-art SED use convolutional recurrent neural networks [6] and transformers [21].

One can internally audiate a dog barking sound in a busy city street, just like one can internally visualize a dog in any environment. This is what motivates the methodology of this thesis, where ProVE learns “machine-interpretable” control variables, and learns a mapping to that control space from “human-interpretable” control.

CHAPTER 3

APPROACH

3.1 ProVE: A General Neural Analysis-Synthesis Abstraction

The "classical" approach to PA relies on the information collected from analysis to create a synthesis model. A "connectionist" approach to PA should be similar except that a learning machine would do the signal analysis instead of the human. The Procedural Variational autoEncoder (**ProVE**) is a 2-stage framework that proposes just that.

The first step in ProVE is to train an autoencoder model to reconstruct audio x from a specific sound class. The encoder $z = f_\theta(x)$ and decoder $\hat{x} = g_\theta(z)$ models are architecture agnostic; however, the quality of outputs may depend on the choice of both. A trained autoencoder learns a latent sequence space $z \in \mathbf{Z}_{class}$ that one can think of as a latent parameter space to the synthesis module (the decoder $g_\theta(z)$), learned by the analysis module (the encoder $f_\theta(x)$). Of course, this latent space of sequences is not interpretable by a human, so a second step is needed to allow for a controllable model.

The second step is to map a set of control variables to the latent sequences learned in the first step. These control variables $\gamma \in \mathbf{\Gamma}_{class}$, which are proxied through classical signal analysis on audio x, are paired with $x \in \boldsymbol{X}_{class}$ of the sound class dataset used to train the autoencoder. In each example model demonstrated in this thesis, the collection or approximation of control variables is explained. This step is where the PA model is developed. As mentioned in the background section, most PA models for environmental sounds have a synthesis module which is a function of the control parameters γ and a uniformly distributed sequence $u \sim \mathcal{U}([-1, 1])$. The control parameters influence the operations done on the random sequence, giving

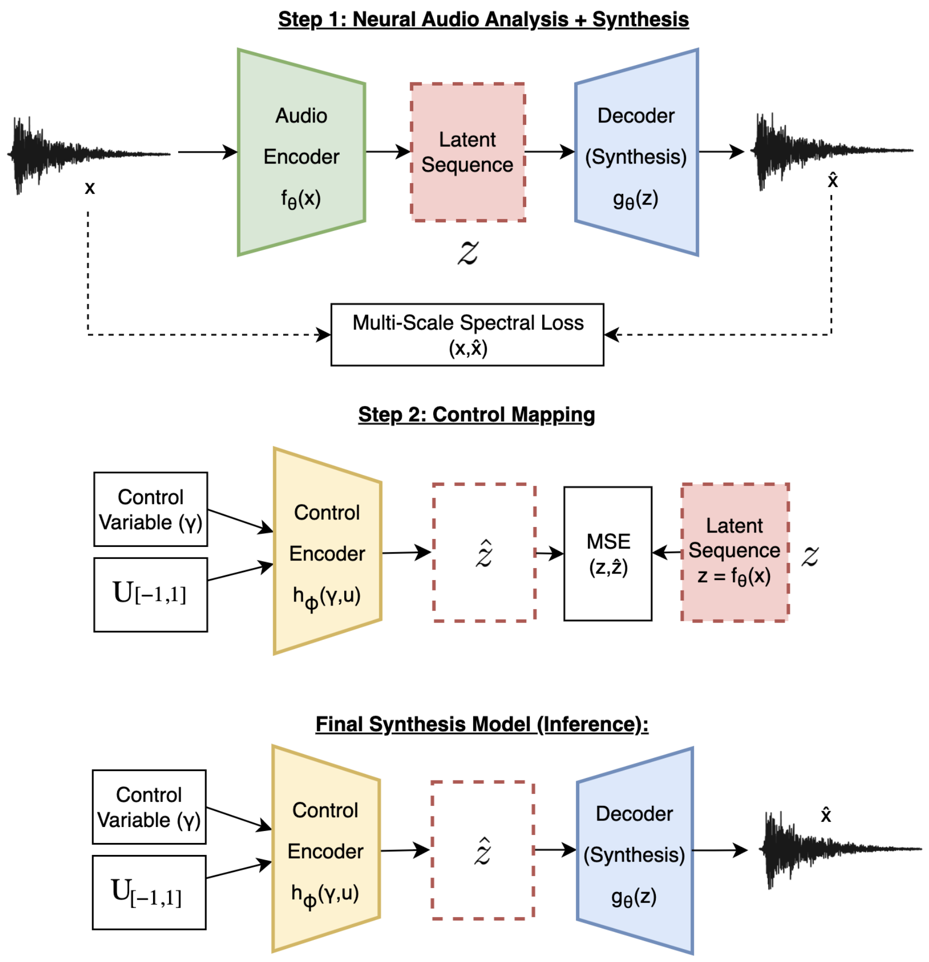

Step 1: Neural Audio Analysis + Synthesis
x
Audio Encoder $f_\theta(x)$
Latent Sequence
z
Decoder (Synthesis) $g_\theta(z)$
$\hat{x}$
Multi-Scale Spectral Loss $(x,\hat{x})$
Step 2: Control Mapping
Control Variable (γ)
$U_{[-1,1]}$
Control Encoder $h_\phi(\gamma,u)$
$\hat{z}$
MSE $(z,\hat{z})$
Latent Sequence $z = f_\theta(x)$
z
Final Synthesis Model (Inference):
Control Variable (γ)
$U_{[-1,1]}$
Control Encoder $h_\phi(\gamma,u)$
$\hat{z}$
Decoder (Synthesis) $g_\theta(z)$
$\hat{x}$

Figure 3.1 The two stage framework of ProVE: (1) Neural Analysis-Synthesis; (2) Learning a mapping from "human-interpretable" to "machine-interpretable" control learned in step 1; (3) The final model is the composition of the mapping from step 2 and the decoder from step 1.

variation to the synthesis. In the second stage of ProVE, a control encoder model $h_\phi(\gamma, u)$ takes in as input the control variable γ that's paired with audio x, as well as $u \sim \mathcal{U}_{[-1,1]}$, which is assumed to be paired with x. The control encoder $\hat{z} = h_\phi(\gamma, u)$ is then trained to optimize a distance loss to the latent sequence of the audio encoder from step 1 $z = f_\theta(x)$, attempting to map an interpretable control variable to a seemingly uninterpretable one with influence from an assumed random sequence $u \sim \mathcal{U}_{[-1,1]}$ to infuse some sort of variation.

In the demos of the subsequent sections, two encoder architectures were tested. The first encoder follows the architecture of the original DDSP paper [16], where mel-frequency cepstrum coefficients (MFCCs) are calculated, normalized over the channel dimension, and passed through a recurrent network. MFCCs are defined as compact representations of the power spectrum (periodogram) of signals and have been used extensively for extracting temporal features in various fields of audio processing [20]. The second architecture that was tested was a fully convolutional network with stacks of 1D inception-like modules where 1D convolutions of different kernel sizes and dilation rates are concatenated in the channel dimension. Both encoders were used and gave similar audio qualities, however, the former MFCC encoder was used as the primary encoder, as it used significantly less parameters. This thesis focuses on audio with a sample rate of $f_{s_a} = 16\text{kHz}$ and a control rate of $f_{s_c} = 250\text{Hz}$, so the encoder has a depth of 6 with a downsampling scale of 2. Each of the inception-blocks of the second encoder type have kernel sizes of 3 and 5 and dilation rates of 1 and 2. The latent space is 512-dimensional and is restricted to be in the range $z \in [-1, 1]^{D_z}$ by using the **tanh** function as a final activation of the encoder.

The decoder model $g_\theta(z)$ used is a recurrent neural network parameterizing the DDSP subtractive noise synthesizer explained in section 2.2. The specific architecture starts with an MLP taking an input latent sequence z. These MLPs have a depth of 3 with a dimension of 1024 in each layer. The output sequence of this MLP is fed through a recurrent layer, and the resulting sequence is passed through a final dense layer before being sent to the subtractive synth, as noise magnitudes with a dimensionality of 4096. The synth itself uses a window size of 4096+1. To optimize spectral reconstruction loss, this project uses the same multi-scale spectral loss used in the original DDSP paper, where the loss is the sum of spectral distance losses with various Short-Time Fourier Transform (STFT) window sizes (S_j^i is the j-th window of the magnitude spectrogram from the STFT of audio x with a window size $i \in \{64, 128, 256, 512, 1024, 2048\}$; $\hat{S}_j^i$ is defined the same way using the audio from the model $\hat{x}$):

$$L_i = \mathbb{E}_j[||S_j^i - \hat{S}_j^i||_1 + \alpha|| \log S_j^i - \log \hat{S}_j^i||_1]$$

$$L = \sum_i L_i$$

The control encoder $h_\phi(\gamma, u)$ used for mapping the human-interpretable control variables to the "machine-interpretable" control in the second step of ProVE is similar to the MFCC encoder, where the control variable $\gamma \in \mathbf{\Gamma}_{class}$ paired with random uniform noise $u \sim \mathcal{U}_{[-1,1]}^{D_r}$ are first normalized over the channel dimension and passed through a recurrent network. The combination of this model with the decoder learned in the first step of ProVE gives us the full PA model.

One may ask why the models demonstrated in this thesis use $f_{s_a} = 16\text{kHz}$ when the goal is high-fidelity audio. Readers are advised to remember that high-fidelity in the case of developing models that synthesize audio is the measure of how well a model produces sounds of a sound class. That is, the output of a decoder $\hat{x} = g_\theta(z)$ is considered high fidelity if the density $p(x|z)$ associated with sound class $\mathbf{X}_{class}$ is high for $\hat{x}$. The reason why the models in this thesis use $f_{s_a} = 16\text{kHz}$ is because classical

PA models synthesize 16 kHz audio which are less likely to be a part of $\mathbf{X}_{class}$ than 16 kHz audio of real recordings. It is a starting point, as 16 kHz was the sample rate from the original DDSP paper [16], and it was shown that the DDSP autoencoder could be scaled up to synthesize with the standard 48 kHz sample rate [36].

3.2 ProVE on Environmental Sounds

The first step of ProVE is the same for all sound classes, however, the process of defining control variables for different environmental sound classes is explained in this section. The sound classes demonstrated in this thesis are Footsteps and Pouring Water.

3.2.1 Footsteps

In [18], a procedural audio model is developed for the controllable synthesis of footstep sounds from bipedal locomotion analysis of foot pressure. The main control variable influencing the puredata[1] patch is the output of a ground reaction force (GRF) curve generator. The GRF is the reactive force the ground applies back toward the human as its feet push against it. The generator is a combination of 3 polynomial segments, one for the initial hit of the heel of the foot, another for the transition rolling from the heel to the ball of the foot, and another for the final push from the ball of the foot as it leaves the ground. This curve is then used as the excitation to a synthesis module, which outputs a synthetic footstep sound depending on the ground material the synthesis module is modeling. Since a footstep sound is dependent on the material being modeled, the idea is to either have a synthesis module for each, all of them having the same excitation curve, or train a model with a parameter for material selection, as in [11].

[1]*puredata* is an open source visual programming language used for modular synthesis: https://puredata.info/

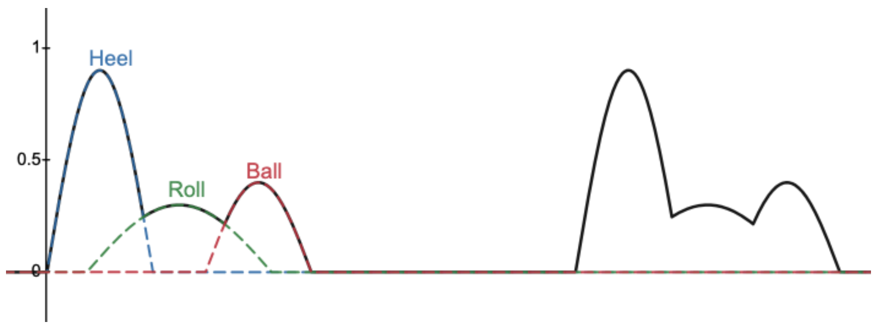


Figure 3.2 GRF curves as 3 polynomial segments.

It is the same for the footstep models in ProVE, dividing the footstep sound class by ground materials. To train the control encoder, there has to be a defined control variable γ to be passed along with the uniform sequence u, paired with input audio x. The control variable for footstep sounds was simply defined as the envelope of the audio, with the motivation that the magnitude of the exertion of force influences the magnitude of the conversion of energy which influences the loudness of an impact-based sound.

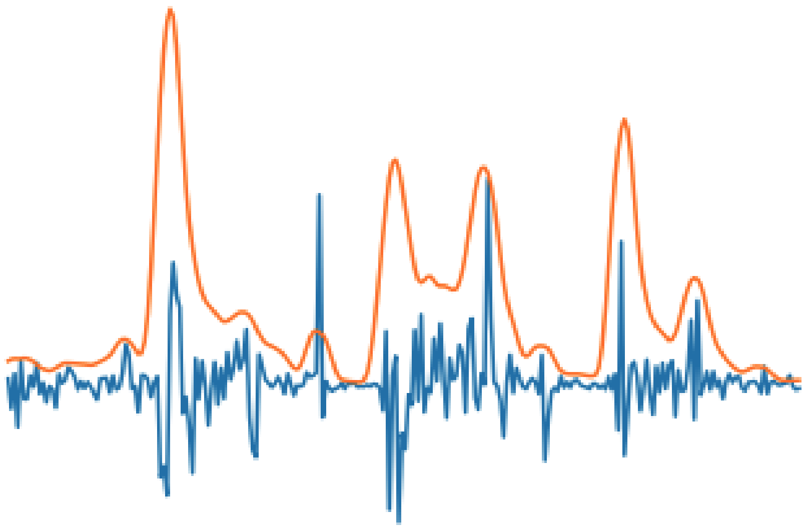


Figure 3.3 The envelope of footstep sounds calculated as the absolute value of the discrete Hilbert transform. This envelope is smoothed either by average pooling, low pass filtering, or Laplacian smoothing.

Hence, the training of the control encoder is done by calculating the envelope of the sounds as γ, passing (γ, u) as input to the control encoder to get $\hat{z}$, and optimizing with respect to a distance loss. Inference on ProVE trained on footstep sounds of a specific material is then done with the same tri-segment GRF curve as input to the control encoder, along with a sample sequence from $\mathcal{U}_{[-1,1]}$. The output of the control encoder $\hat{z}$ is then passed to the decoder model in step 1 for the synthesis of a footstep sound with loudness in relation to the input GRF curve. Figures 3.4 - 3.11 display samples of the full ProVE model for footstep sounds on the surfaces of gravel, dirt, grass, and wood. The even numbered figures show log-scaled spectrograms of the footstep sound from the training dataset and the ProVE spectral reconstruction from the autoencoder $\hat{(}x) = (g_\theta \circ f_\theta)(x)$, as well as the smoothed audio envelope which is used as the proxy to the GRF of the footstep. The odd numbered figures display the log-scaled spectrogram of using the ProVE model with Farnell's GRF curve generator, along with the curve used as input to the model. Some of the curves have additional random curves to account for small debris, like for the gravel surface texture.

The results from the evaluation section of this thesis demonstrate ProVE on two different sets of footstep datasets. The first is a small archive of footsteps from a blog post on PremiumBeat.com[2] which discussed footstep sound effects in sound design. This set had footsteps separated by ground material with varying walking and running speeds in order to test ProVE's ability to learn a higher fidelity synthesis while also retaining the same expressivity of classical PA models. The second set was the same collection of high-heel footstep sound effects from the Zapsplat[3] website used in [11] in order to compare ProVE to an adversarial approach. This set contained one-shot sounds and also separated these by ground material; however, the demonstrations in the evaluation section show a ProVE model which was trained on a long segment

[2]https://www.premiumbeat.com/blog/40-free-footstep-foley-sound-effects/
[3]https://www.zapsplat.com/sound-effect-packs/footsteps-in-high-heels

of audio with footsteps walking on a specific material for about 10 seconds. The rationale behind continuing to train with long segments of walking audio instead of training on one-shots is that audio is temporal, and it is infeasible to increase the fidelity of PA models by bounding the analysis-synthesis approach to look at a brief moment in time. Bipedal acceleration perceptually will sound very different than bipedal deceleration, and that is due to how the control variables (GRF Curve in this case), as well as its corresponding sound, behave over time.

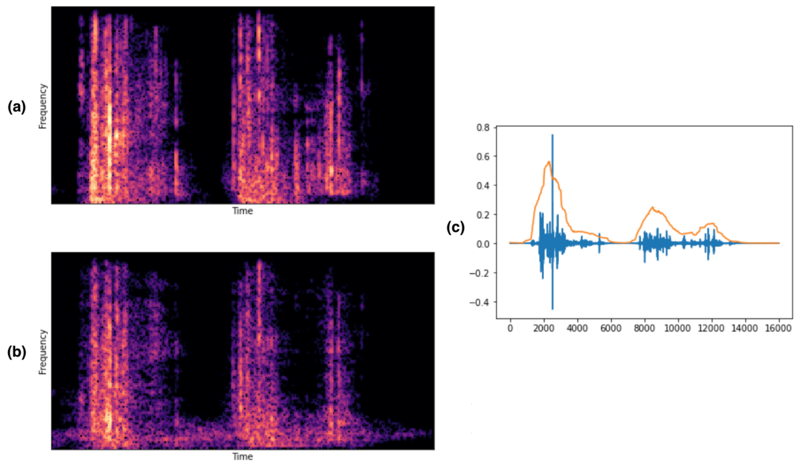


Figure 3.4 (a) Log-scaled spectrogram of Footsteps on Gravel Texture (Real Recording); (b) Log-scaled spectrogram of Footsteps on Gravel Texture (ProVE Reconstruction); (c) Envelope as control variable proxy.

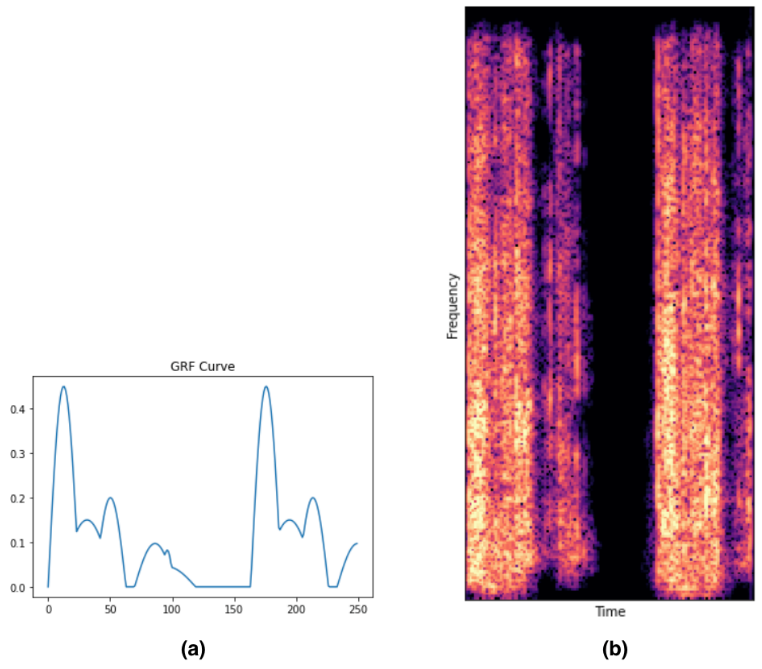


Figure 3.5 (a) Control variable as input to Control Encoder (along with random uniform sequence). (b) Spectrogram of ProVE Inference for Footsteps on Gravel Texture.

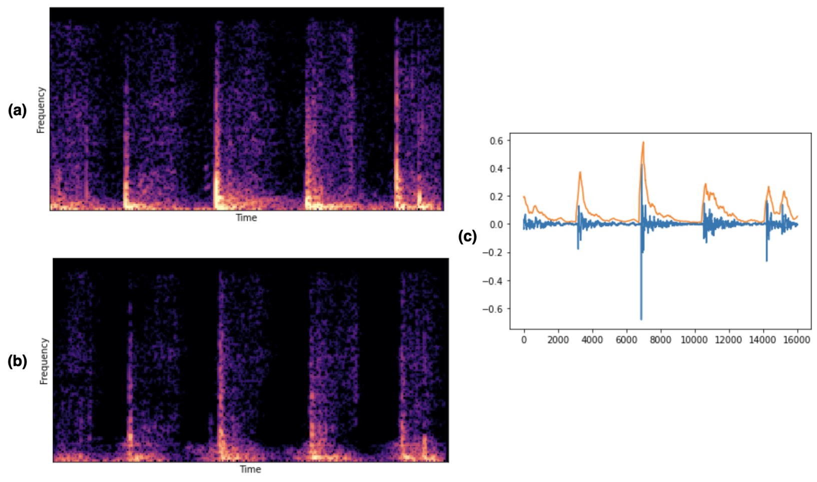


Figure 3.6 (a) Log-scaled spectrogram of Footsteps on Dirt Texture (Real Recording); (b) Log-scaled spectrogram of Footsteps on Dirt Texture (ProVE Reconstruction); (c) Envelope as control variable proxy.

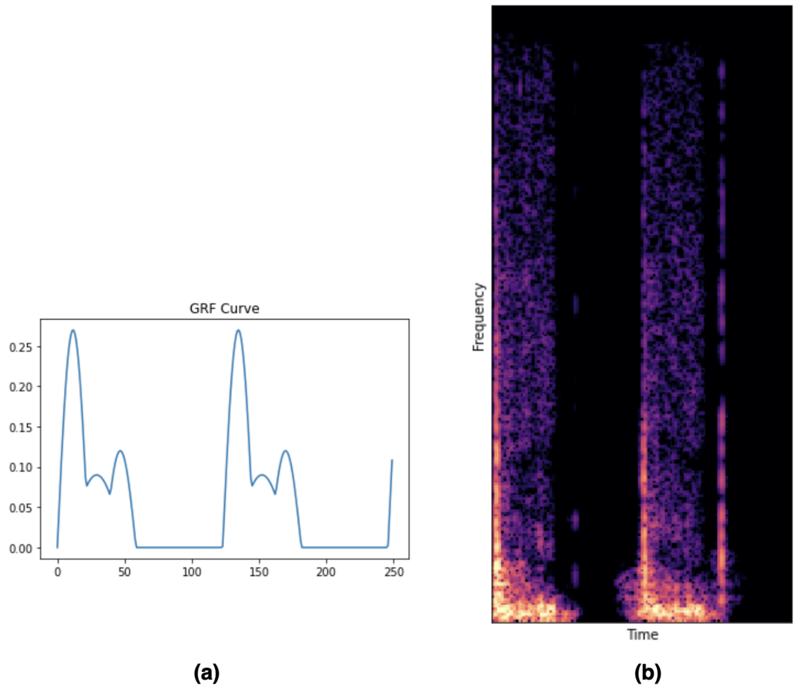


Figure 3.7 (a) Control variable as input to Control Encoder (along with random uniform sequence). (b) Spectrogram of ProVE Inference for Footsteps on Dirt Texture.

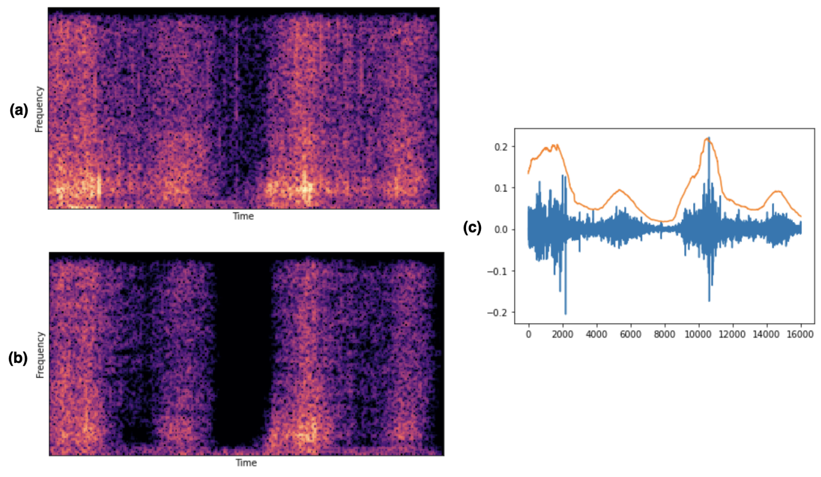


Figure 3.8 (a) Log-scaled spectrogram of Footsteps on Grass Texture (Real Recording); (b) Log-scaled spectrogram of Footsteps on Grass Texture (ProVE Reconstruction); (c) Envelope as control variable proxy.

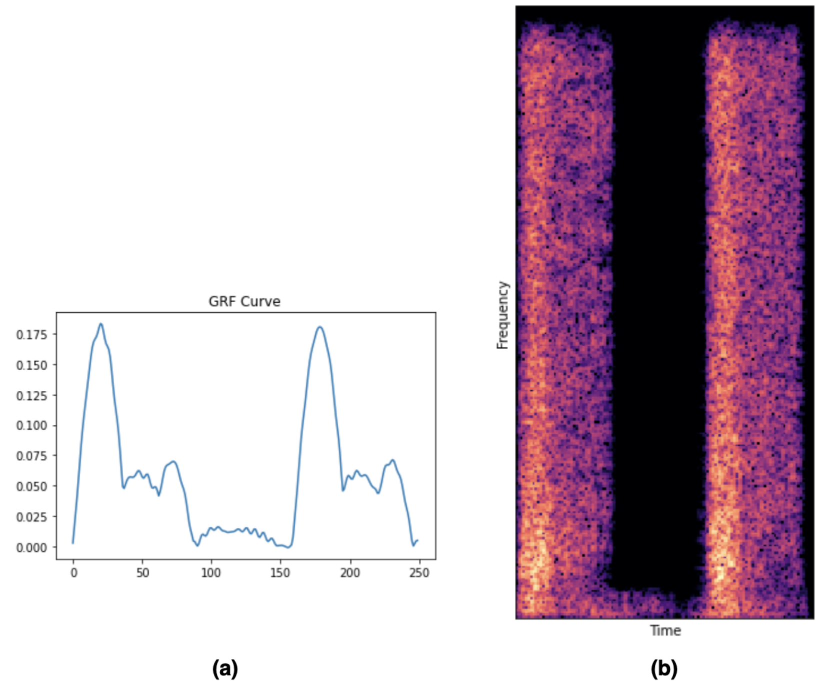


Figure 3.9 (a) Control variable as input to Control Encoder (along with random uniform sequence). (b) Spectrogram of ProVE Inference for Footsteps on Grass Texture.

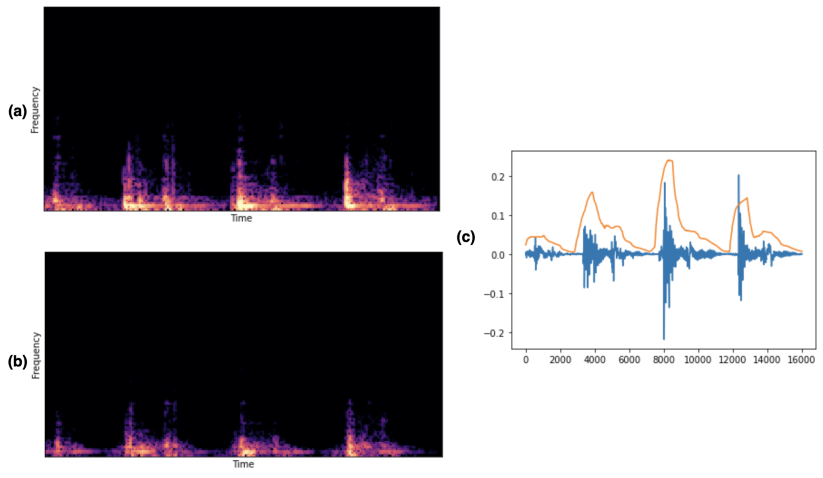


Figure 3.10 (a) Log-scaled spectrogram of Footsteps on Wood Texture (Real Recording); (b) Log-scaled spectrogram of Footsteps on Wood Texture (ProVE Reconstruction); (c) Envelope as control variable proxy.

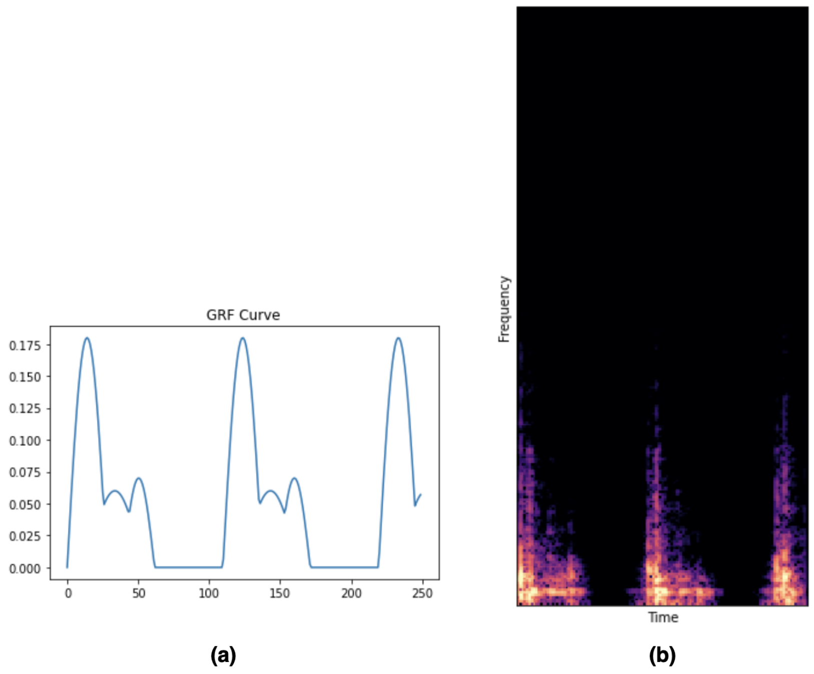


Figure 3.11 (a) Control variable as input to Control Encoder (along with random uniform sequence). (b) Spectrogram of ProVE Inference for Footsteps on Wood Texture.

3.2.2 Pouring Water into A Glass

The control variable proxy for sounds of footsteps was simple to define by using the smoothed envelope of the audio. In this next section, the same proxy is used in modeling the sound class of pouring water into a glass, however, the purpose of describing the process done in modeling this sound class is to demonstrate the simplicity of training a ProVE model with simple user-defined control variables. In this small experiment, a ProVE model for the specific sound class of pouring water into a glass cup is learned. The motivation in the defined control variable is as simple as understanding that the sound of pouring water differs depending on how full the glass cup is. The shape and volume of the glass cup are fixed, with the assumption that the model learned only synthesizes sounds of that specific glass cup. The control variables of this model were simplified to the envelope ($\gamma^{<0>}$) and how full the cup is ($\gamma^{<1>} = 0$ being completely empty and $\gamma^{<1>} = 1$ being completely full).

Audio was recorded with the cup at a starting capacity $\gamma_0^{<1>} \in [0,1]$ and filled to an ending capacity $\gamma_T^{<1>} \in [\gamma_0^{<1>}, 1]$. Since audio samples are recorded in this way, the discrete control signal is assumed to be the linear interpolation from $\gamma_0^{<1>}$ to $\gamma_T^{<1>}$, with a length of T, the recorded audio length. With $\gamma = (\gamma^{<0>}, \gamma^{<1>})$ defined, and $u \sim \mathcal{U}_{[-1,1]^T}$ paired with audio x, the second step of ProVE can begin, mapping the control and uniform sequence to the latent space learned in the first step. Inference of this ProVE model is done by inputting cup capacity $\gamma^{<1>} \in [0,1]^T$ and the input of an envelope generator, defined as the summation of the square of sinusoidal functions with random frequencies and initial phases:

$$\gamma^{<0>}(t) = \sum_{n=1}^{N} a_n \sin^2(2\pi\omega_n t + \theta_{0,n})$$

This envelope generator was approximated by observing the envelopes of the real recordings.

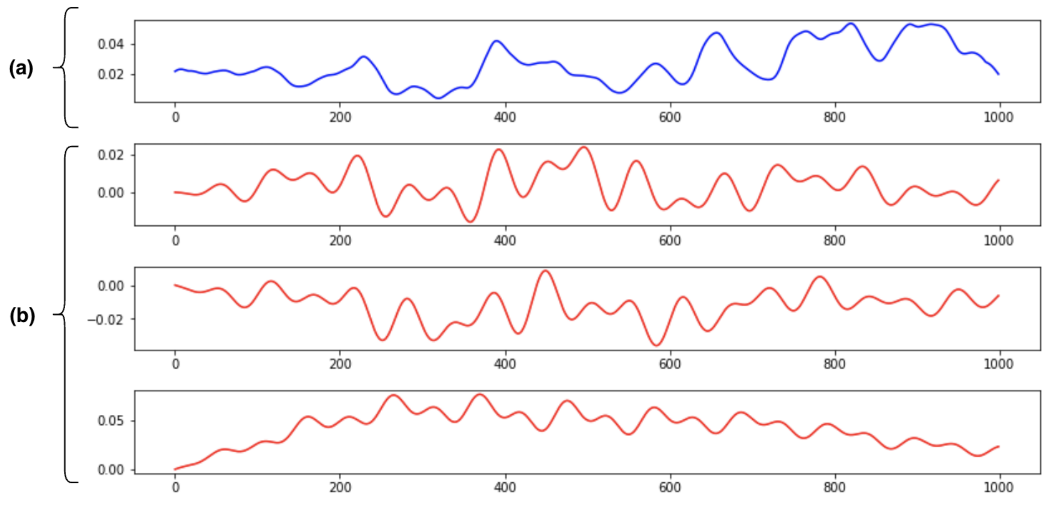


Figure 3.12 (a) An actual envelope calculated from real pouring water recordings; (b) Samples from $\gamma^{<0>}(t)$, of the described envelope generator.

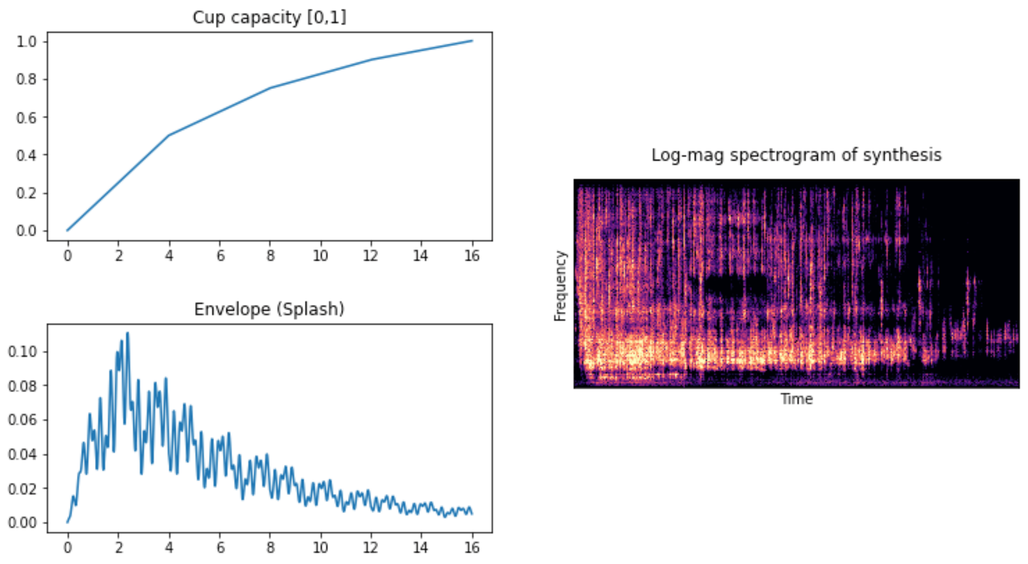


Figure 3.13 ProVE inference of the pouring water model. The input envelope is the product of a negative exponential and the envelope generator. The plots on the left are the control variables, which, when fed into the control encoder along with random uniform noise $h_\phi(\gamma, u)$, gives an approximation to the latent sequence $\hat{z}$, which is then passed to the decoder $g_\theta(\hat{z})$.

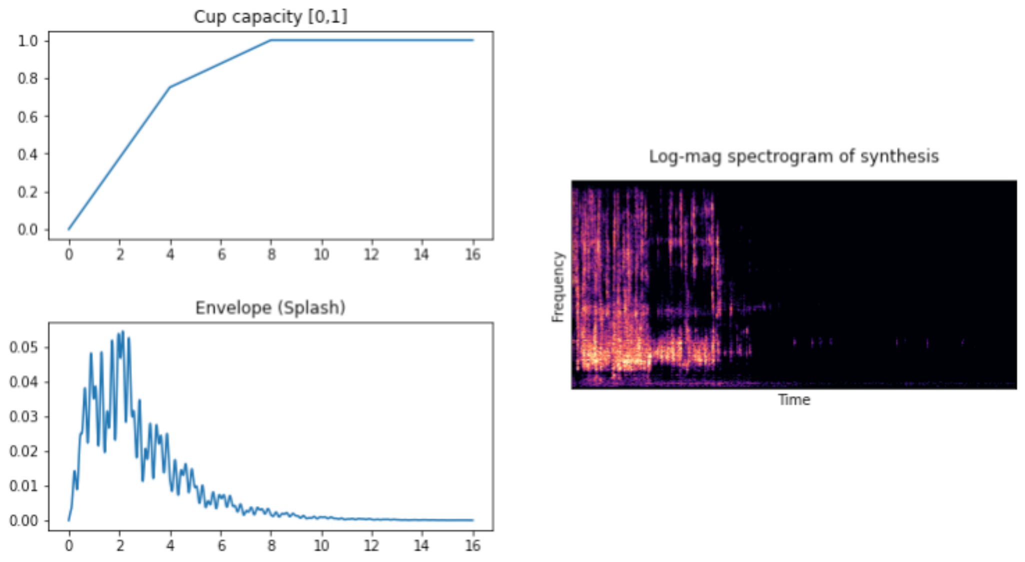


Figure 3.14 Another example of synthesizing water pouring into a glass cup with a shorter sound event duration. The full synthesis is still 16 seconds.

CHAPTER 4

EVALUATIONS AND RESULTS

This section introduces the choices in objective and subjective evaluations of the models learned in ProVE. These evaluations aim to answer the question of whether or not these models are higher in fidelity than classical procedural audio models. Although the second step of ProVE of mapping human-interpretable control to (machine-interpretable) latent control makes ProVE models expressive by definition, the evaluation of the expressivity must still be evaluated. However, due to time constraints, the proposed approach is only described and not conducted.

4.1 Objective Evaluation

Although there is no perfect objective evaluation metric, one can utilize data-driven and perceptually motivated metrics to evaluate how close one synthesis model is to the data distribution it is designed to model. The Frechét Audio Distance (FAD) [30] is used as an objective evaluation method comparing the synthesis from ProVE models to classical PA models. The idea behind FAD is to calculate two sets of embedding vectors from a pre-trained audio classification model: one set of embeddings from the real audio distribution e_{real} and another set from the output of a generative model e_{model}. The next step would be to estimate multivariate Gaussians with the sample means and covariances of e_{real} and e_{model}, and finally calculate the Frechét Inception Distance [25] between the two estimated distributions. This metric has been used in the past to evaluate generative audio models [42]. The Maximum Mean Discrepancy (MMD) [22] was also used as another distribution distance metric to compare between synthesis approaches. The MMD is calculated the same as the FAD, but only as the distance between sample mean embeddings of both distributions[1]. For the objective evaluations done in this thesis, the FAD was calculated from embeddings from a pre-trained audio classification model called VGGish [24] and the MMD was calculated from pretrained OpenL3 [12, 2] embeddings, both models being trained on audio classification datasets.

Another objective measure shown was the accuracy of an audio classification model in classifying the resulting synthesis as the target sound class. A pretrained yamnet[2], a model that was used for classifying 521 different audio classes, calculated a set of 521-dimensional vectors which are classification scores for each synthesis model and the real audio dataset. These scores are interpreted as the confidence that the classifier has in labelling the input audio as some class. In the case of footsteps, the label that was analyzed had the class-name "*walk, footsteps.*"

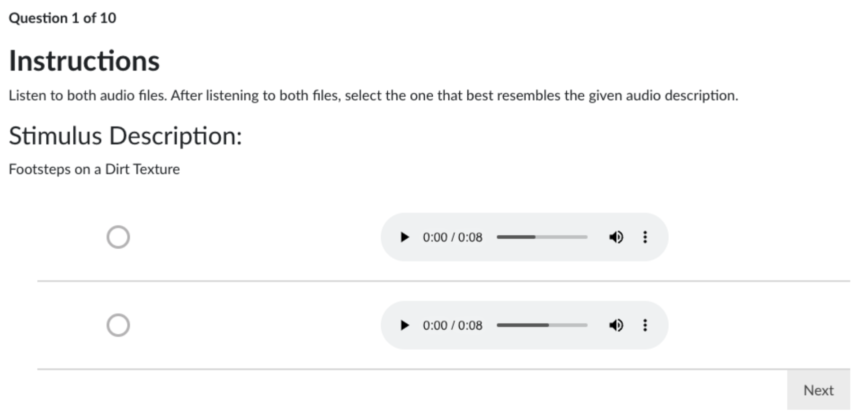
The final objective measurement used is the Perceptual Evaluation of Audio Quality (PEAQ), which is carried out for the evaluation of the basic audio quality (BAQ) of reconstructed audio from ProVE. The implementation of PEAQ used here is PEAQ Basic and is publicly available as MATLAB code from McGill University [51].

4.2 Subjective Evaluation

The main interface for subjective evaluations was the Reproducible Subjective Evaluation (*reseval*) framework [40], which allows for the creation of simple and efficient subjective evaluation tests and the crowdsourcing of these tests as Human Intelligence Tasks (HITs) on Amazon Mechanical Turks. This interface was modified to also show the participants a description of the stimulus being presented on a test page. For example, for an AB test of footstep sounds, the participant is tasked to

[1]Typically, the MMD is calculated as the distance between sample mean embeddings after applying a kernel function to them.
[2]https://tfhub.dev/google/yamnet/1

choose whether stimulus A or B best matches the stimulus description of "Footsteps on a Gravel Texture." The criteria for inclusion in crowdsourcing these sound comparison tests on Amazon Mechanical Turks were limited to those above or at the age of 18 and those who are not diagnosed as deaf or hard of hearing. The HITs also had criteria that required a participant to have at least 1000 approved HITs completed, as well as have a sufficiently high acceptance rate of 99%, in order to ensure all of the responses were reliable for use in analysis. An additional headphone screening test is implemented in *reseval* for testing the competency of a participant in hearing the auditory stimuli of each test page. All participants were compensated a base payment, regardless of whether or not they qualified for inclusion. Those that fit the criteria of inclusion, passed the headphone screening test, and completed the full listening survey, were compensated a bonus payment. Each test called for 50 participants to answer 10 pages of comparing two pairs of audio to their stimulus description.


Question 1 of 10

Instructions

Listen to both audio files. After listening to both files, select the one that best resembles the given audio description.

Stimulus Description:

Footsteps on a Dirt Texture

Figure 4.1 An example question for subjective evaluation of synthesis approaches. In this case it is a test on footstep sounds. The interface is created using a modified version of *reseval* [40].

.

4.3 Results

The main objective of ProVE is to provide a general framework for learning PA models which surpass classical PA models in fidelity, but still retaining the same level of expressivity and controllability that adversarial-based approaches lack. This section is separated into the results of analyzing these two goals by first discussing sound fidelity evaluations, and then discussing the controllability of ProVE models.

4.3.1 Sound Fidelity

As mentioned previously, there is no ideal objective metric for evaluating environmental sounds, but the widespread inclusion of perceptually motivated audio metrics allows for confidence in their utilization as a reliable replacement. Table 4.1 displays the same distribution metrics used in [11], the paper displaying the adversarial approach to learning a distribution of footstep one-shots. In order to use the metrics shown from that paper faithfully and reliably, the same sampling and evaluation processes are strictly followed. This was done because a pretrained model of the adversarial approach was not readily available. Each column of the table is the distance between two audio distributions, for example, the column $(ProVE, Heels)$ would hold the results of evaluating the Fréchet Audio Distance (FAD) on the top cell and the Maximum Mean Discrepancy (MMD) on the bottom cell between the sample datasets of the footstep ProVE model and the training dataset *Heels*. The FAD and the MMD, are two objective metrics used by the previously mentioned adversarial approach to learning one shots of footstep sounds. These two distribution distances are calculated among 3 different synthesis methods for footsteps (PA, GAN, and ProVE) and the real audio footstep dataset from Zapsplat. *PA* denotes the classical procedural audio model for footsteps [18], *ProVE* denotes the ProVE model, and *GAN* represents the adversarial approach. *Heels* represents the training distribution used for the ProVE model and the adversarial approach and *Misc* represents the distribution of the rest of the footsteps collected from the Zapsplat website. One would desire a lower FAD and MMD with *Heels* to show that the synthesis of their model closely matches the training distribution of real audio.

	(PA, Heels)	(ProVE, Heels)	(GAN, Heels)
FAD	10.19	2.41	6.06
MMD	N/A	42.71	53.5

	(PA, Misc)	(ProVE, Misc)	(GAN, Misc)
FAD	11.53	3.96	3.13
MMD	N/A	88.48	56.4

	(PA, ProVE)	(Heels, Misc)
FAD	8.10	6.43
MMD	N/A	N/A

Table 4.1 Frechét Audio Distances (FAD) and Maximum Mean Discrepancies (MMD) of Real and Synthesized Footstep Sound Datasets

The cells of table 4.1 show the FAD or MMD between two sample distributions of footstep sounds. The top table shows the distances between the synthesis methods to the training dataset *Heels*, the middle table shows the distances between the synthesis methods to the *Misc* dataset, and the bottom table shows the distances between the adversarial approach and the ProVE model and the distances between the *Heels* and *Misc*. What is unsurprising from analyzing the FADs is that the classical PA model is the furthest away from both datasets, enforcing the fact that classical PA models lacks realism. What is exciting is that the ProVE model is significantly closer to the datasets of real recordings of footsteps. This was one of the objectives that this thesis aimed to show: that ProVE models have higher fidelity than classical PA models while still retaining the expressivity that the one-shot adversarial approach lacks. The table also shows that the ProVE model is closer to the training data (*Heels (Zapsplat)*), than the adversarial approach is, the ProVE model having an FAD of 2.41 and the adversarial approach having an FAD of 6.06. However, the

FAD between the models and the dataset consisting of the rest of the footstep sounds from the Zapsplat website[3] (*Misc (Zapsplat)*) shows that the adversarial approach has an FAD of 3.13, whereas the ProVE model is further away with an FAD of 3.96. The authors of the adversarial approach interpret this as the generator having the capability of synthesizing samples outside of the distribution of training data. The ProVE model seems to have a consistent distance in both datasets, where the model is closer to the training *Heels* dataset, the distribution it is attempting to model, and it is proportionally further away than the *Misc* dataset. Furthermore, the FAD of the adversarial approach to the *Misc* dataset does not differ significantly to the FAD of the ProVE model to *Misc*. With this, an argument can be made that the ProVE model has the capability of synthesizing samples outside of the distribution of the training data.

The bottom rows of table 4.1 display the Maximum Mean Discrepancies (MMD) between distributions. The results for MMDs are similar to FAD, in that the ProVE model is closer to the *Heels* dataset with a distance of 42.71, compared to the 53.5 from the adversarial approach. The MMDs with the *Misc* dataset shows an interesting result with the adversarial approach, in that it is further away from *Misc* than the *Heels* dataset. This is the opposite of what was shown with FAD. Furthermore, the difference between the two MMDs for the adversarial approach are not significant, but for FAD, they are significantly different. The MMDs of the ProVE model, on the other hand, are consistent with FAD, in that the distance from *Misc* is larger than from *Heels*. The main results from the two objective measures are that the ProVE model surpasses the procedural audio model in terms of fidelity, while being naturally more controllable than the adversarial approach by definition. Along with this, one can argue that the ProVE model also does a better job in matching the original training data distribution, being consistent in its distances.

[3]https://www.zapsplat.com/sound-effect-packs/footsteps-in-high-heels

An interesting result occurs from evaluating synthesized samples and the real sound datasets when using classification accuracy as a metric. The pretrained yamnet used was trained to classify audio data into 521 different classes. The one class that is focused on in this section is the label "*Walk, footstep.*" Table 4.2 shows the confidence probabilities that occur when inputting samples from the footstep sound synthesis methods and the real audio dataset they are trying to model.

Synthesis Method/ Real Audio	Yamnet accuracy (Confidence probability on "walk, footstep")
Zapsplat Real	0.0172
Zapsplat GAN	0.0175
Zapsplat Classical PA	0.0038
Zapsplat ProVE	**0.0357**
PB Real	0.09
PB Classical PA	0.0046
PB ProVE	**0.0357**

Table 4.2 Audio Classification Accuracy of Label "*Walk, Footstep*" From A Pretrained Yamnet Model

What is unsurprising is that classical PA models have a smaller probability on both datasets, lining up with the discussion in the first chapter where it was stated that classical PA models lack the fidelity preferred for immersive medias. On the other hand, what is surprising is that the classification accuracy of the label "*Walk, footstep*" obtained on the ProVE model is greater than the GAN approach and even the real audio dataset. Although purely speculative, these results may be due to the ProVE models emphasizing key characteristics of audio labeled with "*Walk, footstep*" more so than the footsteps in the real datasets and the GAN approach. One can interpret this as the audio from ProVE footstep models sound more like a footstep to yamnet than the other methods and datasets, but further research would be needed to fully understand these results before confirming the speculation. Of course, the main highlight here is that ProVE models beat classical PA models in sounding more like a footstep according to these classification probabilities.

Surface Material (Zapsplat)	ODG [-4.0, 0.0]
Carpet	-2.57
Wooden Deck	-3.54
Metal	-3.56
Pavement	-1.88
Rug	-2.8
Wood	-2.33
Wood Internal	-2.34

Surface Material (PremiumBeat)	ODG (Running) [-4.0, 0.0]	ODG (Walking) [-4.0, 0.0]
Gravel	-1.48	-1.4
Dirt	-1.82	-1.95
Wood	-1.5	-2.98
Grass	-2.09	-2.18

Table 4.3 Objective Difference Grades (ODG) of the Reconstruction of Footstep Sounds Calculated Using PEAQ Basic

The Perceptual Evaluation of Audio Quality (PEAQ) measures basic audio quality (BAQ) through deriving mid-level perceptual features called Model Output Variables (MOVs) [50] and using these as input to a neural network computing the Objective Difference Grade (ODG) between a reference signal and a reconstructed signal, with grades ranging from -4.0 (very annoying impairment) to 0.0 (imperceptible impairment). The ODGs of the ProVE models for each sound class on both footstep datasets are shown in Table 4.3. The ODGs are included in this thesis are calculated in an attempt to give a perceptually-motivated evaluation of the reconstruction from the first step in ProVE, which is the DDSP autoencoder (AE) in this case. In other words, one can interpret the grades from PEAQ Basic as a measure of the capability of the DDSP autoencoder to reconstruct environmental sounds. The left side of table 4.3 shows ODGs of the DDSP autoencoder (AE) model trained on the Zapsplat footstep dataset, where each of the surface material sounds were learned to be reconstructed using a single autoencoder. The right side of the table shows ODGs of a model trained on the footstep dataset from PremiumBeat.com, with varying step speeds (Running and Walking), where each of the surface material sounds were learned to be reconstructed in separate models. From the two tables one can argue that the audio quality is better when further separating a target sound class into subclasses and developing multiple PA models to be chosen during inference.

That is, if a model requires integer valued control variables to model subclasses of an audio class (i.e. surface material selection), then one can train a model per subclass and use a finite index set to select a model. The table also shows some subclasses of footstep sounds being reconstructed with a lower ODG than the rest. On the Zapsplat dataset, the subclasses that had the lowest grades from PEAQ were the *Wooden Deck* and *Metal* subclasses, having ODG's of -3.54 and -3.56 respectively. On the PremiumBeat dataset, the subclass that had the lowest grade was the *Wood* subclass with a slower walkspeed, having an ODG of -2.98. Observing the spectrogram of all of these audio subclasses leads to the speculation that the DDSP AE, with only a filtered noise component, tends to reconstruct sounds with less prolonging frequencies. This speculation makes sense since the sounds with the least ODG's are footstep sounds that have a prolonging resonance when force is acted upon it.

p-value = 6×10^{-7}	Dirt	Grass	Gravel	Wood
ProVE	**0.77**	**0.705**	0.371	**0.81**
PA	0.23	0.295	**0.629**	0.19

Table 4.4 Results of Subjective Evaluation (ProVE, PA)

p-value = 0.1	Dirt	Grass	Gravel	Wood
ProVE	0.5	0.486	0.431	0.387
Real (PB)	0.5	0.514	0.569	0.613

Table 4.5 Results of Subjective Evaluation (ProVE, Real)

Subjective Evaluation Results Tables 4.4 and 4.5 show the results from crowdsourcing the subjective evaluation interface previously described. Two AB tests were conducted, comparing the synthesis of the ProVE model for footstep with the classical PA models and its training distribution, the footsteps from the PremiumBeat dataset. The tables are organized by surface material. The p-values associated with these tables were obtained from a two-sided binomial test, which *reseval* calculates for us during the HITs. Each cell is the ratio of how many participants preferred the ProVE model over the real distribution or the classical PA model. The results show that between ProVE and classical PA, there is a large difference per surface material. This is in contrast with comparing the ProVE model with the real dataset, where the table shows high entropy. It can also be said that most of the participants would choose the synthesis from the ProVE model than the classical PA model for the surfaces of dirt, grass, and wood. However, the majority of the participants more preferred the synthesis of the gravel texture footsteps from the classical PA models than the ProVE model. This is possibly due to the fine grain sounds of the trajectory of small pebbles and rocks from gravel, which is not accounted for by simply using a control variable proxy of the smoothed envelope of footstep sounds.

4.3.2 Sound Expressivity

Although the ProVE model does not surpass the adversarial approach in synthesizing footsteps, the ProVE model surpasses the adversarial approach in expressivity and controllability by definition. The adversarial approach learns to generate one-shot footstep sounds which are limited to the envelope shapes of the one-shots from the Zapsplat dataset. In similar nature as the discussion in the first chapter on using recorded audio vs PA models for sound design, the adversarial approach learns a space of footstep sounds, whereas ProVE learns to synthesize footstep sounds as a temporal process. That is, ProVE models for footstep sounds learn to synthesize the sounds of walking or running given the audio's envelope and assuming a random uniform sample, but are not limited to those envelope shapes during inference.

Due to time constraints, another subjective evaluation was proposed but not posted. This test would be to evaluate the controllability of the ProVE models. For the example of a test for footstep sounds: a participant is shown two audio samples, both synthesized from ProVE models. The first audio sample is synthesized to match the audio context of a person moving with a constant speed, and the second audio sample is synthesized to modify that context into a different one (i.e. from walking to running, running to walking, etc). The participant is then asked whether or not the modified audio sample matches the modification applied, where the modification is described with text as the stimulus description.

4.4 ProVE Application: Draw a GRF Curve

One of the goals of developing ProVE as a general framework for learning data-driven PA models is to show that its high-fidelity sound synthesis and controllability is feasible for interactive use by a sound designer. This would be a step forward in bringing PA models into standardization in sound design workflows. In order to show that models from ProVE have this capability, an interactive web application was developed[4]. The client interface was written in javascript and it allows a user to define control variables for the input to a pretrained ProVE model by drawing a 1D curve on a canvas, select a surface material, and modify walk speed. The drawn curve is then repeated to span 8 seconds with respect to the walk speed. This curve and the selected surface material is sent to an AWS EC2 instance hosting a Flask REST API. The server also hosts a pretrained ProVE footstep model, trained on the Zapsplat Heels dataset.

When a user presses on the "generate" button, the user-defined control variables, along with a randomly generated audio id, are sent to the server, which uses those variables to synthesize 8 seconds of footsteps with the specified surface material. The server responds with the audio back to the client. The client then calls the server again with the same audio id, in order to obtain an image of the magnitude spectrogram of the resulting synthesis, as well as delete it from the server's cache. Finally, the client displays the magnitude spectrogram image and plays the 8 seconds of synthesized audio. The application is simple to interact with, it synthesizes realistic footstep

[4]The demo is publicly available at https://dependanz.github.io/blog/prove.html

sounds, and it is highly controllable. This indicates that the application is feasible to use in a sound design workflow requiring footstep sounds, and further motivates future interactive use cases and projects.

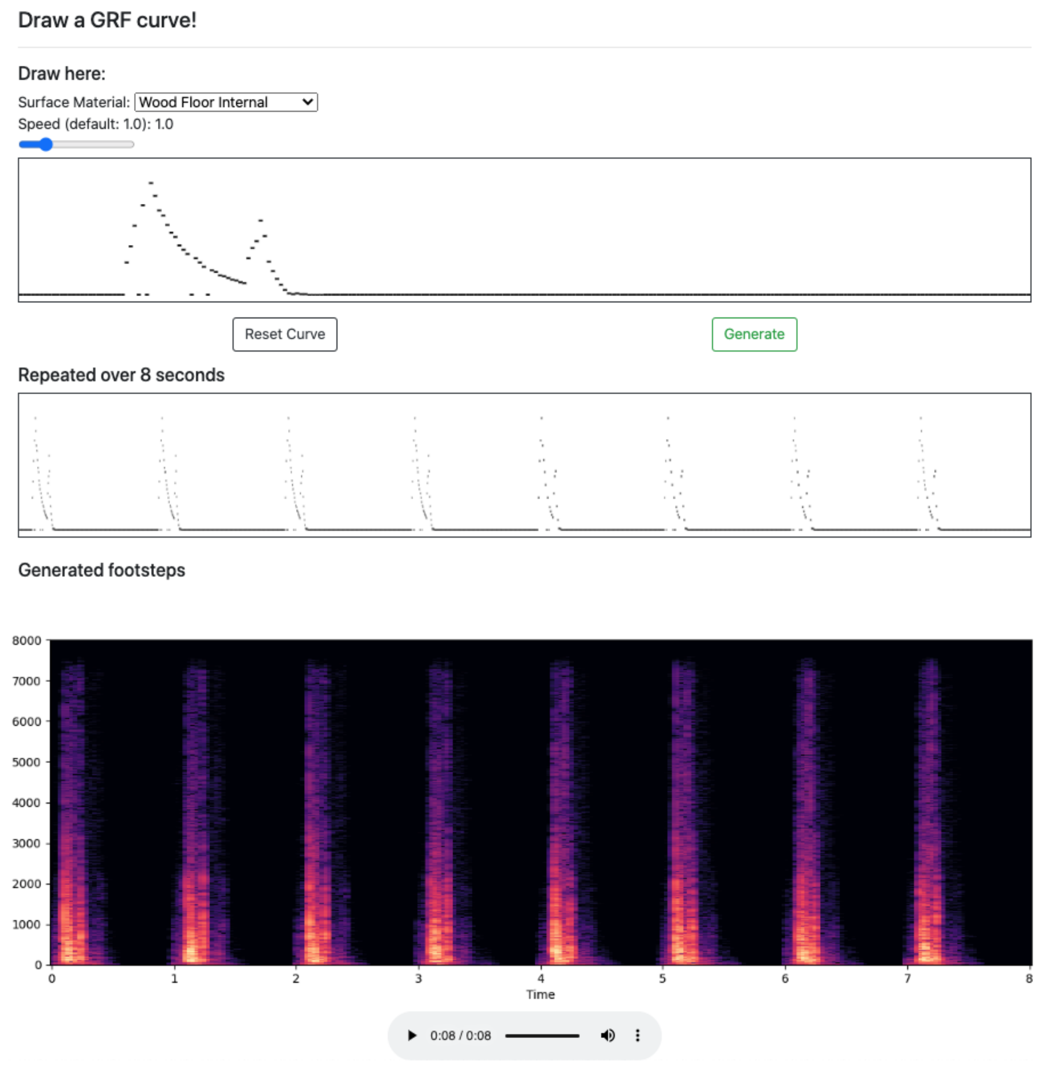


Figure 4.2 ProVE Example Web Application: Draw a GRF to synthesize footstep sounds. A user would draw a curve on the top canvas, select surface materials, set walk speed, and press the green generate button. A REST API responds with the audio from a ProVE model and an image of the magnitude spectrogram of the synthesized audio.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

The goal of this thesis was to develop a general framework for learning high-fidelity procedural audio (PA) models for the use in the sound design of immersive and interactive medias, such as film, games, and virtual environments. To this end, the Procedural (audio) Variational autoEncoder (ProVE) framework was proposed, which learns a data-driven PA model in a way that is loyal to the Analysis-Synthesis abstraction of sound design. ProVE learns these high-fidelity PA models by first learning “machine-interpretable” control variables through an audio autoencoder, and then by learning a mapping from “human-interpretable” control space to “machine-interpretable” control space. ProVE does this by assuming the definition of a PA model is a function of uniform noise, as many PA patches for environmental sounds generate sounds by taking uniform noise and applying transformations and digital filtering. The second stage of ProVE, after training an audio autoencoder, assumes a pair of uniform noise u to audio x and its corresponding control variable γ. A mapping from (γ, u) to z, the latent sequence (or the “machine-interpretable” control) is learned, and is connected to the audio decoder (mapping $z \mapsto x$). Optimal control mappings connected to audio decoders are the controllable data-driven PA models from ProVE (a composite mapping $(\gamma, u) \mapsto x$). In this thesis, the DDSP autoencoder [16] was used as the main neural audio synthesis module for shaping this control space, however, the framework is agnostic to any models that do well in neural audio synthesis.

The main sound classes demonstrated in this thesis were footsteps and pouring water. The latter was demonstrated in order to show the simplicity of using ProVE for a small-scale personal project. For each of the sound classes demonstrated, emphasis is placed on how the “human-interpretable” control variables are defined, as they may be different depending on the type of sound. However, for impact based sounds, like the footstep sounds, the control variable was proxied as the envelope of the sound, with the motivation that the amplitude of a sound corresponds to how much force is applied over some point on a surface.

As a result of objective and subjective evaluations of synthesis samples, one may conclude that ProVE models surpass classical PA models in fidelity, while still retaining the same controllability. For footstep sounds, the use of the GRF curve generator allowed the ProVE model to be expressive in the shape of possible footstep GRF curves, something that is lacking in recent adversarial approaches that focused on generating one-shots of a single envelope shape.

5.2 Future Work

One interesting result that was found through objective evaluations was the classification confidence probabilities of footstep sounds from a pretrained yamnet model. Due to time constraints, what was concluded in that section is purely speculative and may need further investigating.

One limitation of the proposed approach is that training ProVE models relies on the definition of simple control variable proxies directly from classical analysis of audio. The definition was simple for footstep models with the smoothed audio envelopes, however, these proxies may be more difficult to define with classical analysis methods on more complex environmental sound classes. It seems like a reasonable next step to mitigate this limitation of relying on using classical signal analysis methods and try to develop an additional model that approximates control variables of already existing PA models from the real audio distribution that the model tries to match. An example of this limitation was shown in the subjective evaluations, where the classical PA model was more preferred than the ProVE model for footsteps on a gravel surface texture. The speculative reason was that the fine grain sounds of the trajectory of small debris was not accounted for in using the simple GRF curve proxy of smoothed audio envelopes, whereas Farnell's classical PA model maps the GRF curves to densities that approximate the movement of small particles [18].

Another limitation shown was that inference times for the ProVE models are still not practical for use in real-time audio synthesis, which is preferred for interactive medias like games and VR. In order for ProVE models to be adapted for real-time usage, this inference time must be shortened. The latent dimension of the DDSP autoencoder used in this thesis was at a high 512 for a 250 Hz control rate on a 16 kHz sample rate. A research direction that could be taken from ProVE models is to explore neural audio synthesis approaches that have a smaller latent dimension. A paper which synthesizes audio with a real-time variational autoencoder [5] could potentially be investigated as a replacement to the DDSP models. In addition, the models trained for the demonstrations of this thesis only used the DDSP filtered noise components, which does not account for surface textures which have a prolonging harmonic resonance when an impact occurs on it. This was apparent in the objective evaluation section, where the ProVE model had the least objective difference grade (ODG) on the metal surface texture, the surface with the most resonance. This limitation may be alleviated by comparing the ProVE model with only the filtered noise component with a ProVE model that adds the DDSP harmonic component.

Time-permitting, another goal for learning ProVE models was to give a solution to the problem of automatic foley placement in videos, a problem that was recently put under the audio machine learning spotlight. The idea was to develop a model that learns the control variables to the ProVE models directly from the video signals, either as input directly to a convolutional network or having the optical flow as input. The choice of optical flow would have been for event-based sounds like footsteps, where one can follow the path of an actor running or walking.

Recently, denoising diffusion probabilistic models (DDPM) have been extensively researched for generative modeling in the usual modalities. Right now, the second step of ProVE is to assume a sample from uniform noise and pair it with a sample of audio and its associated control. However, this assumption is nothing more than a guess, and a better approach would be to learn a conditional DDPM which goes from the distribution of "machine-interpretable" control, to an isotropic Gaussian, conditioned on the "human-interpretable" control variables.

APPENDIX A

FORMULATION OF THE GRF CURVE GENERATOR

The footstep Ground Reaction Force (GRF) curve generator as formulated by Andy Farnell in [18] is what is used as the control variable to the ProVE model of footsteps. The initial formulation used in this project was 3 half cycles of a cosine to make a 3-component envelope.

$$q(x,a,b,s) = \begin{cases} s \cdot \cos(\frac{\pi}{b-a}(x - \frac{(a+b)}{2})) & \text{if } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

This envelope is then repeated by a parameter phasor, with the magnitudes of each envelope component corresponding to heel, roll, and ball parameters. The walk speed and shortness of step are also added parameters to this GRF generator:

$$g(x) = \max_k(q(x, s_k, a_k, b_k))$$

$$m(x,v) = v \cdot mod(x,2)$$

$$f_{GRF}(x, \omega_{speed}, v_{shortness}) = g(m(\omega_{speed} \cdot x, v_{shortness}))$$

Farnell further improved upon the curve by changing the cosine envelope to a polynomial curve with a steeper beginning to have a curve that better matched actual GRF curves ($N = 3.333$):

$$\kappa(x,b) = -1.5((N(bx)^3) - (bNx))(1 - bx)$$

$$q(x,a,b,s) = \begin{cases} s \cdot \kappa(x-a, \frac{1}{b}) & \text{if } a \leq x \leq (a+b) \\ 0 & \text{otherwise} \end{cases}$$

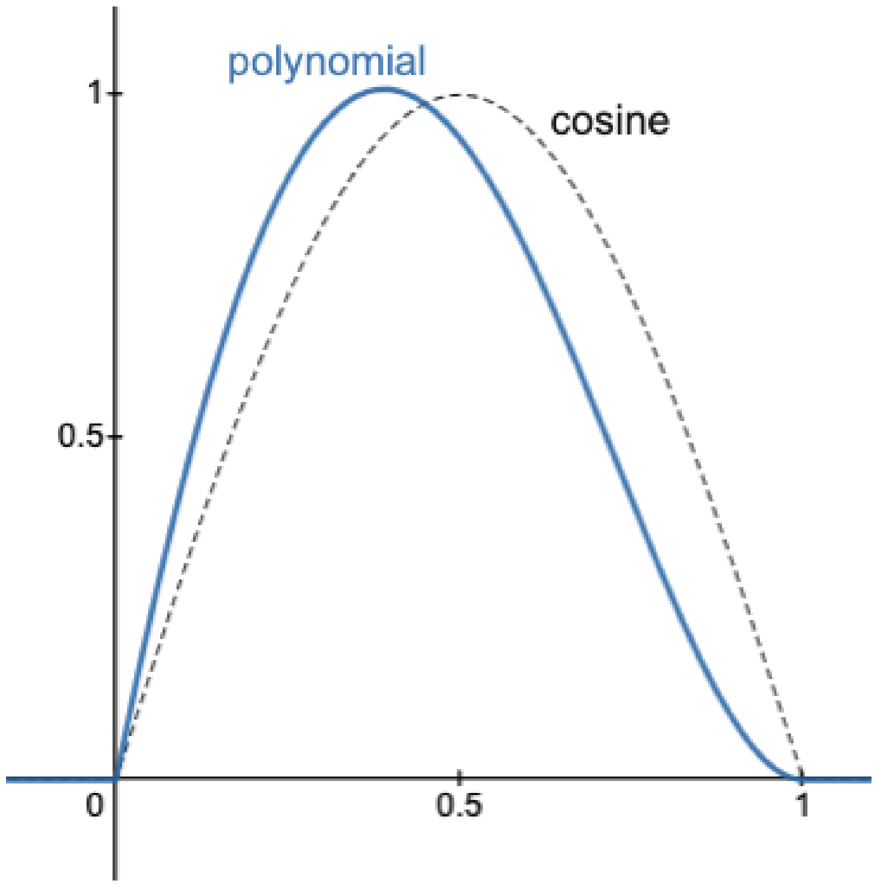


Figure A.1 The modified polynomial (blue curve) as a more realistic force curve than a cosine half angle.

BIBLIOGRAPHY

[1] Sharath Adavanne and Tuomas Virtanen. A report on sound event detection with different binaural features. *CoRR*, abs/1710.02997, 2017.

[2] Relja Arandjelovic and Andrew Zisserman. Look, listen and learn. *CoRR*, abs/1705.08168, 2017.

[3] S. Bilbao. Time domain simulation and sound synthesis for the snare drum. *J Acoust Soc Am*, 131(1):914–925, Jan 2012.

[4] Niels Böttcher. Current problems and future possibilities of procedural audio in computer games. *Journal of Gaming & Virtual Worlds*, 5(3):215–234, September 2013.

[5] Antoine Caillon and Philippe Esling. Rave: A variational autoencoder for fast and high-quality neural audio synthesis, 2021.

[6] Emre Çakir, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *CoRR*, abs/1702.06286, 2017.

[7] Jeffrey N. Chadwick and Doug L. James. Animating fire with sound. *ACM Trans. Graph.*, 30(4), jul 2011.

[8] Jeffrey N. Chadwick, Changxi Zheng, and Doug L. James. Precomputed acceleration noise for improved rigid-body sound. *ACM Trans. Graph.*, 31(4), jul 2012.

[9] Meiying Chen and Zhiyao Duan. Controlvc: Zero-shot voice conversion with time-varying controls on pitch and rhythm, 2022.

[10] Keunwoo Choi, Sangshin Oh, Minsung Kang, and Brian McFee. A proposal for foley sound synthesis challenge, 2022.

[11] Marco Comunità, Huy Phan, and Joshua D. Reiss. Neural synthesis of footsteps sound effects with generative adversarial networks, 2021.

[12] Jason Cramer, Ho-Hsiang Wu, Justin Salamon, and Juan Pablo Bello. Look, listen, and learn more: Design choices for deep audio embeddings. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3852–3856, 2019.

[13] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020.

[14] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis, 2019.

[15] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis, 2019.

[16] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. Ddsp: Differentiable digital signal processing, 2020.

[17] Andy Farnell. An introduction to procedural audio and its application in computer games, 2007.

[18] Andy Farnell. Marching onwards procedural synthetic footsteps for video games and animation. 2007.

[19] Andy Farnell. *Designing Sound*. The MIT Press. MIT Press, London, England, August 2010.

[20] Haytham M. Fayek. Speech processing for machine learning: Filter banks, mel-frequency cepstral coefficients (mfccs) and what's in-between, 2016.

[21] Yuan Gong, Yu-An Chung, and James R. Glass. AST: audio spectrogram transformer. *CoRR*, abs/2104.01778, 2021.

[22] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *J. Mach. Learn. Res.*, 13(null):723–773, mar 2012.

[23] Lamtharn (Hanoi) Hantrakul, Jesse Engel, Adam Roberts, and Chenjie Gu. Fast and flexible neural audio synthesis. 2019.

[24] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin W. Wilson. CNN architectures for large-scale audio classification. *CoRR*, abs/1609.09430, 2016.

[25] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017.

[26] andrew horner, james beauchamp, and lippold haken. methods for multiple wavetable synthesis of musical instrument tones. *journal of the audio engineering society*, 41(5):336–356, may 1993.

[27] Daniel Hug. How do you sound design? an exploratory investigation of sound design process visualizations. In *Proceedings of the 15th International Conference on Audio Mostly*, AM '20, page 114–121, New York, NY, USA, 2020. Association for Computing Machinery.

[28] Doug L. James. Physically based sound for computer animation and virtual environments. In *ACM SIGGRAPH 2016 Courses*, SIGGRAPH '16, New York, NY, USA, 2016. Association for Computing Machinery.

[29] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis, 2018.

[30] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet audio distance: A metric for evaluating music enhancement algorithms, 2018.

[31] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello. Crepe: A convolutional representation for pitch estimation, 2018.

[32] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis, 2020.

[33] Timothy R. Langlois and Doug L. James. Inverse-foley animation: Synchronizing rigid-body motions to sound. *ACM Trans. Graph.*, 33(4), jul 2014.

[34] Timothy R. Langlois, Changxi Zheng, and Doug L. James. Toward animating water with complex acoustic bubbles. *ACM Trans. Graph.*, 35(4), jul 2016.

[35] Mats Liljedahl and Johan Fagerlönn. Methods for sound design: A review and implications for research and practice. In *Proceedings of the 5th Audio Mostly Conference: A Conference on Interaction with Sound*, AM '10, New York, NY, USA, 2010. Association for Computing Machinery.

[36] Anton Lundberg. *Data-driven procedural audio : Procedural engine sounds using neural audio synthesis*. PhD thesis, 2020.

[37] Annamaria Mesaros, Toni Heittola, Tuomas Virtanen, and Mark D. Plumbley. Sound event detection: A tutorial. *IEEE Signal Processing Magazine*, 38(5):67–83, sep 2021.

[38] Gautam Mittal, Jesse Engel, Curtis Hawthorne, and Ian Simon. Symbolic music generation with diffusion models, 2021.

[39] james a. moorer. the synthesis of complex audio spectra by means of discrete summation formulas. *journal of the audio engineering society*, 24(9):717–727, november 1976.

[40] Max Morrison, Brian Tang, Gefei Tan, and Bryan Pardo. Reproducible subjective evaluation. In *ICLR Workshop on ML Evaluation Standards*, April 2022.

[41] J. Nistal, S. Lattner, and G. Richard. Drumgan: Synthesis of drum sounds with timbral feature conditioning using generative adversarial networks, 2022.

[42] Javier Nistal, Stefan Lattner, and Gaël Richard. Comparing representations for audio synthesis using generative adversarial networks, 2020.

[43] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio, 2016.

[44] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech 2: Fast and high-quality end-to-end text to speech, 2020.

[45] Curtis Roads. *Microsound*. The MIT Press, 2002.

[46] Matthias Robine, Robert Strandh, and Sylvain Marchand. Fast Additive Sound Synthesis Using Polynomials. In *Digital Audio Effects (DAFx06) Conference*, pages 181–186, Montréal, Canada, September 2006.

[47] Xavier Serra and Julius Smith. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):12–24, 1990.

[48] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions, 2017.

[49] Julius O. Smith. *Physical Audio Signal Processing*. `http:`http://ccrma.stanford.edu/ jos/pasp/`//ccrma.stanford.edu/~jos/-pasp/`, accessed ¡date¿. online book, 2010 edition.

[50] Matteo Torcoli, Thorsten Kastner, and Jurgen Herre. Objective measures of perceptual audio quality reviewed: An evaluation of their application domain dependence. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1530–1541, 2021.

[51] Matteo Torcoli, Thorsten Kastner, and Jürgen Herre. Objective measures of perceptual audio quality reviewed: An evaluation of their application domain dependence. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1530–1541, 2021.

[52] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards end-to-end speech synthesis, 2017.

[53] Yusong Wu, Ethan Manilow, Yi Deng, Rigel Swavely, Kyle Kastner, Tim Cooijmans, Aaron Courville, Cheng-Zhi Anna Huang, and Jesse Engel. Midi-ddsp: Detailed control of musical performance via hierarchical modeling, 2021.