Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Eivind Aksnes Rebnord

Generating Audio from Sample Libraries

With Variational Autoencoders Using Visualizations of Latent Space

Master's thesis in Computer Science
Supervisor: Björn Gambäck
June 2022

NTNU
Norwegian University of Science and Technology

Eivind Aksnes Rebnord

Generating Audio from Sample Libraries

With Variational Autoencoders Using Visualizations of Latent Space

Master’s thesis in Computer Science
Supervisor: Björn Gambäck
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Eivind Aksnes Rebnord

Generating Audio from Sample Libraries

With Variational Autoencoders Using Visualizations of Latent Space

Master's Thesis in Computer Science, June 2022
Supervisor: Björn Gambäck

Data and Artificial Intelligence Group
Department of Computer Science
Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology

Abstract

The sample library is a collection of digital sounds, known as samples, used by composers, performers, and producers of music. The process of selecting sounds for an audio production can be tedious and expensive, as it usually involves scrolling through large corpora of sounds and buying other collections of audio in the search for “the right” sample. Deep learning-based visualization tools have been developed to cope with the difficult sample selection process, clustering similar sounding samples together in two-dimensional maps. However, none of the existing visualization tools can generate new sounds if a user does not have the desired sample in their collection. Recent machine learning research has shown that generating audio in the waveform domain and learning timbre from training data is possible. This thesis bridges the gap between sample library visualization and generative audio modeling to create an interactive two-dimensional map of audio samples that lets the user meticulously generate samples with desired characteristics.

Considering this, a spectrogram-based system with a pipeline architecture was created. The generative model in the system is a Variational Autoencoder (VAE) with Inverse Autoregressive Flow which is responsible for learning and generating Mel spectrograms of samples. By using the VAE encoder to create latent embeddings for the spectrograms of the sounds in a sample library, these can be visualized in a two-dimensional map by performing dimensionality reduction with Uniform Manifold Approximation and Projection (UMAP). The system can generate a new latent embedding from any point on this map with an inverse UMAP transform and use the VAE decoder to output a new spectrogram. The Griffin-Lim algorithm is then used to reconstruct audio from the generated Mel spectrogram.

The system was evaluated by its ability to generate diverse samples and the quality of the generated audio. The system generated a wide variety of timbres from the samples in the training data but struggled with learning the differences within each class of samples. Overfitting the VAE to the training data resulted in a more local diversity and detailed spectrograms. Finer-detailed spectrograms resulted in slightly better audio quality for the reconstructed samples. However, the audio quality of most samples was considered to be poor. The reduction in quality was considered a result of spectrogram reconstruction limitations. The generated samples were limited to a fixed length of two seconds.

Examples of the generated samples can be found on https://tinyurl.com/4s4cwmaf, and it is strongly suggested to listen to these ahead of reading this report. The code repository can be accessed on https://github.com/EivindRebnord/SampleGenerator and includes instructions for using the system.

Sammendrag

Et sample-bibliotek er en samling av digitale lyder, kalt samples, og brukes av komponister, utøvende musikere og musikkprodusenter. Prosessen der lyder velges til en lydproduksjon kan være tidkrevende og vanskelig, siden det vanligvis innebærer mye leting og til og med kjøp i jakten på den “riktige” lyden. Nye visualiseringsteknikker for sample-bibliotek har blitt laget for å gjøre det lettere å finne lyder, blant annet ved å lage kart der lyder med de samme karateristikkene samles i kluster. Ingen av disse visualiseringsteknikkene er derimot kapable til å generere lyder som ikke finnes i lydbiblioteket fra før. Ny forskning innen maskinlæring har på den andre siden vist at det er mulig å generere lyd i bølgedomenet og å lære lydkarakteristikker fra treningsdata. Denne masteroppgaven kombinerer visualisering av sample-bibliotek med generativ lydmodellering ved å beskrive design og implementasjon av en interaktiv todimensjonal visualisering av samples som kan brukes til å generere lyder med spesifikke karakteristikker.

Denne masteroppgaven implementerer et spektrogram-basert system med en modulær arkitektur. Den generative modellen i systemet er en Variasjonell Autoencoder (VAE) med invers autoregressiv flyt, og er ansvarlig for å lære og generere Mel spektrogram av lyd. Ved å bruke VAE-enkoderen til å lage flerdimensjonale latentvektorer av spektrogrammene fra lydene i biblioteket kan lydene visualiseres i to dimensjoner ved å redusere dimensjonaliteten med Uniform Manifold Approksimasjon og Projeksjon (UMAP). Systemet lager nye latentvektorer fra brukerdefinerte punkt på den todimensjonale visualiseringen med en invers UMAP-transformasjon og bruker VAE-dekoderen til å generere spektrogram. Disse spektrogrammene kan deretter rekonstrueres tilbake til lyd ved hjelp av Griffin-Lim-algoritmen.

Systemet evalueres basert på hvor mange ulike samples som genereres og hvor god lydkvaliteten er. Systemet genererte et bredt spekter av lydkarakteristikker fra treningsdata, men hadde problemer med å lære forskjellen på ulikhetene internt i hver klasse. VOvertilpasning av VAE til treningsdata resulterte i bedre lokal variasjon og mer detaljerte spektrogram. Mer presise spektrogram resulterte i bedre lydkvalitet. Til tross for dette ble lydkvaliteten til mesteparten av de generte samplesene evaluert til å være lav. Reduksjonen i lydkvalitet skyldes begrensninger i prosessen som rekonstruerer lyd fra spektrogram. Lengden til de genererte samplesene var begrenset til to sekunder.

Eksempler av de genererte lydene finnes på linken https://tinyurl.com/4s4cwmaf og det anbefales på det sterkeste å lytte til disse lydene før denne rapporten leses. Koden til det implementerte systemet finnes på lenken https://github.com/EivindRebnord/SampleGenerator og inkluderer instruksjoner for trening og bruk av systemet.

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) at the Department of Computer Science (IDI) as a part of the requirements for the degree of Master of Science in Computer Science. The thesis was written under supervision of Professor Björn Gambäck. The thesis was written during the spring semester in 2022 and is based on a preliminary specialization project, *Generating music in the waveform domain*, written during the preceding semester. Sections that are adapted from the specialization project are indicated at the beginning of each chapter.

Special thanks go to Björn for supervising this thesis, and sharing his insight and feedback throughout the last year. I would like to thank Kyle McDonald, Manny Tan, Yotam Mann, and the Google Creative Lab for permission to use figure 1.1, and Sander Dieleman for permission to use figure 3.2 and 4.1. I would also like to thank Max Frenzel for his work on SampleVAE and his encouragement for other developers to contribute. Much of the code base developed in this thesis is based on his work. Moreover, I would like to thank the HPC group at NTNU, allowing the IDUN cluster (Själander et al., 2019) to be used for much of the experimentation conducted in this thesis. Finally, I would like to thank my family for encouraging me to do what I love, and Dina for always being there for me.

Eivind Aksnes Rebnord
Trondheim, 9th June 2022

Contents

List of Figures

List of Tables

Acronyms

API	Application Programming Interface. xi
CNN	Convolutional Neural Networks. xi
CQT	Constant-Q Transform. xi
DFT	Discrete Fourier Transform. xi
DR	Dimensionality Reduction. xi
FID	Frechet Inception Distance. xi
GAN	Generative Adversarial Networks. xi
GLA	Griffin-Lim Algorithm. xi
GUI	Graphical User Interface. xi
MOS	Mean Opinion Score. xi
NDB	Number of statistically Different Bins. xi
ODG	Objective Difference Grade. xi
PCA	Principal Component Analysis. xi
PEAQ	Perceptible Evaluation Metrics for Audio Quality. xi
PESQ	Perceptible Evaluation Metrics for Speech Quality. xi
RNN	Recurrent Neural Networks. xi
SDG	Subjective Difference Grade. xi
STFT	Short-Time Fourier Transform. xi
t-SNE	t-distributed Stochastic Neighbor Embedding. xi
UMAP	Uniform Manifold Approximation and Projection. xi
VAE	Variational Autoencoder. xi

1. Introduction

As music software such as digital audio workstations (DAW) has become more affordable, so has the accessibility to sound libraries. Free sound sharing platforms like Freesound (Font et al., 2013) and increasingly popular paid services like Splice and Noiiz give audio creators access to an enormous amount of samples, which can pile up on their hard drives in the number of thousands. Searching through a sample library can be an intimidating and time-consuming task, as samples are, in the best case, often organized with high-level descriptors such as "cow_bell", "kick" or "atmosphere_1". But how similar are sounds like "atmosphere_1" and "atmosphere_2"? Various software development, such as XO [1] and The Infinite Drum Machine [2], has been motivated by using audio similarity to navigate large sound corpus. These sample library visualization techniques use dimensionality reduction (DR) methods to project high-dimensional auditory features into 2D space for visualization, as shown in figure 1.1.

These tools are handy for visualizing sample libraries, but they cannot generate audio if the desired sound is not already in the existing library. Recent research, such as Latent Timbre Synthesis (Tatar et al., 2020), has used deep learning to generate audio in the waveform domain and has addressed the challenge of creating new sounds. However, without any visualization of audio characteristics, it is hard to know the kind of timbre generated.

1.1. Motivation

Deep generative models have never been combined with sample library visualizers before. The lack of such a system expresses the motivations for this thesis, which are to investigate how deep generative models in combination with visual representations of audio sample libraries can be used to generate entirely new sounds with specific audio characteristics. A use case of this would be that a user wants an open hi-hat sample that has the same characteristics but sounds a bit different from the selected sample marked with a green dot in figure 1.1. The user does not have any similar sample in their audio corpus but can click at any point close to the green dot and get an entirely new sample generated with the desired characteristics. Such a system would reduce the need for downloading and buying expensive sample packs and possibly reduce the number of samples needed in an audio corpus. The Infinite Drum Machine can ironically, given its name, only visualize a finite number of samples. In contrast, the system developed in this thesis will have an

[1] https://www.xlnaudio.com/products/xo
[2] https://experiments.withgoogle.com/drum-machine

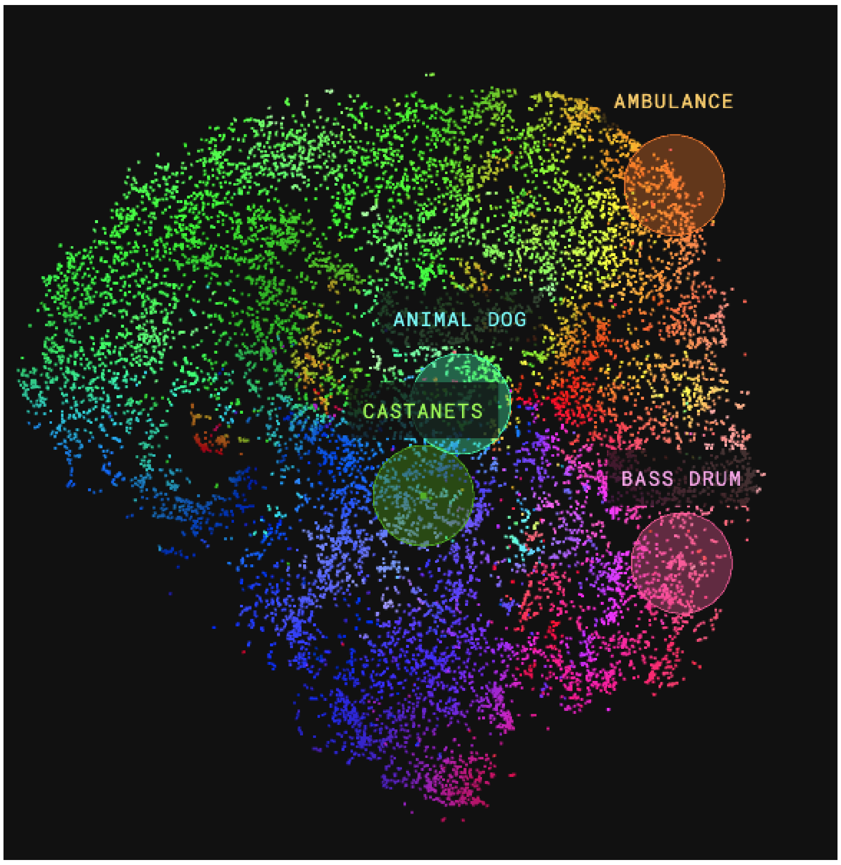


Figure 1.1.: The Infinite Drum Machine by Google is an experimental sound organizing tool using the DR method t-SNE for visualizing sample libraries in 2D space.

Source: McDonald et al. (2017) with permisssion.

infinite number of points in a two-dimensional visualization that can be used to generate new samples.

1.2. Goals and Research Questions

The driving force of this thesis is outlined by the goal, which is formulated in this section. The goal includes investigating state-of-the-art generative audio modeling, audio visualization, and reviewing evaluation methods for audio quality. The goal is more specifically formulated as follows:

Goal *Create a deep learning system with the capability of generating a diverse set of high-quality audio samples from an interactive visual representation of a sample library.*

To be able to evaluate different aspects of the goal, different conditions were formulated in table 1.1.

ID	**Conditions**
C1	Deep learning based models are the basis for the developed system.
C2	Audio in the waveform domain will be output by the system.
C3	The system should work for any type of audio sample regardless of the characteristics.
C4	The generated audio must be of high quality.
C5	The system should be able to generate a variety of samples that are not already in the sample library.
C6	The new samples should be generated from an interactive visual representation of the existing sample library.

Table 1.1.: Conditions that the system must fulfill throughout development.

In order to reach this goal, the following research questions will be answered:

Research question 1 *How well can audio be generated in the waveform domain with deep learning?*

R1 Understanding the model's domain is essential to understanding how well music and sound can be generated with deep learning. Music can have different representations when processed by the models. It is also essential to understand the work leading up to the current approaches and why they have become the leading methods today.

Research question 2 *How well can new sounds and timbres be generated from a visual representation of a sample library?*

R2 Understanding how to represent music's timbre is crucial for a deep generative model to learn such a high-level concept. Also, the representation of timbre should allow for a visualization that lets the user of the system navigate between different types of sounds in a sample library. Experimenting with different generative models and dimensionality reduction algorithms is crucial to answering this research question.

Research question 3 *How can the quality of generated samples be evaluated?*

R3 To answer this research question, various evaluation criteria must be defined based on factors that determine audio quality. Defining these criteria should propose evaluation metrics that, by themselves or as a part of an evaluation system, can indicate whether the audio produced by the developed system is of high audio quality or not. The evaluation methods needed to answer **R1** will be provided by answering this research question.

1.3. Contributions

The most notable contributions of this thesis are:

- The creation of a system generating diverse audio samples from an interactive visual representation of a sample library.
- An overview and evaluation of state-of-the-art deep generative models in the waveform domain.
- An understanding and evaluation of how different metrics can be used to measure the audio quality of generated samples.
- An evaluation of dimension reduction algorithms for sample library visualization.

1.4. Thesis Structure

The thesis will introduce and explain terminology and methods used in music and audio processing in chapter 2, as well as the deep learning foundations for generative models in chapter 3. These chapters provide the background theory for the systems described in the subsequent chapters. Chapter 4 covers the state-of-the-art for audio generation with deep learning and outlines the challenges of the different approaches. It also reviews systems attempting to learn the concept of timbre, which is described in chapter 2. The chapter also describes methods for evaluating audio quality. Chapter 5 describes the datasets that were used throughout the development of the system and explains why they were chosen. Chapter 6 gives an overview of the system architecture that was used when implementing the system, in addition to an overview of the software tools that were used during development. Chapter 7 describes the experiments conducted to make the final architectural decisions as well as testing and evaluating the performance of the system. Subsequently, chapter 8 evaluates the entire system based on the experimentation phase, its limitations, and related work. Lastly, chapter 9 concludes the thesis by outlining the work performed and recommended suggestions for how the work could be improved in future systems.

2. Music and Audio Processing

This thesis consists of techniques and terminology used across the fields of music and signal processing. This section will explain concepts within music and audio processing that are necessary for understanding mechanisms and decisions in the system developed in this thesis, as well as the systems described in the literature review in chapter 4. The following sections describe how audio and music can be represented by spectrograms and how spectrograms can be reconstructed back to raw audio waveform. The conditions **C2** and **C4** introduced in section 1.2 demand that the system outputs raw waveform and that the generated audio is of high quality, respectively. The representation and reconstruction methods described in this chapter were used throughout experimentation with the developed system to meet these conditions. The following sections are based on a preliminary specialization project (Rebnord, 2021); 2.1 and 2.2 with slight modifications.

2.1. Music Representations

To understand generative systems for music, it is essential to understand different representations of it and the challenges they represent. As an art form, music represents many relationships that can be treated mathematically, such as rhythm and harmony. Emotion, expectancy, and tension are other aspects that cannot be measured quantitatively. Music can refer to a printed notation, performance information, or the resulting sound. Raw waveforms and spectrograms are the only representations of music containing timbre information. Research question **R2** asked how well new sounds and timbres can be generated. Answering this research question implies that the system must use one of these representations. Therefore, the following subsections also include a dedicated subsection about the concept of timbre, which is unique for these representations.

2.1.1. Symbolic domain

The traditional way of studying music has been in the symbolic domain, where the representation can be a musical score, as shown in figure 2.1. Each note represents a distinct pitch, which corresponds to a specific frequency. Different tuning systems have different ways of mapping the notes to different frequencies, but the concept remains the same; high pitches have high frequencies and short wavelengths.

Even though sheet music is the preferred way of transcribing music, the piano roll has become a popular representation for music production. The original representation dates back to the early 19th century when holes were punched into a paper roll, indicating which note should be played. This representation was adapted into digital form and is

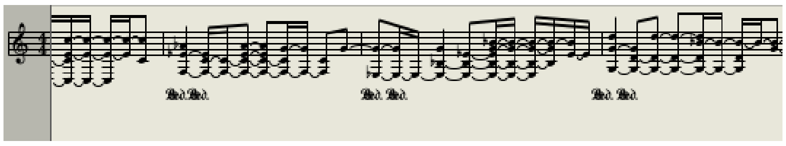


Figure 2.1.: Musical notation.

commonly used in music production. Figure 2.2 shows how a MIDI, Musical Instrument Digital Interface (Moog, 1986), file can be edited in a digital version of the piano roll in the digital audio workstation (DAW) Logic Pro. The piano roll can represent most aspects of piano performances with the three dimensions timing, pitch, and velocity. Velocity determines how hard a note is pressed and usually corresponds to the amplitude or loudness of a sound. This representation is, however, limiting the expressiveness of other instruments.

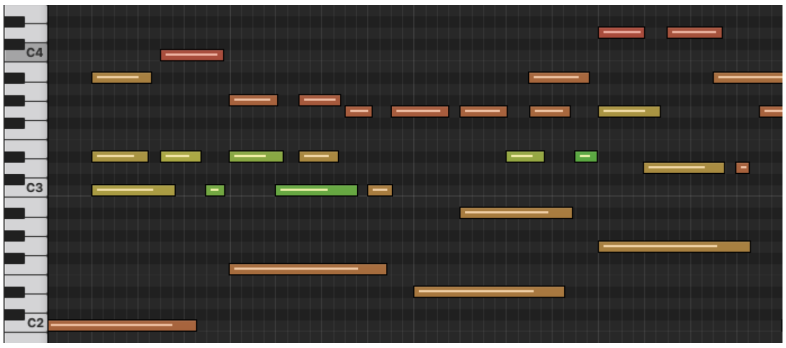


Figure 2.2.: MIDI file edited in the piano roll visualization in Logic Pro.

2.1.2. Timbre

Pitch, time and amplitude are aspects of music that can be represented in different ways. With timbre, what we are trying to represent is, according to Dannenberg (1998), still not defined. When aspects of timbre are understood in isolation, like spatial location and reverberation, they will be regarded as separate components, leaving timbre hard to comprehend. Composer (Schoenberg, 1911) states in his *Harmonilehre*: "I think that sound reveals itself by means of the timbre and the pitch is a dimension of the timbre. The timbre is, therefore, the whole, the pitch is part of this whole, or better, pitch is nothing more than timbre measured in just one dimension." Since the popularity of synthesizers and digital sound design emerged, Risset and Wessel (1982) stated that it is conceivable that proper timbral control might lead to entirely new musical architectures. Today, several deep learning-based systems can modify and transfer the timbre of sounds, utilizing latent representations to abstract the concept of timbre. Some of these systems are discussed in the literature review in section 4.2.

The complexity of music further increases when compositions include more than one instrument. The concept of density refers to the number of sound elements or instruments in a composition, where a dense or thick composition uses many instruments and vice versa. Range is another concept referring to the difference between the highest and lowest note used in a composition and is usually measured in octaves. For reference, the range of a full-size piano is 88 notes making the range 7 and 1/4 octaves since there are 12 notes in an octave in the western music scale. Benward and Saker (2009) introduce texture as a concept that describes the overall quality of the sound in a composition, where the most common are:

- Monophonic: A single melodic phrase is played by one or several instruments in the composition.
- Homophonic: One prevalent melody dominates the composition, based on chords moving together at the same speed.
- Polyphonic: Two or more independent melodies are played together simultaneously.

Homophonic textures dominate popular music, but the type of texture used in a single composition can also intentionally be changed to increase production value.

2.1.3. Digital Audio

It is essential to understand the digital representation of audio, as the constraints of the format determine how it should be processed. Analog continuous-time signals must be represented as discrete-time signals in the digital domain. Pulse-code-modulation (Bosi and Goldberg, 2002) is a method used to represent sampled analog signals digitally. The amplitude of the signal is sampled at uniform intervals, where each sample will be quantized to the range of digital steps, called bit depth, used by the format, which is usually 2^{16}. A bit depth of 16 means that the dynamic range of the signal will be theoretically limited to 96 dB, while the human hearing, in comparison, can perceive a dynamic range of 120 dB.

The highest frequency component that can be captured by the digital representation of an audio signal is given by the Nyquist Sampling Theorem (Cook, 2007), stating that a continuous-time signal can be sampled and perfectly reconstructed from its samples if the waveform is sampled over twice as fast as its highest frequency component. The highest frequency f_{max} that can be accurately represented with sample rate f_{s} is

$$f_{\text{max}} = f_{\text{s}}/2 \tag{2.1}$$

The highest frequency that can be sampled by CDs, which usually have a sampling rate of 44.1 kHz, is 22.05 kHz. In comparison, the human hearing range is 20 Hz - 20 kHz. Because of this, down-sampling is usually applied to the signal to reduce the amount of information needed to represent it so that complexity and processing time are reduced.

The formats used for digital audio are usually divided into lossless formats that retain the quality of an existing file and lossy formats that reduce the quality of an existing file by applying some form of lossy compression, which typically introduces artifacts. An example of a format typically used with audio is the lossless format WAV, which is also the format used for the audio in this project. Among the popular lossy formats, we find MP3 (Brandenburg and Popp, 2000), used in section 4.3 to show how audio quality degradation can be measured.

Digital audio has the highest information content of the different music representations discussed and is said to be a dense representation of music. Musical notation and MIDI contain a lower amount of information and are thus considered sparse representations of music. Timbre is one of the features that cannot be represented in the symbolic domain and is only present in denser representations like spectrograms and digital audio. Section 2.2 introduces digital signal processing and its application for analyzing and representing music as spectrograms. A comparison of the different densities of the representations can be seen in figure 2.3.

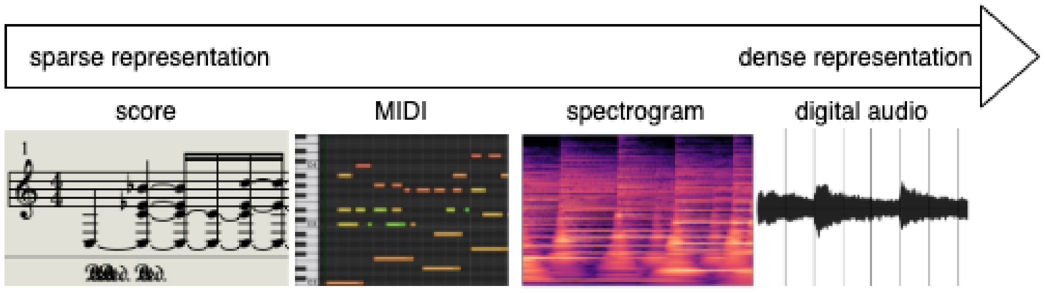


Figure 2.3.: Examples of sparse and dense representations of music.

2.2. Digital Signal Processing

Computational methods used for modification of audio signals are known as the field of digital signal processing (DSP). DSP methods can extract features from the audio, alter its representation and dimensionality, and reconstruct signals. Generating raw digital audio requires generating a number of data points equal to the sample rate per second. However, DSP methods can lower the temporal resolution and increase the information needed per timestep. This section explains different types of processing that are commonly used with audio signals and in generative systems for audio.

2.2.1. Fourier Transform

The Fourier transform is a tool for time-frequency analysis to measure the frequency domain representation of a signal. The discrete Fourier transform, DFT (Heideman et al., 1985), produces a finite trigonometric spectrum of a finite continuous signal divided into

N components and is formulated as follows:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j\omega_k n} \tag{2.2}$$

where the frequency of component k is $\omega_k = \frac{2\pi k}{N}$. The Short-time Fourier transform (STFT) performs the DFT across R timesteps, known as the hop-size. These chunks or frames of the signal are multiplied with a window function of length R samples that is only non-zero for a short period of time, ensuring that we perform the transform over a continuous signal. STFT results in a two-dimensional representation of the signal, where each frequency component is a complex number where the real part is the amplitude and the complex part is the phase, for each point in time m and frequency k.

$$STFT(k, m) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(n)w(n - mR)e^{-j\omega_k n} \tag{2.3}$$

Similarly, the inverse of the STFT can be computed simply by summing the inverse transforms over all the frequency components. Computing the inverse is possible since we have both the amplitude and phase information of the signal. When analyzing signals, it is preferred to visualize the intensity plot of the frequencies over time in the form of a spectrogram. A spectrogram can be made by looking at the magnitude components of the STFT, which is done by taking the absolute value of the complex matrix. This matrix of magnitudes is very difficult for humans to interpret, as seen in figure 2.4. Since the spectrograms only contain the magnitude information, it is impossible to invert back to the original signal.

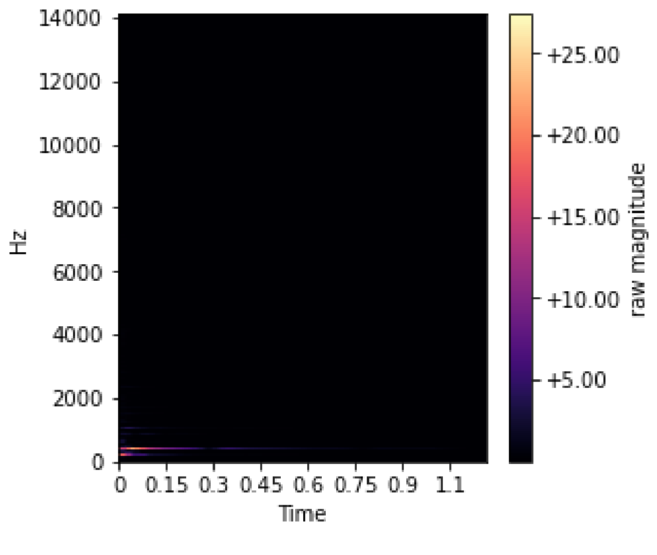
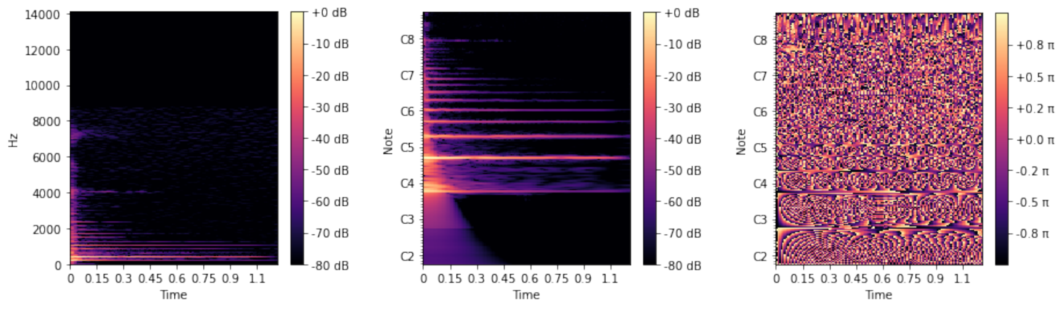


Figure 2.4.: STFT with linearly scaled magnitudes.



(a) Short-Time Fourier transform. (b) Constant Q-transform (c) Phase

Figure 2.5.: Different representations of the note A3 played on a piano.

To obtain the spectrogram, as seen in figure 2.5a, the intensity of the frequency component must be scaled in terms of decibels, which means that the magnitudes should be log-scaled with equation 2.4.

$$Y_{db} = 20 * log_{10}(y) \tag{2.4}$$

The phase of the signal is shown in figure 2.5c and illustrates how it is hard to obtain useful information because of the cyclic nature of the angles. The phase is, therefore, also usually discarded when making spectrograms.

2.2.2. Mel Spectrograms and Constant-Q Transform Spectrograms

Since the analysis window width of the STFT is the same for all frequencies, there will be a trade-off between resolving low-frequency information (large window) and having a good time resolution (small window) at the cost of low-frequency resolution. Multiple logarithmic frequency scales have been introduced to fit how the human ear perceives music. The Mel scale (Stevens et al., 1937), a subjective scale for the measurement of the psychological magnitude of pitch, was introduced in 1937 and differs from both the musical scale and the frequency scale, which are both objective. The Mel scale typically assigns a perceptual pitch of 1000 Mels to a 1000 Hz tone. Equation 2.5 shows a popular formula for converting f hertz into m Mels. Studies (Md Shahrin, 2017) has shown that Mel scaled STFT representations outperformed linear STFT representations in the classification of environmental sounds with deep learning.

$$m = 2595 \log_{10}(1 + \frac{f}{700}) \tag{2.5}$$

The Constant Q-transform, CQT (Brown, 1991), was derived by Brown as a means of creating another log-frequency resolution spectrogram, where the time resolution is a function of the frequency range of interest. Logarithmic scaling of the frequency lets the user choose a number of bins to represent each musical note instead of just a constant spacing between the frequency components, which is the case for the STFT. The frequency values are geometrically spaced, so that $\omega_k = 2^{\frac{k}{b}} \cdot \omega_0$ where k are the frequency bins and b is the geometric separation between each bin. This leads to a geometric spacing of $\delta_k = \omega_{k+1} - \omega_k = \omega_k(2^{\frac{1}{b}-1})$. In contrast to the STFT, the window length in the CQT-transform is a function of the frequency bin and is not a parameter that the user can choose.

$$CQT(k) = \frac{1}{N(k)} \sum_{n=0}^{N(k)-1} w(k,n)x(n)e^{-j\omega_k n} \tag{2.6}$$

For example, setting $b = 12$ means that there will be one bin per semi-tone for the equal-tempered scale, which is the scale that is used the most in popular and western music. The log-frequency resolution of the CQT transform makes it a convenient representation for music, as shown in figure 2.3. Since all octaves are scaled evenly, CQT has more information in the lower octaves and less information in the higher octaves than the STFT. A consequence of the scaling is that the CQT is not invertible since the transform matrix is not a square. Pseudo-inverse approaches (Fitzgerald et al., 2006) have been suggested, inverting C QT to DFT and then inverting the DFT back to the signal.

Another possible advantage of the CQT-transform is that it is possible to set w_0 as a frequency higher than 0, which means that it is possible to exclude the lowest octaves that would usually not contain much information for musical notes. In comparison, the STFT will always decompose the signal into $N/2$ frequency bins from 0 Hz up to the highest frequency in the signal. The CQT-transform spectrogram of the piano note A3 is shown in Figure 2.5b.

2.2.3. Audio Signal Reconstruction

Some reconstruction method is needed for time-frequency representations of audio, such as spectrograms, to be converted back to audio without losing signal information in the conversion process. Waveforms can be reconstructed by the inverse of either the CQT and STFT if frequency, amplitude, and phase data are available (Velasco et al., 2011). As the spectrograms used in audio processing usually involve the log-scaled STFT or CQT, the amplitudes of the frequency components will represent the decibel scale. The phase component of the signal is usually discarded, as only the magnitudes of the frequency amplitudes are shown in the spectrogram. Discarding the phase information means that such spectrograms cannot be reconstructed perfectly back into audio. Methods to create an approximate signal from magnitude spectrograms exist and are discussed in this section.

Griffin-Lim Algorithm With algorithm 1, missing phase information can be estimated from the magnitude (Griffin and Lim, 1984). The Griffin-Lim Algorithm (GLA) randomly guesses an initial phase $\angle c_0$ for each of the frequency components of the magnitude spectrogram. The inverse of this signal is computed so that a time series based on only the amplitude is obtained. STFT is then applied to this initial approximation and

Algorithm 1 Griffin-Lim algorithm (GLA)

```
Fix the initial phase $\angle c_0$                                  ▷ spectrogram $c$
Initialize $c_0 = s \cdot e^{i\angle c_0}$                          ▷ $s$: magnitude of signal
Iterate for $n = 1, 2, ...$
    $c_n = Pc_1(Pc_2(c_{n-1}))$            ▷ $Pc$: projection onto set of consistent spectrograms
Until convergence
$x^* = G^\dagger c_n$                                               ▷ $G^\dagger$ is inverse STFT
```

results in a complex matrix with some useful phase information. The magnitudes in this new matrix are then replaced with the original magnitudes from the spectrogram so that it now consists of unchanged amplitude information and some useful phase information. The algorithm then performs the inverse and STFT operations until it converges, which is done by minimizing the mean squared error between the target and predicted spectrogram.

Audio signal reconstruction utilizing Griffin-Lim is not desirable when high-quality audio is preferred due to its characteristic metallic-sounding artifacts when estimating phase. Figure 2.6 shows the resemblance of the original waveform and the waveform reconstructed with Griffin-Lim, compared to the inverse CQT transform without any phase information.

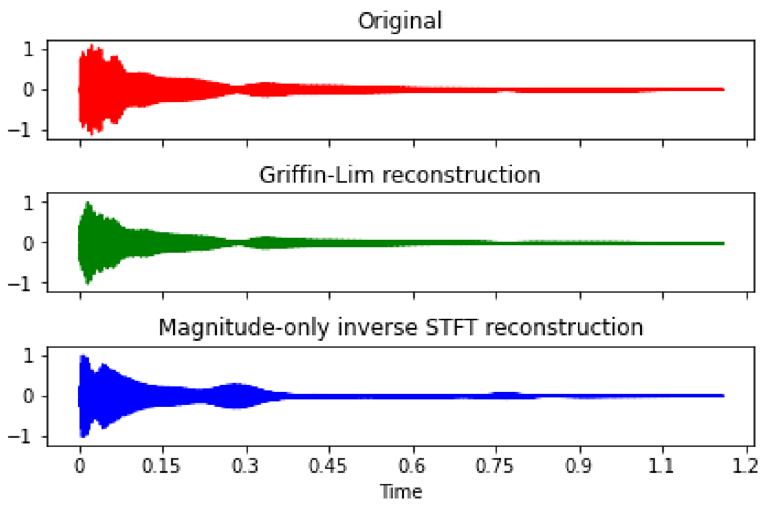
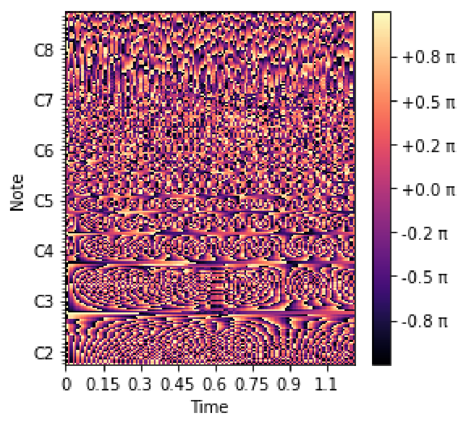
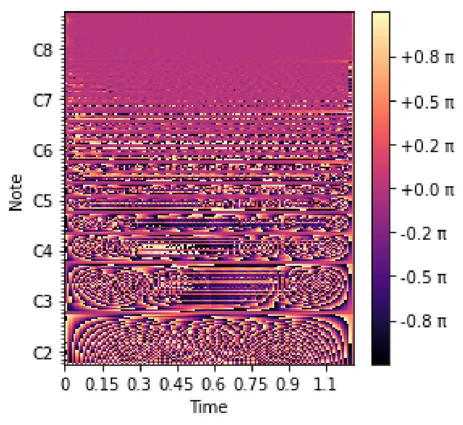


Figure 2.6.: Reconstruction of audio.

Figure 2.7 compares the phase information estimated by the GLA after converging versus after one iteration. Even though the algorithm will converge, it does not guarantee that the estimation will be similar to the original phase. This is because the algorithm does not have any prior knowledge of the original phase. The phase inconsistencies from the GLA can also be heard in the form of pre-echos preceding transient components, even though extensions (Dittmar and Müller, 2015) of the algorithm have improved the issue.



(a) Converged estimation.



(b) Estimated phase after one iteration of the GLA.

Figure 2.7.: Estimated phase with the GLA.

Reconstruction with Deep learning Spectrograms are a useful way of representing audio and have been the intermediate step for audio processing in many deep learning-related tasks. As STFT spectrogram is the only representation capable of a lossless reconstruction, the task of converting any time-frequency representation that is not STFT back to audio will suffer from reduced audio quality. Reconstruction will be possible, but due to the uninformed nature of Griffin-Lim, the result will not guarantee sufficient quality. The STFT transform is not preferred for audio processing because it is not as informative as its log-scaled counterpart. As discussed in Section 2.1, human hearing also perceives frequency logarithmically, and the CQT is known to be preferred when a sparse representation of the audio is needed (Cheuk et al., 2020).

The arguments for using other transforms than STFT outweigh the drawback of not being able to reconstruct it perfectly, and other methods have been proposed for reconstruction. Approaches include enhancing the time-frequency representation with phase information and using generative adversarial networks to reconstruct the waveform, while other approaches use vocoders to generate waveforms from the spectrograms. These deep learning-based models are detailed in the literature review in chapter 4.

3. Deep Learning

This chapter introduces concepts within deep learning employed by systems described in the subsequent chapter. Deep learning is, by definition, simply a term for artificial neural networks (ANNs) that have more than one hidden layer. Deeper networks with more layers can learn higher-level concepts. Some of the applications for deep learning are image recognition, computer vision, recommendation systems, natural language processing, and, more recently, synthesis of music and speech in the waveform domain. The very first approach to generating raw audio waveform was made by van den Oord et al. (2016). It was built on the concepts of a convolutional neural network (CNN), described in section 3.3. Since then, other generative deep learning models have been applied to music and audio, such as the Variational Autoencoder (VAE), which is detailed in subsection 3.4.1 and is used by the implemented system described in this thesis. Additionally, section 3.5 describes different dimensionality reduction (DR) techniques that were used for visualizing sample libraries, hence fulfilling condition **C6** of the goal. The following sections are based on a preliminary specialization project (Rebnord, 2021); 3.1 through 3.3, and 3.4 with slight modifications.

3.1. Feedforward Neural Networks

ANNs can be modified to obtain many properties, but the first variant that was introduced was the feedforward neural networks (McCulloch and Pitts, 1943). A network of nodes is set to computationally emulate the animal brain by learning how to predict an output based on input through training. Feedforward neural networks are characterized by the nodes forming a directed acyclic graph, meaning the computation is directed forward towards the output. The nodes have activation functions and are connected via weights to emulate the biological process of firing neurons. The weights are updated iteratively through training on large amounts of data so that the network learns to approximate the desired function. Figure 3.1 shows the structure of a feedforward neural network.

3.2. Recurrent Neural Networks

Recurrent neural networks (RNNs) are similar to feedforward neural networks but have loops in the hidden layers, feeding each neuron with information from the previous state, like working memory. RNNs are similar to chaining copies of feedforward neural networks together. The concept of recurrent networks was introduced by Rumelhart and McClelland (1987). Since RNNs can use information from previous states, they are

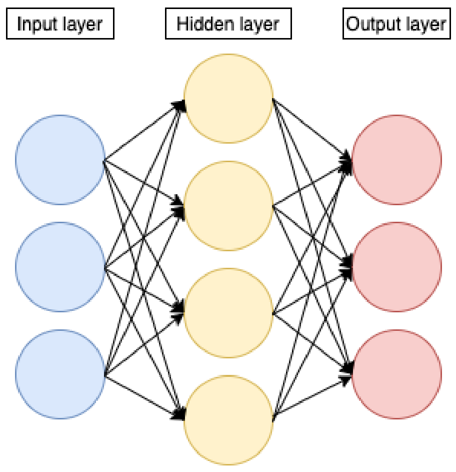


Figure 3.1.: A simple feedforward neural network with one hidden layer.

helpful for sequence prediction problems. Different input and output configurations yield different use cases. Examples of configurations are:

- One-to-many: An input observation is mapped to a sequence with multiple steps as output.
- Many-to-one: Sequences are mapped to a class or quantity prediction. An example is sound classification, where an input sequence of audio timesteps is classified into a class.
- Many-to-many: Sequences are classified into multiple categories or classes.

One of the challenges with RNNs is that they are hard to train. Factors like choosing the appropriate activation functions are essential for their performance. RNNs are also prone to suffering from vanishing or exploding gradients during training unless special mechanisms to regulate the information flow are implemented. The long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) architecture introduces such mechanisms by using gates that decide whether a cell should remember or forget information from the input states.

3.3. Convolutional Neural Networks

Convolutional neural networks (CNNs) are especially good at learning patterns in the input, making them the desired approach for image processing. CNNs use matrices of weights, known as kernels, to learn low-level features in the input, like, for example, edges. The kernel is multiplied with the input data and summed to create a single value. The kernel is then convolved across all input data, creating a smaller matrix known as a feature map. If the CNN is fed a large input image, it may be computationally expensive to perform convolutions. Strided convolutions can be performed to reduce the computational cost. Stride is the spatial distance between where the kernel is applied, and when it is larger than one, we call it strided convolution, resulting in a smaller output feature map the larger the stride is. The feature map typically goes through an activation function to eliminate or reduce negative values so that it is easier to train (Nair and Hinton, 2010). Stacking many convolutional layers makes the network learn higher-level features of the input. Such use of CNNs is employed by WaveNet detailed in section 4.1.1. Pooling layers are used to reduce dimensions further and thus the complexity and computation time. On the other hand, transparent convolutional layers can be used to upscale a latent representation. These layers are typically used to upscale the latent representation to a full-size image.

3.4. Generative Models

Given a dataset of examples $x \in X$ independently drawn from an underlying distribution $p_X(x)$, generative models are used to approximate this distribution $p_X(x)$, meaning they can be used to generate new samples that look like they could have been a part of the original dataset. Implicit generative models can only produce the new samples $x \sim p_X(x)$, but not infer the likelihood of that sample. In contrast, explicit generative models can infer the likelihood, at least to some extent.

It is possible to control what kind of samples we draw from a distribution to make generative models more practically useful. A conditional signal c can be introduced, fitting the model to a conditional distribution $p_X(x|c)$ rather than $p_X(x)$.

3.4.1. Likelihood Based Models

The foundation for a likelihood based model is parameterizing $p_X(x)$. The parameters θ are fit by maximizing the likelihood of the data under the model:

$$\mathcal{L}_\theta(x) = \sum_{x \in X} \log p_X(x \mid \theta) \quad \theta^* = \arg\max_\theta \mathcal{L}_\theta(x). \tag{3.1}$$

The direct parameterization of $p_X(x)$ makes it possible to infer the likelihood of any x, making these models explicit. Several variants of likelihood-based models have proven to be effective in generating music in the waveform domain.

Autoregressive Models Autoregressive models treat the examples $x \in X$ as sequences $\{x_i\}$. The distribution can then be factorized into a product of conditionals:

$$p_X(x) = \prod_i p\left(x_i \mid x_{<i}\right) \tag{3.2}$$

Since the model can be assumed to be stationary, the parameters can be the same for all the factors in the product. Due to the sequential nature of digital audio, autoregressive models are attractive since they capture the correlations between the different elements

x_i. They are fast for inferring $p_X(x)$ given x but slow to generate audio since sampling must be done sequentially.

Variational Autoencoders (VAEs) Kingma and Welling (2014) introduced the VAEs as a model that consists of two neural networks: an inference network $q(z|x)$ that learns to map examples x into a latent space probabilistically and a generative network $p(x|z)$ that learns the posterior distribution of the data conditioned on the latent distribution $p(z)$, also known as the latent space. Since the z's are not observable in the real world, they are called latent variables. The posterior distribution tries to explain how likely a latent variable z is given the input x. Since the model cannot infer $p_X(x)$ from x, it is trained by maximizing a lower bound on $p_X(x)$, called the Evidence Lower Bound (ELBO), which is given in equation 3.3. The intuitive explanation of ELBO is that the first part of the equation maximizes the log-likelihood, trying to make the generated sample more correlated to the latent variable and thus more deterministic. The second part of ELBO minimizes the divergence between the prior and posterior distribution, making the distributions more similar while preserving the latent space. The second part of the equation is also known as the Kullback-Leibler divergence (Kullback and Leibler, 1951).

$$\ln p_X(x) \geq \mathbb{E}_{q(z|x)}[\ln p(x \mid z)] - D_{KL}[q(z \mid x) \| p(z)] = likelihood - KL \tag{3.3}$$

Variational Autoencoders are popular deep learning architectures for generative modeling. One of the advantages of VAEs over GANs is that they will not suffer from mode collapse. Mode collapse can occur in GANs when the generator learns to fool the discriminator by generating only a few classes from the training data. In contrast, the latent space of VAEs is built from all the samples in the training data. The objective function of VAEs wires it to focus on all input classes. If the decoder somehow would only focus on outputting one class, this would lead to inadequate recovery of the other classes, leading to a huge loss. Many approaches using VAEs in music generation apply it to learn a higher-level concept, such as timbre. Due to the VAE learning to average over examples, it is a preferred method when diversity is desired in the generated output.

Normalizing flows In most VAEs, the prior and posterior distributions are modeled as Gaussians since their derivative is easy and efficient to compute. Many real-world distributions are more complex than Gaussians, and the concept of normalizing flows was introduced to better approximate more flexible and complex distributions. Normalizing flow layers shape a simple distribution into a more robust distribution by applying a series of transforms in each flow layer. With k transformations, the latent vector z_0 is sampled from a simple distribution and goes through a series of k invertible transformations $f(z_i)$ until it is shaped into a more complex distributed z_k, which is then passed to the decoder.

Inverse Autoregressive Flow, IAF (Kingma et al., 2016), extends the concept of normalizing flows to scale well to high-dimensional space. By using inverse autoregressive transformations that can be parallelized, the computational cost is reduced without losing the flexibility of the resulting distribution. The mathematical details explaining IAF are omitted from this section, and the reader is advised to inspect the original paper for more information. The VAE architecture employed by the developed system in this thesis utilizes inverse autoregressive flow layers to better approximate the posterior distribution.

3.4.2. Adversarial models

Goodfellow et al. (2014) introduced the generative adversarial network (GAN) architecture. GAN consists of two neural networks, a generator G, and a discriminator D, that compete against each other in an adversarial game. G has the task of generating a plausible sample given the dataset, while D has the task of classifying whether a sample is from the dataset or not. In essence, G is trying to fool D, and D is trying to determine if this is the case.

G is given a noise vector z to generate a diverse range of samples. Therefore, a generated sample can be represented as $G(z)$. The objective of D is to maximize

$$log(D(x)) + log(1 - D(G(z)) \tag{3.4}$$

which means maximizing $log(D(x))$ when x comes from the dataset, and minimize $log(D(G(z))$ when a sample is generated by G. The objective of G is to minimize $log(1 - D(G(z)))$, which means fooling the discriminator. The immediate advantage of the generator loss is that it is not dependent on any ground truth. The generator loss will indicate whether the generated example could have been a part of the training dataset or not without comparing it to the training examples. The independent behavior of the generator and the discriminator does not allow the probability of the generated examples to be computed. One of the disadvantages of GANs is that it is not possible to infer the likelihood of an example, making it an implicit model.

Mode-seeking and Mode-covering Behavior When choosing a generative model, it is crucial to determine if the behavior of the model should be *mode-covering* or *mode-seeking*. Compromises are made when a model cannot capture the variability of the data. If all examples should be likely under a model, a mode-covering behavior would lead to the model overgeneralizing and interpolating between examples in ways that might not be meaningful. If covering these examples is not a requirement, a mode-seeking behavior could focus the probability mass on a subset of the examples. The two behaviors are shown in figure 3.2. Since likelihood-based models maximize the joint likelihood of the data, they have a mode-covering behavior. Adversarial models are usually mode-seeking.

The density of the domain representation usually determines if the generative approach should be likelihood-based or adversarial-based. If a sparse representation of high-level concepts is used, then diversity and a certain degree of "creativity" should be expected from the model, meaning a mode-covering behavior is desirable. A dense representation like waveform samples or image pixels will usually favor realism in the generated output of a model rather than diversity. Blurriness is typically an artifact of mode-covering

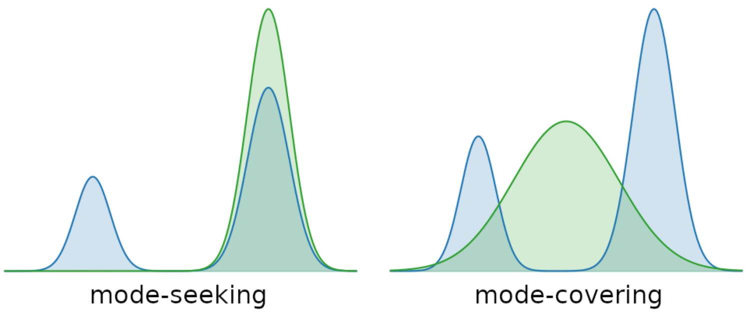


Figure 3.2.: Illustration of mode-seeking and mode-covering behavior in model fitting. The blue density represents the data distribution, while the green density is the single normal distribution representing the generative model.

Source: Lee et al. (2019) with permission.

approaches in image generation, as it tends to average over examples. Generating random patterns is thus another weakness of mode-covering models.

3.5. Dimensionality Reduction Techniques

Visualization of high-dimensional datasets is challenging. The natural structure of the data can be plotted when it is in two or three dimensions, but high-dimensional plotting is not as intuitive. This section describes the basics of dimensionality reduction (DR) techniques frequently used to visualize high-dimensional space. Considering different methods for DR is essential as there tend to be trade-off qualities that one technique will have over another. The computational properties of the following techniques were also important for choosing the most suitable method for the developed system.

3.5.1. PCA

Principal Component Analysis (PCA), as defined by Hotelling (1933), is the orthonormal axes onto which the variance under projection is maximal for a set of d-dimensional data vectors. In other words, the objective of PCA is to find the axes of a dataset that have the highest variance. The most significant advantage of PCA is also one of its most prominent limitations and is the fact that PCA only defines a linear projection of the data, not considering any underlying nonlinearities in the data. Its global linearity yields efficient computation, and one of its implementations (Tipping and Bishop, 1999) was experimented with for visualization in section 7.2.2. Page 61 shows how the latent embeddings of one of the sample libraries used for experimentation spread out across the axes in a PCA plot. Even though PCA suffers from not emphasizing local clusters in the data, the plot shows the data's dominating global structures, which can be interpreted as the sample length on the x-axis and frequency on the y-axis.

3.5.2. t-SNE

Manifold learning techniques can be regarded as an attempt to generalize linear frameworks like PCA to be responsive to nonlinearities in the data. One of the popular manifold learning techniques is t-distributed Stochastic Neighbor Embedding, t-SNE (van der Maaten and Hinton, 2008). The algorithm creates a similarity matrix for each data point in the high-dimensional space and a similarity matrix for each map point in the reduced space, considering Gaussian distribution for the data points and t-Student (hence t-SNE) distribution for the map points, respectively. The similarity matrices are then optimized by reducing the Kullback-Leibler (KL) divergence between the two distributions. Due to the heavier tail of the t-Student distribution over the Gaussian distribution, for a given similarity of two data points, the two corresponding map points have to be much further apart to minimize the KL-divergence between the distributions. Typical hyperparameters for t-SNE are perplexity and number of iterations. Perplexity defines the size of the neighborhood, while the number of iterations sets the number of optimization steps, respectively.

t-SNE can produce effective visualizations of complex datasets, uncover hidden structures and expose natural clusters and nonlinearities in the data. There are, however, several known disadvantages with t-SNE:

- On million-sample datasets, t-SNE is computationally expensive, with execution times of several hours compared to minutes for PCA.
- To preserve a global structure, the points must be initialized with PCA, requiring even more computation.
- The algorithm is stochastic and can result in different embeddings for the same data.

Despite some drawbacks, t-SNE is still a widely used dimensionality reduction technique. It is experimented with as one of the methods for visualizing the latent space of the developed system in section 7.2.2.

3.5.3. UMAP

Uniform Manifold Approximation and Projection, UMAP (McInnes et al., 2018), is a manifold learning technique that is competitive with t-SNE for DR and debatably has better global structure preservation with a much lower computational cost. The UMAP algorithm creates a graph with the shape of the high-dimensional data points. Comparable to the similarity matrix of the t-SNE data points, the edge weights of a UMAP data point correspond to the similarity measure to the other points. While t-SNE optimizes the difference between two similarity matrices, UMAP compresses its graph of the high-dimensional space into a lower dimension. The reduction in computational cost with UMAP is due to its rough estimation of the high dimensional graph instead of measuring every point.

There are mainly four parameters impacting how UMAP will compute the embeddings

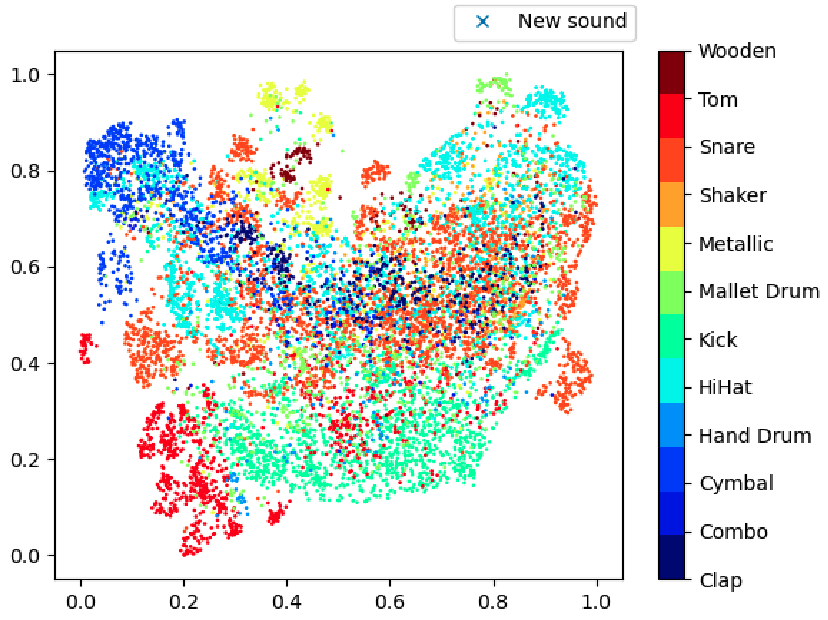


Figure 3.3.: An example of UMAP used for reducing the high dimensional latent embeddings of a sample library.

- The number of neighbors controls the balance of local versus global structure in the dataset. A low value for the number of neighbors parameter forces the algorithm to focus on locality, while high values will result in larger neighborhoods.
- Minimum distance determines how close each data point can be in the embedding space. Low values result in tight clusters and vice versa.
- Number of components specify the number of dimensions of the reduced dimension space.
- Metric decides how the distances are computed for the input data.

A visualization of the 64-dimensional latent vectors reduced to two-dimensional embedding space with UMAP (number of neighbors=100, distance=0.8) from the sample library used with the developed system is shown in figure 3.3. These were the preferred visualization parameters from the experimentation conducted in section 7.2.2.

The UMAP library also supports inverse transformation, generating high-dimensional data points from a specific point in the low-dimensional embedding space. Therefore, it is possible to fit a dataset, such as the high-dimensional latent variables of a sample library, to a UMAP transform and then generate new high-dimensional samples from the embedding space. The fact that invertible UMAP methods are available was a deciding factor for including it in the developed system.

4. Literature Review

This chapter presents the three-part literature review conducted to answer the research questions in this thesis. Section 4.1 describes state-of-the-art within generating music in the waveform domain and builds the foundation to answer research question **R1**, asking how well audio can be generated in the waveform domain. Section 4.2 outlines recent attempts at learning the concept of timbre with generative models and is essential for answering how well new sounds and timbres can be generated from a sample library, as asked by research question (**R2**). Lastly, section 4.3 describes frequently used evaluation techniques for audio quality and is the basis for answering how the audio quality of generated samples can be evaluated (**R3**). The performed literature review is the foundation from which the architectural decisions detailed in the following chapter have been made.

4.1. Generative Models in the Waveform Domain

The two most common techniques of generating audio with deep learning are discussed in this section. Section 4.1.1 details the very first generative approach of direct waveform modeling, without any use of intermediate representations like spectrograms. Since the initial attempts to generate music by directly modeling the waveform, there have been suggested various other approaches for neural synthesis of music. Section 4.1.2 describes state-of-the-art within neural audio synthesis methods conditioned on spectrograms. The following subsections are based on a preliminary specialization project (Rebnord, 2021); 4.1.1 with slight modifications and 4.1.2 with modifications.

4.1.1. Direct Waveform Modeling

Directly modeling waveforms means generating audio as a discrete quantization of the signal amplitude through time, as explained in section 2.1.3. The output will typically be in an audio file format, such as WAV (*.wav*). Using "raw audio" as the conditioning representation was initially motivated by rapid progress in the field of text-to-speech synthesis, and these techniques got a head start compared to the methods described in subsection 4.1.2.

Likelihood-based Synthesis WaveNet is an autoregressive model introduced in 2016 by van den Oord et al., initially targeted at generating speech. It appeared together with SampleRNN (Mehri et al., 2017) as the first models to directly generate waveforms. Before this, it was not seriously considered to model long-term correlations in thousands of timesteps of audio sequences. To deal with the long-time dependencies, WaveNet introduced dilated causal convolutions to extend the receptive field of the model. Causal convolutions ensure that the model cannot violate the ordering in which the data is modeled, which means that the model will not use future timesteps to infer the next. A dilated convolution is a convolution where the filter is applied over an area larger than its length by skipping input values with a certain step. The process is equivalent to convolution with a larger filter, where the original filter is dilated with zeros. Using a dilated filter keeps the output the same size as the input, even though it practically calculates the strided convolution. More layers, larger filters, and greater dilation factors can be incorporated to increase the receptive field further. Figure 4.1 shows an example of a stack of dilated convolutional layers.

WaveNet's audio modeling capabilities were tested on multi-speaker speech generation, text-to-speech (TTS), and music audio modeling. Applied to speech generation, WaveNet improved the state-of-the-art, reducing the gap with human performance by over 50%. It was able to capture characteristics of several different voices and the acoustics and recording quality of the data from which it was trained. With a receptive field size of 240 milliseconds, it could also outperform the previous baseline models. Even though it was able to synthesize natural-sounding speech, the relatively short receptive field was not enough to learn which words to stress in a sentence, but further conditioning on linguistic features solved this problem. The short receptive field was also the limiting factor when applying the model to the music waveform domain. The authors trained the model on 200 hours of different genres of music audio. Subjective evaluations of the generated results found that further enlarging the receptive field to several seconds was not enough to enforce long-range consistency across the generated music. By further conditioning the model on a set of tags specifying genre and instruments, it was possible to control the output.

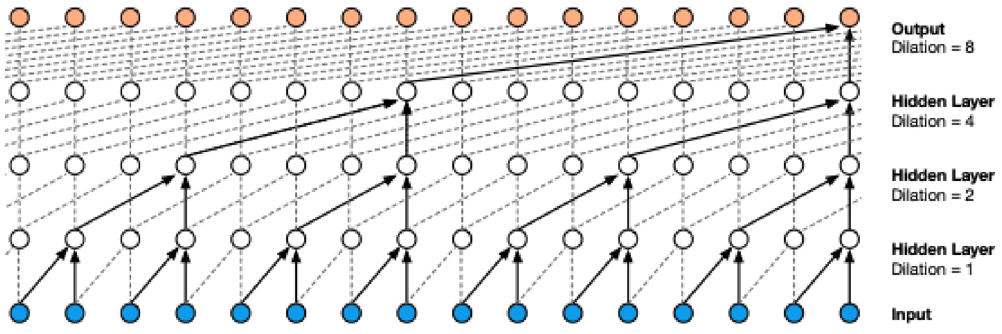


Figure 4.1.: Visualization of a stack of dilated casual convolutional layers.

Source: van den Oord et al. (2016) with permission.

SampleRNN has a different architecture than WaveNet, as it utilizes a hierarchical stack of recurrent neural networks operating at different temporal resolutions. The lowest module is autoregressive and processes individual samples, while each module above increases the timescale, reducing the temporal resolution. The higher modules condition the lower, eventually taking the longer-term dependencies into account on the output. Even though SampleRNN performs excellent on TTS, it still suffers from the same issues as WaveNet with capturing coherent long-term musical structures.

Adversarial Synthesis Directly modeling raw audio with GANs was first approached by WaveGAN (Donahue et al., 2019). By training a GAN on single-word speech recordings, bird vocalizations, individual drum hits, and short excerpts of piano music. Samples with a duration of one second were generated with global coherence, and the model was suggested for sound effect generation. The architecture was based on deep convolutional GAN, DCGAN (Radford et al., 2016), but modified by the authors to output 128x128 pixel images.

One of the challenges of using GANs to model audio waveforms directly is the intrinsic differences between audio and images. The principal components of images capture edge characteristics, intensity and gradient, while audio forms periodic shapes. For the DCGAN to exhibit periodicity, it needs a receptive field that is large enough to capture a single cycle. Using a sample rate of only 16 kHz, the authors applied one-dimensional kernels to capture the periodicity.

The authors also made SpecGAN, generating STFT spectrograms with DCGAN and using Griffin-Lim to invert the spectrograms back to audio, with which WaveGAN was compared. Only qualitative human judgment was used for evaluating the audio generated by the models. The human judges were set to identify the audio examples presented with a label from 1 to 9 and rate the quality of the audio from 1 (poor) to 5 (best). WaveGAN achieved a mean accuracy of 0.58 for the labels, but better quality than SpecGAN, most likely due to the shortcomings of Griffin-Lim. The authors explained that the slightly better accuracy with SpecGAN could be that it might better capture the underlying variance.

4.1.2. Spectrogram Reconstruction

The following section describes approaches to generating audio by using intermediate representations in the form of time-frequency spectrograms. The representation of audio with magnitude spectrograms has seen wide usage in discriminative audio models. The field of automatic music transcription (AMT) has typically utilized various spectrograms to analyze the onset and offset of notes for monophonic and polyphonic music (Cheuk et al., 2020). Spectrograms are convenient as they are less dense than raw waveform audio. Using spectrograms opens up possibilities for more efficient models with fewer parameters. Section 2.2.3 specified how phase reconstruction with Griffin Lim degrades audio quality since it is impossible to restore lost phase information perfectly. Numerous deep learning methods for reconstructing audio from spectrograms have therefore been suggested.

Conditioning Generative Models on Spectrograms Shen et al. introduced Tacotron 2 in 2018, suggesting a system for speech synthesis directly from text. The system introduced a many-to-many RNN that mapped character embeddings to Mel spectrograms and a modified WaveNet model (subsection 4.1.1 acting as a voice coder (vocoder) to synthesize waveforms directly from those spectrograms. A vocoder is a term for devices breaking the spectrum of a signal (typically the human voice) into several sub-bands so that the resulting parametric representation easily can be manipulated in different ways (Cook, 2007). Instead of applying traditional phase reconstruction algorithms to the spectrograms, they modified the WaveNet to be conditioned on frames of the spectrograms, where each frame had a hop size of 12.5 milliseconds. The conditioned WaveNet model was an independent component in Tacotron 2, meaning that it did not rely on other components in the system. The training was done simply by feeding the model pairs of spectrograms and the original audio. The system was intended for TTS and was trained on 24.6 hours of speech from a female speaker.

Using a WaveNet vocoder to generate audio from spectrograms, the authors reported impressive results when evaluating it with human ratings. The system was evaluated with a mean opinion score (MOS) of 4.53, comparable to 4.58 for professionally recorded speech. MOS as an evaluation method is discussed in section 4.3. The authors also mention factors that could lead to an inflated MOS since each human evaluation was done on the examples independently instead of comparing each example. The evaluation set also contained similar patterns and words as the training set, leading to a MOS that would be better if the system was trained sentences generated from random words. Regardless of possibly inflated MOS, Tacotron 2 was compared to its predecessor, Tacotron (Wang et al., 2017), with Griffin-Lim, a WaveNet trained on linguistic features and parametric and concatenative models that were previously used in production at Google, and outperformed all of them.

MelGAN was introduced by Kumar et al. (2019) as an approach to directly model audio in the waveform domain by inverting magnitude Mel spectrograms. Due to the popularity of spectrogram representations of music but the difficulties of inverting spectrograms without reducing audio quality, MelGAN appeared in 2019 as a less computationally expensive option to invert Mel spectrograms back to audio. The generator architecture takes an input spectrogram and upsamples this with a stack of layers that use transposed convolutions and dilated convolutions, resulting in a sequence of raw waveforms on the output. Three discriminators dissect the generated audio at different scales of downsampling so that the generator more correctly models the high-frequency content. It is difficult to estimate how MelGAN would perform when inverting music spectrograms rather than speech spectrograms, as the original paper did not evaluate the model when it was used with music. However, the authors state that the MelGAN can replace the modules converting spectrograms to audio in existing systems and produce audio of "decent quality". When applied to Mel spectrograms of speech, the model was evaluated with mean opinion scores (MOS) against several other methods, where WaveNet and Griffin-Lim were some of them. On a scale from 1 (worst) to 5 (best), MelGAN performed significantly poorer than the other approaches, except Griffin Lim, which introduces characteristic artifacts that reduce quality.

Alternative Spectrogram Representations An ideal alternative approach to modeling waveforms directly would be to use invertible spectrograms. Section 2.2.3 explained how phase information is necessary for a spectrogram to be perfectly invertible, but also how hard it is to model phase due to its cyclic nature. In spectrograms, the phase information will essentially be random as the magnitude tends towards 0, as shown in figure 2.7. An important aspect of phase is, therefore, that the absolute value is not very informative but that the relative phase differences over time matter perceptually.

Engel et al. (2019) introduced a new spectrogram representation by unwrapping the phase, essentially adding 2π whenever a phase discontinuity is crossed, causing the phase to grow linearly with time. The difference of the derivatives of the unwrapped phase would then show the relative phase differences across each of the frequency components in the spectrogram, defining what the authors called the "instantaneous frequency" (IF). The relative phase information can be encoded into the spectrograms as the color on a rainbow color map, where the magnitude determines the brightness, naming the resulting representation “rainbowgrams”. Figure 4.2 shows the IF encoded in the rainbowgram of a piano chord progression.

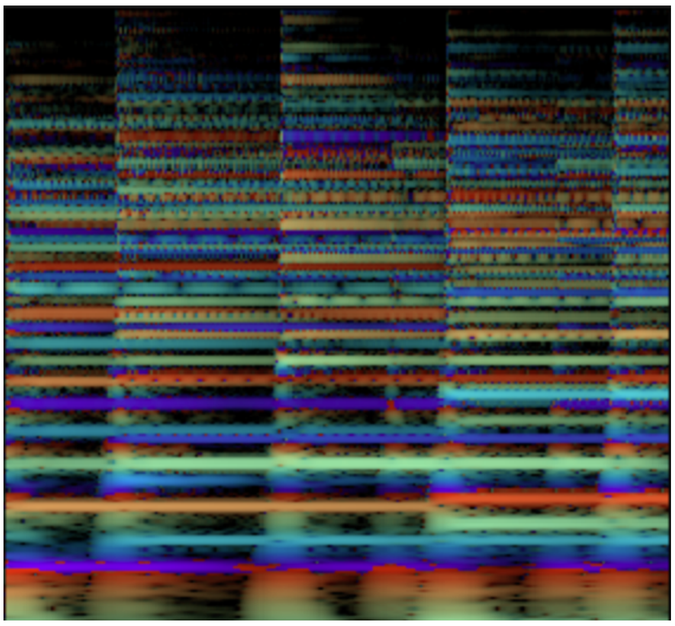


Figure 4.2.: Rainbowgram of a piano chord progression.

GANSynth produces audio by generating log-magnitude spectrograms and phases directly and applying an inverse transform instead of directly generating a waveform with strided convolutions. Based on recent success within progressive training methods (Karras et al., 2017) for image generation with GANs, the authors generated audio spectra of notes. The NSynth dataset, a collection of 305,979 musical notes from 1,006 different instruments, was used to train their model. Instead of just training the GAN on log-scaled magnitude spectrograms, the authors also conditioned the model on pitch information by appending a one-hot pitch encoding to the latent vector. They implemented an auxiliary classification loss for the discriminator, predicting the pitch label so that the generator would use the pitch information. They achieved independent control of pitch and timbre by applying pitch information to the latent vector.

The performance of GANSynth was evaluated on variants of STFT and Mel spectrogram representations of the dataset with 512 and 1024 frequency bins, where all of these representations were tried by either separately generating the absolute phase or generating the instantaneous frequency. When asking participants in a survey what sample they thought had better audio quality, the reconstructed notes with a Mel spectrogram with instantaneous frequency and 1024 bins were rated almost as high as the original audio samples. Qualitative analysis of the generated audio was also phase-coherent, meaning that the harmonics modulo the fundamental frequency align. If the harmonics of a signal do not align, it will sound blurry. The authors chose WaveGAN (Donahue et al., 2019) as one of the baselines for comparison, and it showed many phase irregularities. Due to the sequential dependencies of autoregressive models, GANSynth could generate 4-second long samples 54,000 times faster than WaveNet. GANSynth is limited to only generating single monophonic notes despite providing efficient content generation with high accuracy.

Deep Griffin-Lim Iteration Another approach to phase reconstruction is to enhance the existing Griffin-Lim Algorithm (GLA). Deep Griffin-Lim Iteration (DeGLI) was introduced by Masuyama et al. (2019) by combining the original GLA with a deep neural network (DNN). It is motivated by using information from intermediate representations of an STFT spectrogram as its phase is reconstructed with the GLA. The GLA tries to reduce the difference between the amplitude replaced spectrogram and the closest consistent spectrogram, but the GLA does not use this information. The authors recognized this as an optimization problem and tried to reduce this difference with a DNN. The suggested model was only applied to speech audio and evaluated with Perceptual Evaluation of Speech Quality, PESQ (Rix et al., 2001), a method for speech quality assessment of telephone networks and codecs, in which it scored significantly better than the reconstructions with only the GLA. However, this approach has not been applied to reconstruct spectrograms of music despite its promising results. DeGLI was investigated further in the audio quality experiment in section 7.1.

4.2. Learning Timbre

This section describes state-of-the-art deep learning-based systems that generate music, samples, and audio and facilitate architectures for learning the concept of timbre. Most of the following systems are based on the generative models in the previous section and tend to be modular. Modularity seems typical for generative models in the waveform domain that try to learn higher-level concepts. It allows a dedicated module to convert raw audio into a less dense representation that the system can process before another module converts it to audio. The following subsections are based on a preliminary specialization project in (Rebnord, 2021); 4.2.1 and 4.2.2 with modifications.

4.2.1. Jukebox

One of the more comprehensive likelihood-based systems for generating music in the waveform domain is Jukebox Dhariwal et al. (2020). Jukebox is a WaveNet-based system that generates music with singing in the raw audio domain. It captures the context of the raw audio with three vector-quantized variational autoencoders (VQ-VAEs). VQ-VAEs are similar to the standard VAEs described in section 3.4.1, except that the latent representation is quantized. The Jukebox architecture is based on the latent representation of three VQ-VAEs running at three different temporal resolutions to capture different features of the input audio. The content generation itself is based on a model that generates sequences of latent vectors. The latent content vector is fed to the VQ-VAE decoder, transforming it into waveform audio. Each of the three VQ-VAE decoder layers employs a WaveNet-style network with dilated convolutions to generate the raw audio waveform. The system can generate diverse songs with coherence up to multiple minutes, including singing when the model is conditioned on lyrics and was trained for several months. The system was trained with a spectral loss calculated over multiple STFT parameters. The authors reported artifacts in the form of missing and blurry high frequencies, even with a model size of 5 billion parameters. The authors evaluated Jukebox on coherence, musicality, diversity, and novelty. No audio quality metrics were used. Subjectively listening to examples[1] provided by the authors gave the impression of an average quality similar to MP3-files with a bit rate of less than 100 kbps.

4.2.2. Differentiable Digital Signal Processing

Differentiable Digital Signal Processing, DDSP (Engel et al., 2020), is a very different approach for generating audio in the waveform domain and is based on digital signal processing and synthesizers with deep learning. A synthesizer typically uses oscillators to continuously output waveforms based on parameters such as pitch and amplitude. Engel et al. utilize the strong inductive bias of controlling the parameters of a synthesizer with neural networks and avoid the challenges with neural audio synthesis by implementing a fully differentiable synthesizer and audio effect architecture.

The DDSP modules are made differentiable by approximating the time-varying values of synthesizer parameters with neural networks. The modules in the DDSP library that generate audio are the additive synthesizer and the filtered noise synthesizer. A neural network outputs values for the additive synthesizer module's fundamental frequency, amplitude, and harmonic distribution. For the filtered noise module, a neural network predicts a vector used as the frequency-domain transfer function for a uniform stream of noise. The outputs of these modules are typically combined to reconstruct the audio from

[1] https://openai.com/blog/jukebox/

an instrument. When used to reconstruct the audio of a violin, the additive synthesizer generates the harmonic content, while the filtered noise synthesizer typically mimics the bowing noises. Since the primary objective of the DDSP modules is to minimize reconstruction loss, the absolute difference between the spectrograms of the original audio and the spectrograms of the generated audio is backpropagated through the neural networks of the modules.

DDSP can accurately model various instruments with high fidelity, without comprehensive autoregression or costly computations. Since the neural networks only predict the parameters of the synthesizer modules instead of each audio sample, which is the case for WaveNets, the authors could run the model at a rate of only 250 Hz and still achieve realistic-sounding results. This improvement in computational efficiency allowed a DDSP model to train for only a couple of hours on a V100 GPU on 13 minutes of solo violin performance and still resynthesize violin performances very accurately. Audio quality metrics were not used, but subjectively listening to the provided audio examples confirmed that the resynthesized audio sounded like an authentic violin player, with bowing noises, reverberation, and audio quality similar to the original dataset. The DDSP authors specify in a comment[2] that the fundamental frequency parameter is a constraint of the model by construction, and therefore that the model is not appropriate for modeling non-periodic signals such as experimental electronic audio samples.

4.2.3. Latent Timbre Synthesis

Latent Timbre Synthesis, LTS (Tatar et al., 2020), is a deep learning-based audio synthesis method using Variational Autoencoders. LTS is intended to be used by composers of electronic music better to explore the timbre space of their sample libraries. The authors implemented a graphical interface for LTS, providing options for interpolating and extrapolating between arbitrary audio excerpts rather than just timbres of instruments, the latter being constricted to only pitched sounds.

The implemented system processes audio recordings as CQT-spectrograms and converts the output spectrograms to audio with the Griffin-Lim Algorithm (GLA). The authors built the LTS system on VAEs, as they allow for encoding audio frames to a latent space and generate new audio frames from latent vectors. By combining multiple latent vectors, the system can interpolate and extrapolate between timbres. LTS also allows for flexible changing of the duration of the generated sounds, rather than only generating audio excerpts with a fixed duration.

The authors utilized two different VAEs in series, where the first one was inspired by previous work (Generative Timbre Spaces) that synthesized conventional musical instrument timbres. The first VAE has a latent space of 256 dimensions, but the authors added another VAE to generate a latent space of 8 dimensions while trying to maintain reconstruction quality. The authors did not evaluate the audio quality but mentioned that the artifacts from reconstructing the CQT spectrograms would result in artifacts regardless of the deep model generating "perfect" spectrograms.

[2] https://openreview.net/forum?id=B1x1ma4tDr

4.2.4. SampleVAE

SampleVAE (Frenzel, 2019) is a deep learning-based tool developed by Max Frenzel intended to generate samples for music producers and sound designers. The tool consists of a convolutional Variational Autoencoder (VAE) with normalizing flow layers designed to be trained on user sample libraries. Since the model processes audio as Mel spectrograms, it reconstructs the spectrograms to audio with the Griffin-Lim Algorithm (GLA). The sample libraries used to train the model are thus converted into a dataset consisting of fixed length spectrograms.

SampleVAE was intended for sample generation, classification, and similarity search. After training on a dataset of spectrograms, SampleVAE is used as a generative model by sampling the latent space and decoding the latent vector. Generating samples with SampleVAE means that the decoded point from the latent space will resemble a realistic example of a spectrogram from the training examples. SampleVAE allows for randomly sampling the latent space, re-generating variations of a sound by modifying its embedding, and averaging over multiple embeddings to combine the characteristics of multiple sounds.

The VAE architecture of the model is specified in figure 4.3. The author stated that adding inverse autoregressive flow (IAF) layers qualitatively led to better overall results. The idea behind IAF is based on normalizing flows (Rezende and Mohamed, 2015). It is introduced to shape flexible posterior distributions by iterating over a series of invertible transformations known as flow layers. These layers are simplified as the flow module in figure 4.3, transforming the simple distribution z_0 to the more complex distribution z_k. The concept of normalizing flows is explained in more detail in the chapter about deep learning on-page 18.

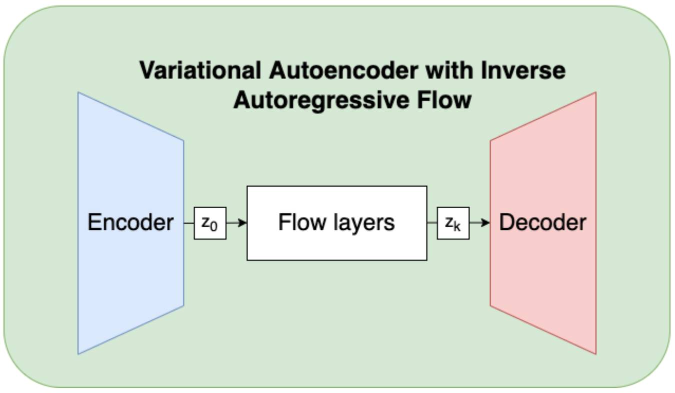


Figure 4.3.: The SampleVAE architecture utilizes a Variational Autoencoder with Inverse Autoregressive Flow layers.

The author did not conduct a rigorous evaluation of SampleVAE, and decisions for architecture and hyperparameters are based on the intuition of the author. Furthermore, the author suggests several improvement aspects for the model, including trying different architectures for audio reconstruction, such as WaveNets. Using two dimensions for the latent space to generate sounds from a 2D grid was suggested as an idea for further work and is similar to the system developed in this thesis. Compared to other state-of-the-art systems, SampleVAE seems to be a robust framework for generating audio from a sample library with great potential for experimentation.

4.3. Evaluating Audio Quality

There exists no single, consistent definition of quality, even though quality has an understood meaning when applied to audio. Objective measurement techniques exist for assessing audio quality but seek to identify the difference in quality between a gold standard reference signal and a test signal that has undergone some destructive process, like the effect of compression codecs. Other aspects than possible distortions from down-sampling are essential for determining the quality of a produced piece of music. This section outlines evaluation metrics and approaches to audio quality evaluation.

4.3.1. Mean Opinion Scores

Mean Opinion Scores (MOS) is defined by the International Telecommunication Union (ITU) as “The value on a predefined scale that a subject assigns to his opinion of the performance of the telephone transmission system used either for conversation or for listening to spoken material”.

ITU presents MOS as a standardized system for evaluating telecommunication services and recommends specifying what kind of frequency band the test evaluates and what kind of audio material. Even though MOS is used mainly for subjective opinion, it is also used for scores originating from objective models or scores that are estimated, and ITU encourages specification of whether a listening test is subjective (S), objective (O), or estimated (E). In addition, ITU states that factors like level, application, listening device, and environment also impact the absolute MOS value and should be reported. They state that general audio signals, such as music or mixed speech and music, should not be used with objective models but can be used with subjective models of MOS. The typical five-point Likert scale used with MOS is shown in table 4.1.

Value	**Opinion**
1	Bad
2	Poor
3	Fair
4	Good
5	Excellent

Table 4.1.: Mean Opinion Score scale.

4.3.2. Perceptible Evaluation Metrics for Audio Quality (PEAQ)

An alternative system for evaluating perceived audio quality was suggested by the ITU (Thiede et al., 2000) to simulate the audio quality rankings produced by humans. The intention was to develop a system to respond similarly to a subjective listening test comparing a reference and test audio signal. The test audio signal usually has undergone perceptual coding or degradation in quality. Examples of perceptual coding are encoding the test audio signal with a lower bit rate or sample rate, as discussed in section 2.1.3. While PEAQ is intended for use in the broadcasting domain, it is also helpful as an aid to subjective assessment. Employing such a system is motivated by saving time and resources spent executing a listening test and calculating the mean opinion score (MOS). The duration of the test signal should be about the same as if it was used in a listening test, which is typically around 10 to 20 seconds. Such a listening test is, in general, based on the Subjective Difference Grade (SDG), which is defined as:

$$SDG = Grade_{\text{Test Signal}} - Grade_{ReferenceSignal} \tag{4.1}$$

where the values range from 0 to -4, where 0 means the difference is imperceptible and -4 means that the impairment is considered very annoying. PEAQ defines the Objective Difference Grade (ODG) in terms of the "judgment of impairment," and its corresponding categories are shown in Table 4.2.

ODG	Judgement of impairment
0	Imperceptible
−1	Perceptible but annoying
−2	Slightly annoying
−3	Annoying
−4	Very annoying

Table 4.2.: Audio reconstruction comparison.

As a reference to illustrate the sensitivity of the PEAQ metric, Salovarda et al. (2005) conducted tests of various codecs from a reference audio clip in WAV format (16 bit, 44.1 kHz). They compared several degrees of audio degradation by reducing the bit rates for the different codecs. Table 4.3 shows the resulting ODG values for the MP3 format. PEAQ is used throughout the audio quality experimentation in section 7.1 for measuring audio degradation between reference and generated audio.

4.3.3. Audio Quality in Music Productions

Determining the quality of recorded music is a highly disputed subject. Wilson and Fazenda (2016) observed a difference in the perception of audio quality and liking of the music from commercial CDs by measuring the subjective and objective reactions of

Bit rate	**Cut-off frequency**	**ODG**	**File size**
32 kbps	5 kHz	-3.67	56kB
64 kbps	11 kHz	-3.46	112kB
128 kbps	15 kHz	-1.08	223kB
160 kbps	16 kHz	-0.47	276kB
256 kbps	17 kHz	-0.01	445kB
320 kbps	19.5 kHz	0.04	556kB

Table 4.3.: ODG values and file size for MP3 format on most common bit rates.

test subjects. The authors found that audio quality was perceived by characteristics of signal features related to perceived loudness and dynamic range compression. The sonic attributes that affected the subjective quality ratings were timbre, space, and defects. What decided the subjects' liking of the music was their familiarity with stimuli. Listener expertise did not affect the result. It was also observed that the perceived quality of popular music may have decreased over recent years, but like ratings were unaffected.

Another study, Wilson and Fazenda (2013) found that the listeners' perception of audio quality was linked to their emotional reaction to the sample. The emotional link meant high-quality ratings were awarded to happy-sounding recordings, and low-quality ratings were awarded to angry-sounding recordings. They further found that spectral features, such as higher bandwidth, meaning audio with significant low- and high-frequency energy, would result in the sample being perceived as high quality. Amplitude features, such as having enough dynamic range, and spatial features, such as the perceived width, were linked to the subjects' quality assessment. Rhythmic features, such as a low tempo, were associated with higher quality due to the space between notes to hear instrument details and evaluate spaciousness.

5. Datasets

This section discusses the data that has been considered and used for experimentation in chapter 7. For music producers, sound designers, and audio creators, the sample library will be one of the factors that decide artistic style (Epworth-Sawyer et al., 2019). The immediate advantage of this is that large amounts of data are available for the potential users of the developed system. For example, my personal sample library consists of roughly 100k sounds due to years of collecting sounds from other libraries, recording my own samples, and processing existing ones. After all, the intended use of the system is for the creator to extend their sonic palette. However, the drawback is that there are almost no benchmark datasets that can be used for reproducible experimentation, as audio creators possess different audio corpora. Even though the complete sample libraries of audio creators are unique, and will be unique, there exist sub-libraries that are more common than others.

5.1. Native Instruments Battery 4 Drums library

One of the sub-libraries that is common among audio creators is the Battery 4 Drums library produced by Native Instruments [1]. Native Instruments has been producing audio technology for more than 20 years, including many sample libraries. Their audio software is promoted towards amateurs and professionals, and their monthly user base counts more than 1.5 million. They also claim that eight out of ten songs on the Billboard Top 100 feature Native sounds from Native Instruments. One of the popular sample libraries made by Native Instruments is the drum sample library that comes bundled with their drum machine Battery 4[2]. This library comes with 10275 different sounds across 12 different categories of drum hits, as shown in figure 5.1. Due to the popularity of the sample library, a decision was made to use it as a dataset for training the generative model in the experiment detailed in section 7.3. Another advantage of using this library is the diversity of styles within each type of drum hit. Even though categories like cymbals, hi-hats, and shakers are mostly non-harmonic and noise-based, categories like tom, hand drum, and wooden usually have harmonic features. For this reason, constraining the system to be trained on this dataset is an essential step in fulfilling condition **C3**, which demands that the system should work for any audio sample regardless of the characteristics.

All samples have a sample rate of 44.1 kHz. The sample length varies from 8 ms to 40 seconds with a mean length of 1.214 seconds. The dynamic range varies from -158.4 dB to -1.7 dB measured in full scale (FS) decibels, where 0 dB is the maximum, and

[1] https://www.native-instruments.com/en/company/about-us/

[2] https://www.native-instruments.com/en/products/komplete/drums/battery-4/

Drum Hit	n	Mean Length
Clap	423	0.737 s
Combo	83	1.090 s
Cymbal	767	4.063 s
Hand Drum	142	0.852 s
HiHat	1934	0.811 s
Kick	1594	0.846 s
Mallet Drum	644	1.051 s
Metallic	643	1.450 s
Shaker	377	0.293 s
Snare	2596	0.859 s
Tom	918	1.962 s
Wooden	154	1.095 s

Table 5.1.: Drum hits in the Native Instruments Battery 4 Drums library varies in numbers and lengths.

anything softer is negative. The acoustic properties of the sounds vary across the dataset and within each sample category. This variation is evident when visualizing the different sample categories in section 7.2.2, as there are several overlapping areas. This behavior is expected, as long hi-hat sounds will have similar high-frequency dominant auditory characteristics as a cymbal. Having a large variety of sounds in the dataset makes it very appropriate for training the generative model of the developed system. The intention is for the generative system to produce as many different sounds as possible, and the training data should facilitate this. Therefore, the diversity was the main reason for choosing the Battery 4 Drums library as the training dataset for the generative part of the developed system.

Native Instruments claims that a valid license for their sample libraries allows for commercial and non-commercial use in audio productions. Any usage of their samples for creating sound libraries or sample-based instruments is strictly prohibited. Individual samples cannot be distributed. Because of this, the datasets used with the developed system only include magnitude spectrograms data and not any copyrighted raw waveform.

5.2. NSynth dataset

Just like datasets such as MNIST, CIFAR and ImageNet are typical baseline datasets in the image domain, the NSynth dataset (Engel et al., 2017) was established as one of the first benchmark datasets for audio in the waveform domain. The dataset consists of

$305,979$ notes from $1,006$ different instruments. Each note is 4 seconds long and has a sample rate of 16 kHz. Each instrument has an average of 65.4 notes with 4.75 different velocities per pitch. The dataset is also labeled based on a combination of human and algorithmic evaluations determining each instrument's source, family, and qualities.

Even though the NSynth dataset is not intended to be used for audio production, it still represents a benchmark dataset for measuring audio quality and has been used by several projects discussed in section 3.4.1. The primary motivation for experimentation with this dataset is its establishment as a point of reference in other audio research. It also has a large number of different timbres available. The dataset also includes a variety of sound characteristics such as percussive and transient sounds, fast decaying sounds, and sustained sounds. The NSynth dataset is also monophonic, just like a standard sample library consisting of short sounds, making it appropriate for experimentation with the audio quality of the system in section 7.1. Every note in the dataset has a pitch label and a fundamental frequency, which is not necessarily the case for a typical experimental electronic sample library with all kinds of dissonant and atonal sounds. The NSynth dataset was used only for the audio quality experiment and not the other experiments since it is less diverse than the Battery 4 Drums library.

6. Architecture

This chapter details the fundamental architecture used to develop the system, including the tools and frameworks used during experimentation and development. The architecture serves as one of the main structures for reaching the project goal. The system design is similar to the modular approach adopted by Tatar et al. (2020) in the literature review in that the developed system includes an audio to spectrogram module, spectrogram processing module, and a spectrogram reconstruction module. The developed system also includes a sub-module for data visualization and interactivity. All of these modules are independent processes and contribute to specific conditions formulated in section 1.2 in order to reach the project goal.

The information flow of the system is shown in figure 6.1. An immediate advantage of this modular approach is that alternatives for each module can be compared and changed without affecting any of the other modules. The following section describes only the architecture of the implemented system but mentions the options that were considered for the different modules. Implementation details are specified in chapter 7. Section 6.5 gives an overview and justification of the software frameworks and tools that were used.

6.1. Audio to Spectrogram Module

The audio to spectrogram module is the first module in the system's pipeline and is responsible for transforming any audio input into a spectrogram representation. Condition **C3** demands the system to work for any type of audio sample regardless of the characteristics. As any audio input can be converted to a spectrogram, condition **C3** is fulfilled. There are many advantages of converting an audio input to a spectrogram and using this as an intermediate representation. From the literature review, there are many takeaways:

- Spectrogram-based models are not necessarily dependent on having a fundamental frequency constraint or pitched instrument on the input. In experimental electronic music and sound design, working only with sounds having fundamental frequencies can be limiting. The authors behind Latent Timbre Synthesis (Tatar et al., 2020) justified their use of spectrograms given the fact that they wanted to be able to generate any sound.
- There are few attempts at learning timbre with raw waveform, and the few existing approaches have shown to be computationally expensive. The first WaveNet architecture (Engel et al., 2017) attempted to learn the timbre of single musical

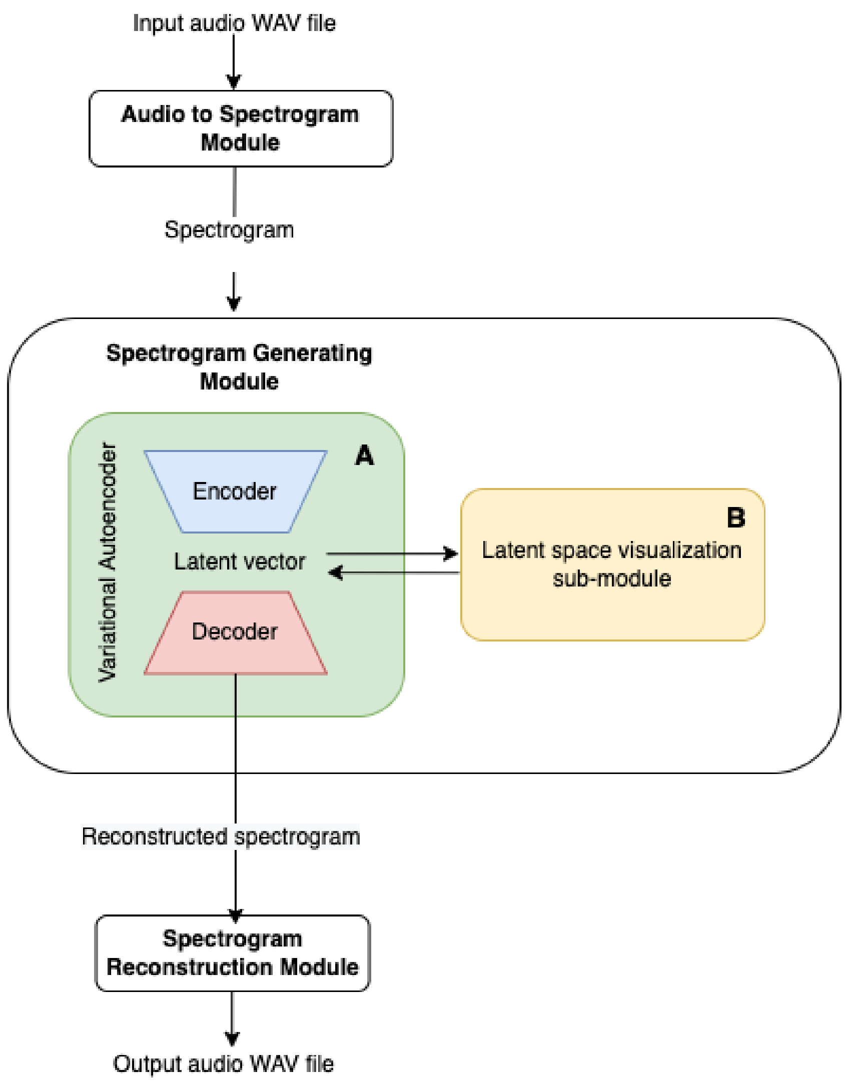

Input audio WAV file
Audio to Spectrogram Module
Spectrogram
Spectrogram Generating Module
Variational Autoencoder
A
Encoder
Latent vector
Decoder
B
Latent space visualization sub-module
Reconstructed spectrogram
Spectrogram Reconstruction Module
Output audio WAV file

Figure 6.1.: The system architecture.

notes with the NSynth dataset and trained for 250k iterations, using multiple days. Jukebox (Dhariwal et al., 2020) used a different approach with raw waveform and entire songs, using months to train. Techniques using spectrograms as intermediate representations from the literature review were less computationally expensive compared to models using raw waveform.

- Since there are many ways of reconstructing a spectrogram back to audio, the system will be able to output audio. Condition **C2** demanded the system to output audio in the waveform domain and will thus be satisfied.

Using spectrograms as the intermediate representation affects all the succeeding modules in the system pipeline. The entire spectrogram reconstruction module is, in fact, a result of this choice, and options for this were taken into consideration when deciding on the architecture. Experimentation described in section 7.1 found that Mel spectrograms reconstructed with Griffin-Lim yielded the highest quality audio results.

6.2. Spectrogram Generating Module

For the spectrogram generating system to fulfill the project goal, it has to be trained on a sample library, learn representations for the different timbres, and generate new timbres based on this representation space. The input for the system is the spectrogram from the preceding module, while the output is a generated spectrogram going into the spectrogram reconstruction module. The spectrogram generating system should be split into two parts. The first part is the generative system, and the second part is the system for visualizing the latent space of the generative system. The following sections elaborate on the architecture of the two submodules, while specific implementation details are given in the subsequent chapter.

6.2.1. Generative Sub-module

As a part of meeting the project goal, condition **C1** was formulated in the introduction demanding that the developed system must employ a deep learning-based generative model. The system must learn the timbre of the input spectrograms from an entire sample library dataset so that it can be used to generate new spectrograms from any point in the latent space.

The literature review in section 4.2 discusses several deep learning-based systems for audio that generate new sounds by interpolating over or transferring timbre. Even though Jukebox seems capable of learning the timbres in a sample library, its large number of parameters appears excessive for being used with relatively short excerpts of monophonic samples. NSynth, GANSynth, TimbreTron, and DDSP are all using a neural network architecture conditioned on pitch since they generate sounds from various pitched instruments. A sample library typically has a variety of sounds that are atonal or noisy and do not rely on any fundamental pitch. Latent Timbre Synthesis (LTS) (Tatar et al., 2020) was able to generate a variety of sounds by training variational autoencoders on spectrograms without any pitch information. A similar variational autoencoder architecture was used by SampleVAE (Frenzel, 2019) to generate new sounds based on a sample library. SampleVAE implements a more complex VAE architecture by using flow layers to shape the latent space accurately and was adopted as the generative system in the pipeline. The decision to use this architecture was based on preliminary experimentation with a pre-trained model provided by the author. The model was trained on 60,000 audio samples converted to Mel spectrograms and generated a variety of sounds by converting the spectrograms to audio with the Griffin-Lim Algorithm. A more detailed figure of the adopted VAE architecture is shown on page 54.

6.2.2. Latent Space Visualization Sub-module

Condition **C6** was formulated as a requirement for reaching the project goal in the section 1.2 and demands that samples should be generated from an interactive visual representation of the existing sample library. A dimension reduction algorithm is needed to transform the high dimensional latent representations into two dimensions for visualization. Also, the user should be able to select a point in this 2D visualization and invert this back into a high-dimensional latent representation. The new latent representation will then go through the decoder part of the VAE, which outputs a new spectrogram. The requirement for a point in the 2D visualization to be converted to a latent vector implies that the sub-module needs to employ an invertible dimension reduction algorithm.

The majority of the reviewed literature in section 4.2 uses different dimension reduction algorithms for visualizing sample libraries such as PCA, t-SNE, UMAP, or combinations of them. Visualization is subjective, and there is not necessarily one technique better than the other. UMAP was chosen as the dimension reduction algorithm for the system as it is invertible, computationally efficient, and provides flexible embeddings with few parameters. The experiments in section 7.2.2 compare visualization results of dimension reduction with PCA, t-SNE, and UMAP.

6.3. Spectrogram Reconstruction Module

Condition **C2** demands that the system outputs raw audio waveform. As the spectrogram generating module outputs spectrograms, a reconstruction module is needed for the system to fulfill condition **C2**. Condition **C4** demands that the system outputs audio of high quality and is also very much dependent on the spectrogram reconstruction module, as there is significant variance in audio quality between the different approaches. Section 7.1 also shows how the spectrogram resolution affects the audio quality. The literature review mentioned several methods for inverting spectrograms back to audio, including the Griffin-Lim algorithm (Griffin and Lim, 1984), WaveNet (van den Oord et al., 2016), and Deep Griffin-Lim Iteration for better phase reconstruction of magnitude spectrograms. Based on the results from the experimentation in section 7.1, Mel spectrograms with Griffin-Lim as the phase reconstruction method was chosen as the module for reconstructing audio.

6.4. Information Flow

This section describes how information flows through the system during training and sample generation. The complete system with all its modules is shown in figure 6.2. The modules used during the system training are different from the modules used during sample generation. Therefore, the following subsections will clarify how the information flow differs depending on the usage of the system.

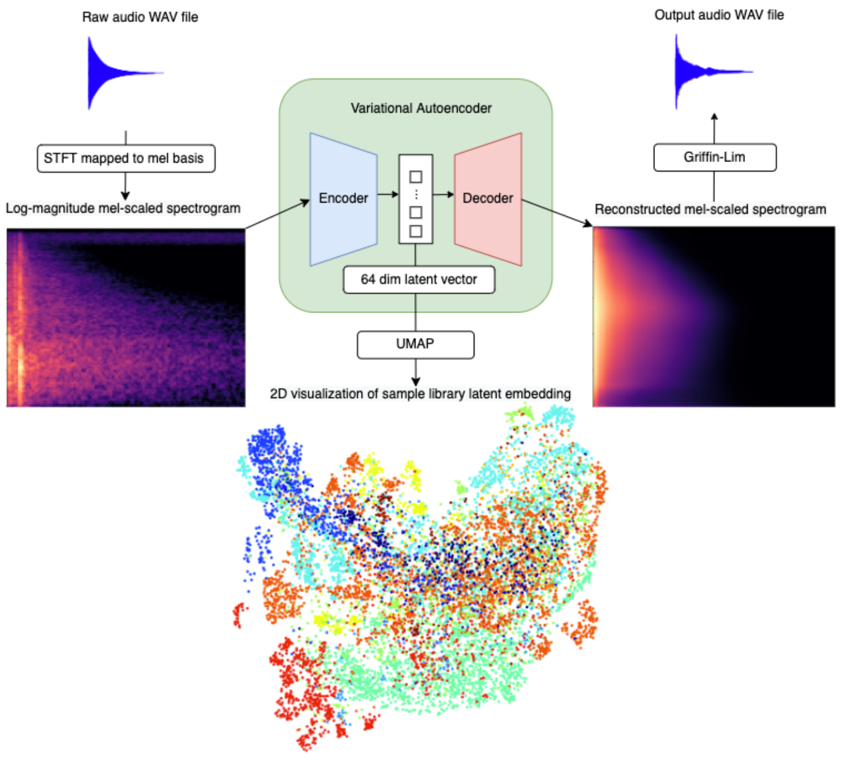


Figure 6.2.: All modules in the complete system.

6.4.1. Information Flow During Training

The spectrogram generating module employs a VAE with inverse autoregressive flow layers and comes after the audio to spectrogram system and before the spectrogram reconstruction system. The VAE is trained on spectrograms in the form of two-dimensional matrices representing images and, thus, outputs reconstructed matrices corresponding to the input during training. Before training, the sample library is preprocessed by truncating or padding the audio samples to a fixed length of two seconds. The audio files are then transformed into Mel spectrograms which are input to the VAE in mini-batches. Training details are included in section 7.3 as different hyperparameters are tested for the system. During training, the input spectrograms are encoded into a high-dimensional latent vector. The latent vector represents the space in which the model learns input representations. The decoder of the VAE then applies transposed convolutions to the latent vector. It shapes it into a matrix with the exact dimensions as the input, trying to reconstruct the training input data as accurately as possible.

6.4.2. Information Flow During Sample Generation

The sample generating process of the system starts with the visualization of the latent space. The user of the system selects a sample library such as the one described in

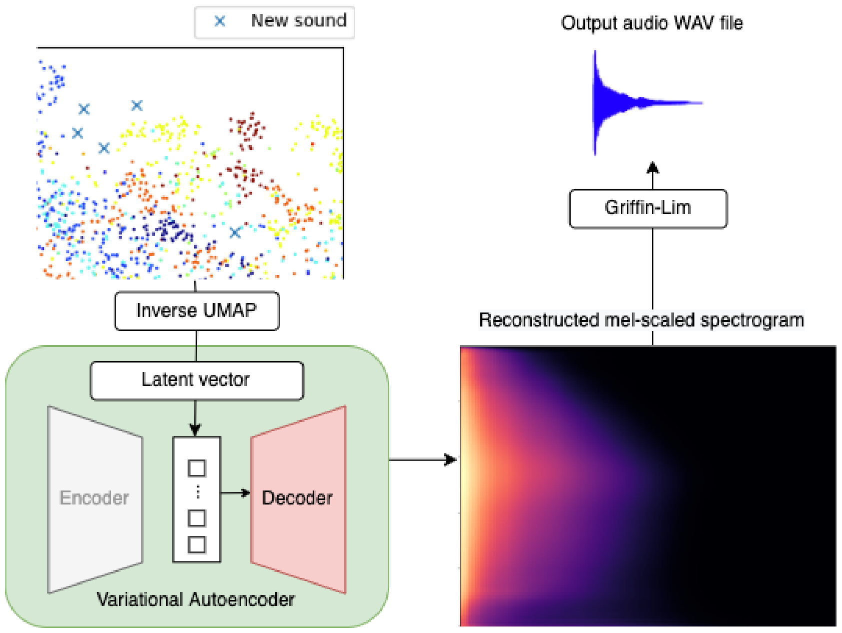


Figure 6.3.: Modules used for generating new samples. The user can click on any point on the sample library visualization. The result is the generation of an audio sample with audio characteristics defined by the selected point on the map.

section 5.1. A trained VAE is a prerequisite for sample generation. The VAE encoder must embed all the audio files in the selected sample library into its high-dimensional latent vectors. UMAP can then reduce these vectors into a set of two-dimensional points, which can be plotted for visualization. Figure 6.3 shows how a user can click on multiple points of the visualization, marked on the map with a cross. The x and y coordinates are transformed into high-dimensional latent vectors with the inverse UMAP transform. The latent vectors can then be decoded into Mel spectrograms by the decoder part of the VAE. Griffin-Lim is then applied to the spectrograms, and audio is output by the system.

6.5. Tools

In this section, all of the software used during the system's development process is documented. An explanation of how the different software frameworks were used and why they were used is also given. Working with audio in combination with complex deep learning models demand computational power, and many of the computational tasks of the system were performed on the NTNU IDUN high performance computing cluster (Själander et al., 2019). The cluster currently runs multiple nodes with two Intel Xeon cores per node and up to 128 GB of RAM and has NVIDIA Tesla P100 or V100 GPUs with 16 or 32 GB RAM available. The nodes run with the CentOS Linux distribution. Most of the experimentation phase was conducted through running interactive Jupyter Notebook sessions hosted on the IDUN servers from a home computer using SSH tunneling.

- Python **3.7.7**
  - Anaconda **4.11.0**
  - TensorFlow **1.15.5**
  - Scikit-learn **1.0**
  - Librosa **0.9.1**
  - Pyacoustid **1.2.2 + Chromaprint**
  - FuzzyWuzzy **0.18.0**
  - Pysoundfile **0.10.3.post1**
  - UMAP **0.5.2**
  - Matplotlib **3.4.2**
- gstPEAQ **0.6.1**
- GNU Bash **4.4.20**

6.5.1. Python

Python is not only a familiar programming language but a language offering a vast range of machine learning, data visualization, and audio processing frameworks. The different framework options available resulted in efficiency throughout the development process. The majority of the frameworks covered by the literature review in chapter 4 were also developed in Python. A discussion of the different libraries used with Python is given in the following paragraphs.

Anaconda Anaconda (Anaconda Inc.) is an easy-to-use distribution platform and package management system for Python and R programming languages for scientific computing. The Anaconda distribution comes with more than 250 packages automatically installed - many of which are mentioned in this section.

Tensorflow Tensorflow (Abadi et al., 2015) is an open-source software library initially developed by researchers and engineers from the Google Brain team, providing tools for deep learning and flexible numerical computations. The architecture is flexible and intended to be used across a range of different processing units and devices. Tensorflow 1, the version used in the developed system, is based on defining the computational graph for the model before running it, as opposed to Tensorflow 2, which uses an immediate execution style. Both versions have advantages over the other, but as the initial Variational Autoencoder model used Tensorflow 1, this was used throughout the development phase.

Scikit-learn Scikit-learn is a Python module for machine learning. The library provides several tools for machine learning-related tasks, including file handling, preprocessing of data, dimension reduction, data exploration, and evaluation. The evaluation and dimension reduction classes within Scikit-learn were used frequently during the development process and are a frequent dependency among many of the projects detailed in chapter 4.

Librosa Librosa (McFee et al., 2022) is a Python framework for music information retrieval systems that offer methods for audio loading, time-domain processing, spectral representations, and phase recovery. The framework was frequently used in the early experimentation phase for this thesis, especially in section 7.1 when comparing different spectrogram formats and parameters, as well as spectrogram reconstruction methods. Librosa was used in numerous of the studies mentioned in the literature review in chapter 4, where it was most commonly used as a framework for audio to spectrogram systems, but also spectrogram reconstruction systems.

Pyacoustid and Chromaprint The Chromaprint-based web service Acoustid is a high-quality, open-source acoustic fingerprinting system. Chromaprint is an FFT-based algorithm with logarithmically scaled frequency bins known as a Chromagram (Bartsch and Wakefield, 2005), indicating how much energy is in each of the 12 pitch classes. Acoustid is an API written in C utilizing the Chromaprint algorithm for generating compact fingerprints from audio files. Pyacoustid provides the Python bindings for Acoustid and was used for fingerprinting audio throughout the development process.

FuzzyWuzzy Fuzzywuzzy provides an easy-to-use string matching library for many languages, including Python. For evaluation of the generated audio, fingerprinting was done. A standard metric for measuring the distance between the generated fingerprint strings is Levenshtein distance. Fuzzywuzzy also offers a speedup option for more efficient calculation of large fingerprints. Among the provided methods for distance calculation, "Simple Ration was used.

SoundFile Librosa utilizes the Soundfile framework for handling read and write operations with audio files. It is built upon libsndfile, CFFI, and NumPy. Experiments involving audio files and development involving input-output (IO) operations with different audio formats were done using SoundFile.

UMAP Uniform Manifold Approximation and Projection (McInnes et al., 2018) is a manifold learning and dimension reduction algorithm provided as an Anaconda and Python library compatible with Scikit-learn. It is designed using the same API and is intended to be used in the same manner as other dimension reduction algorithms provided in Scikit-learn, such as t-SNE. UMAP has been used throughout the entire development process for dimension reduction and produced the most desirable data embedding, and used significantly less time and computer resources.

Matplotlib Matplotlib (Hunter, 2007) is a comprehensive plotting library for Python for creating animated, interactive, and static visualizations. Static visualizations were used not only for the main visualizations of the latent space but also for visualizing waveforms and spectrograms through Librosa's Matplotlib-based display functions. A third-party package, mpl_point_clicker, was used for interactivity with the plot when generating new samples with the system. This package conveniently changes the visualization map into a user interface, facilitating the generative possibilities of the system.

6.5.2. GstPEAQ

Based on the PEAQ system, as discussed in Section 4.3.2, Holters and Zolzer (2015) made an open-source implementation of the algorithm. Their implementation, GstPEAQ, utilizes Gstreamer (Taymans et al., 2018), which is a versatile framework for creating multimedia applications. Even though it does not compute all values within the allowed tolerance for the test signals originally used by Salovarda et al. (2005), it still claims to provide an alternative to listening tests. GstPEAQ is used as an evaluation metric in all the reconstruction experiments in Section 7.1 and as an addition to Mean Opinion Scores (MOS) for evaluating the complete system.

6.5.3. GNU Bash

As some of the development was conducted on IDUNs Linux-based operating system, GNU Bash (Foundation, 2020) was an essential tool for automating and scripting command-line tasks. Some projects discussed in the literature review, such as WaveNet (van den Oord et al., 2016), offered several Bash script recipes for preprocessing, training, and audio generation.

7. Experiments and Results

The architecture defined in chapter 6 was chosen based on experimentation with available options for each module conducted in this chapter. Therefore, experiments with the entire system pipeline were conducted in section 7.3 after selecting the best options for each module in the preliminary experiments.

The experimentation conducted in section 7.1 investigated the performance of different spectrogram formats and reconstruction methods to find the best options for the first and last module in the system shown in figure 6.1. Section 7.2.2 experimented with the dimensionality reduction (DR) techniques detailed in section 3.5 by trying to find the best way to visualize a sample library. Based on the results from the experiments, the complete system pipeline was used for experimentation with sample generation in section 7.3.

7.1. Spectrogram Representation and Reconstruction Experiments

The following experiment was performed as a part of meeting conditions **C2** and **C4**, which restrict the system to output audio of high quality. The literature review indicated that Mel and CQT spectrograms are the most frequently used spectrogram formats in systems that generate music since the logarithmic scaling of frequencies makes the spectrogram more compact (Cheuk et al., 2020). Linearly scaled STFT spectrograms need exponential times the pixels of a logarithmically scaled spectrogram to describe the same number of details in all octaves, which results in higher computational cost and inefficiency for a neural network. Despite not considering STFT as the spectrogram format of choice for the system, STFT spectrograms were included in this experiment to show how the audio quality metrics varied.

Subsection 7.1.1 details how the spectrogram representations were created and which parameters were used. In addition, two deep learning-based reconstruction methods are described, the first being a model based on WaveNet (subsection 4.1.1) and the second being based on Deep Griffin-Lim Iteration (subsection 4.1.2). In subsection 7.1.2, the resulting audio quality metrics are discussed.

7.1.1. Experiment Setup

Librosa (McFee et al., 2022) was used to create the STFT, Mel, and CQT spectrograms from 16-bit *.wav* files with a sample rate of 16 kHz. The audio files used for this experiment used a subset of the NSynth train set consisting of 4257 single note files of the note C4, which have a fundamental frequency of 261.6 Hz. One hundred four-second samples of this subset were concatenated to one audio file, which was used as the reference file for the rest of the experiment. The concatenated file was then transformed into spectrograms. As the system was built to feed a two-dimensional spectrogram into the subsequent pipeline module directly, it was unnecessary to save the spectrograms as actual image files. The following code snippet shows the parameters used to generate the spectrograms with Librosa.

```
import librosa
fmin = librosa.note_to_hz('C1')
s, sr = librosa.load('concatenated.wav', sr=16000)
mag_spec = np.abs(librosa.stft(s, n_fft=512, hop_length=128))
mel_spec = np.abs(librosa.feature.melspectrogram(y=s, sr=16000,
    n_fft=512, hop_length=128, n_mels=128))
cqt_spec = np.abs(librosa.cqt(s, sr=16000, hop_length=128,
    n_bins=168, bins_per_octave=24, fmin=fmin))
```

An essential aspect of the spectrogram is its size, as a larger spectrogram demands more computational resources further down the pipeline than a smaller spectrogram. Since the samples in the dataset were padded or truncated into a fixed length of two seconds with a sample rate of 16 kHz, a hop size of 128 would result in 250 feature vectors for each spectrogram. A window size of 512 would result in feature vectors with a size of 257 for the STFT spectrogram. The feature vectors of the Mel and CQT spectrograms were set to 128 and 168, respectively, which were typical values used in the literature review.

For reference, a reconstruction without any phase estimation algorithm was done with the istft function from the Librosa library. For the phase estimated reconstructions using the Griffin-Lim Algorithm, the functions griffinlim, griffinlim_cqt, and mel_to_-audio were used to reconstruct the STFT, CQT and Mel spectrograms, respectively. An open-source implementation of Deep Griffin-Lim Iteration (DeGLI) and WaveNet was setup. DeGLI was designed to be used with STFT spectrograms. Even though STFT spectrograms were not considered a format that the spectrogram generating module should use, it is still an intermediate format for some Mel spectrogram functions in Librosa. The mel_to_audio function is a convenience wrapper for the functions librosa.feature.inverse.mel_to_stft and the griffinlim function. DeGLI was included in the experiment to see if it would outperform the Librosa implementation of the GLA. If so, DeGLI could replace the Librosa implementation when reconstructing Mel spectrograms. The WaveNet was designed to be used with Mel spectrograms, and the same resolutions for window size, hop length, and the number of Mels from the Librosa reconstructions were used. DeGLI was trained for 50 epochs according to recommendations and configuration details from (Masuyama et al., 2019) while WaveNet trained for 1 million steps according to training recommendations provided by the open-source forum[1]. The WaveNet seemed to converge after around 800 000 steps, as shown in figure 7.1.

[1]https://github.com/r9y9/wavenet_vocoder/issues?q=training

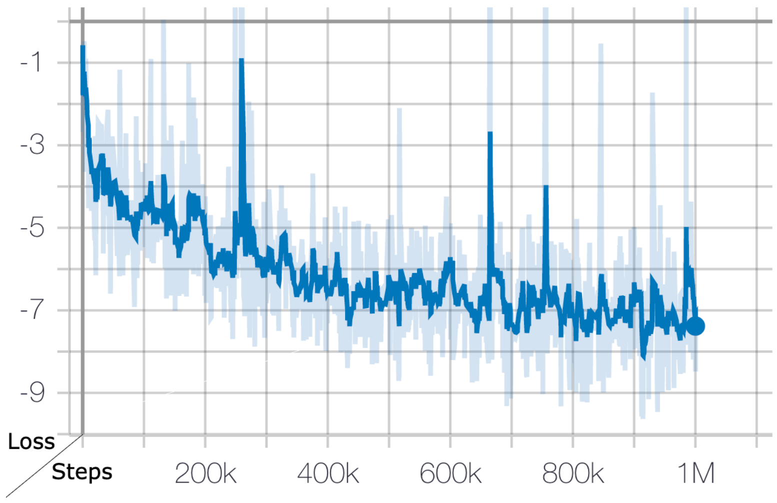


Figure 7.1.: WaveNet training loss.

The same 100 four-second samples used for processing in Librosa were held out from the training set for both DeGli and WaveNet. After training, these samples were reconstructed and concatenated to one single file.

7.1.2. Experiment Results

Two metrics were applied to determine if the reconstructed audio was of high quality and thus if condition **C4** was fulfilled. Audio fingerprinting is a method that can measure audio similarity objectively by creating a fingerprint for each piece of audio. A fingerprint is typically a string acting as a digital summary of the main attributes of the recording, such as intensity, frequency, and their anchor points in time. In this setup, the similarity between two audio fingerprints was computed with the Levenshtein distance (Levenshtein, 1966), with a score of 0 indicating different fingerprints and a score of 100 indicating identical fingerprints. Additionally, gstPEAQ was used to measure the Objective Difference Grade (ODG). ODG values were calculated using the original audio as the reference and the reconstructed audio as the signal that had undergone degradation. Table 7.1 shows the ODG and audio similarity results for the spectrograms and reconstruction methods.

The results from the audio fingerprinting clearly showed that the GLA reconstructions of CQT spectrograms were the most similar. After listening to the original and reconstructed audio files, however, there were severe degradations in the high frequencies of the audio with the CQT spectrogram. It is important to stress that audio fingerprinting is intended to compare audio characteristics rather than measure audio quality. The ODG score for the CQT spectrogram reconstruction seemed to account for the distortion of the high frequencies.

The audio generated by the WaveNet produced the same pitch and occasionally the

Reconstruction Method	**Spectrogram**	**Avg ODG**	**Avg Fingerprint Similarity**
No reconstruction	STFT	-3.506	57
Griffin-Lim	STFT	-0.585	63
Griffin-Lim	Mel	-1.966	61
Griffin-Lim	CQT	-3.326	69
Deep Griffin-Lim	Mel	-3.913	53
WaveNet Vocoder	Mel	-3.911	53

Table 7.1.: Reconstruction methods and spectrogram representations.

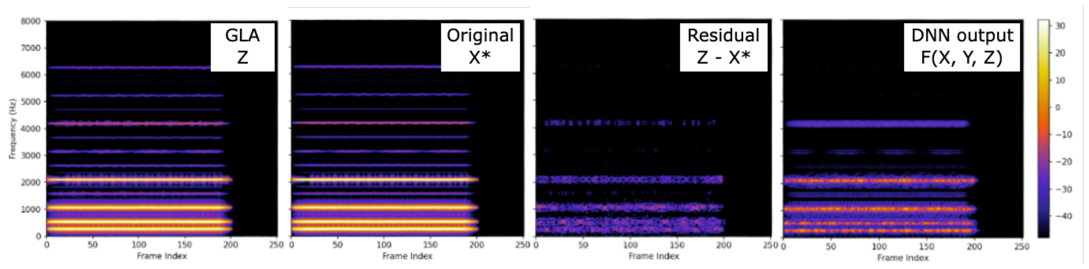


Figure 7.2.: Spectrograms produced by DeGLI. Z is the GLA estimated spectrogram. The residual information is shown as the difference between the estimated and the original spectrogram. The estimation $F(X, Y, Z)$ is denoted by "DNN output", where X is the initial spectrogram, Y is the amplitude-replaced spectrogram, and Z is the closest consistent spectrogram to Y.

same timbre as the audio files in the test set. The generated samples occasionally lacked low and high frequencies and did not keep the same structure, as the timbre sometimes seemed to change throughout each two-second sample. Based on the default parameters recommended in the open-source implementation, the receptive field was 31.5 ms, which is likely to cause fluctuations in the timbral structure. The relatively small receptive field of WaveNets is a known limitation from the literature review, and the model was discarded as a spectrogram reconstruction method for the system. DeGLI scored the lowest on both the ODG and fingerprint similarity metrics. Figure 7.2 shows how the DNN produced an inaccurate phase residual estimate for a sample in the test set. The DNN appeared biased from training on a dataset of notes with the same pitch. The solid horizontal lines seemed to overfit the typical harmonics produced by the middle C note. A personal listening test revealed that DeGLI produced blurry high frequencies and occasionally smeared out the audio transients. DeGLI was discarded as a spectrogram reconstruction method for the system based on these results.

Evaluating audio quality is challenging as it is difficult for a single metric to evaluate all aspects of the audio. The experiment indicated that the Mel spectrograms achieved the highest ODG and similarity scores among the logarithmically scaled spectrograms. The subjecitve opinion of the author after listening to the reconstructions was that the high frequencies were less distorted, and the blurry artifacts that the literature review considered problematic for the GLA were not as prominent. Based on these results, Mel spectrograms and reconstruction with the GLA were used as the system's spectrogram format and reconstruction method.

7.2. Spectrogram Generation Experiments

The second module in the system pipeline consisted of the generative sub-module, a Variational Autoencoder (VAE), and a latent space visualization sub-module used to select points in latent space from which Mel spectrograms were generated. Together they form the spectrogram generating module, which is essential for answering research question **R2**, asking how well new timbres can be generated from a visual representation of a sample library.

7.2.1. Variational Autoencoder Experiments

The purpose of the generative sub-module in the system pipeline was to generate a variety of samples that were not already in the sample library, which solves condition **C5**. The sub-module had to be able to learn the concept of timbre and map out audio characteristics from the training data across the latent dimensions to generate a diversity of samples. Even though the sub-module output spectrograms, it was also indirectly responsible for meeting condition **C4** and ensuring that high audio quality was output because imprecise and blurry spectrograms could introduce audio artifacts when reconstructed. Thus, the objective of this experiment was to investigate architectural decisions and hyperparameters to ensure that the generated results accurately captured the diversity of the training data.

Experiment Setup Before setting up the model, the dataset of Mel spectrograms was created from the 10275 samples in the Native Instruments Battery 4 sample library. Based on the parameters that resulted in the highest ODG score in the audio quality experiment, the samples were converted to 128x250 Mel spectrograms according to the experiment. The spectrogram dataset was further divided into train, validation, and test sets consisting of 80%, 10%, and 10% of the training data. Even though the model was trained unsupervised, the dataset was split in a stratified manner, meaning that the class representation in each split was similar. Stratification ensured that the model could be properly tested by recreating all the drum hits in the test set.

The generative sub-module was initially set up with a Variational Autoencoder architecture based on SampleVAE (Frenzel, 2019) and is shown in figure 7.3. The encoder used four convolutional layers that reduced the size of the input spectrograms to feature maps of size 7x15 with 32 channels. A dense layer with 512 nodes and dropout was then

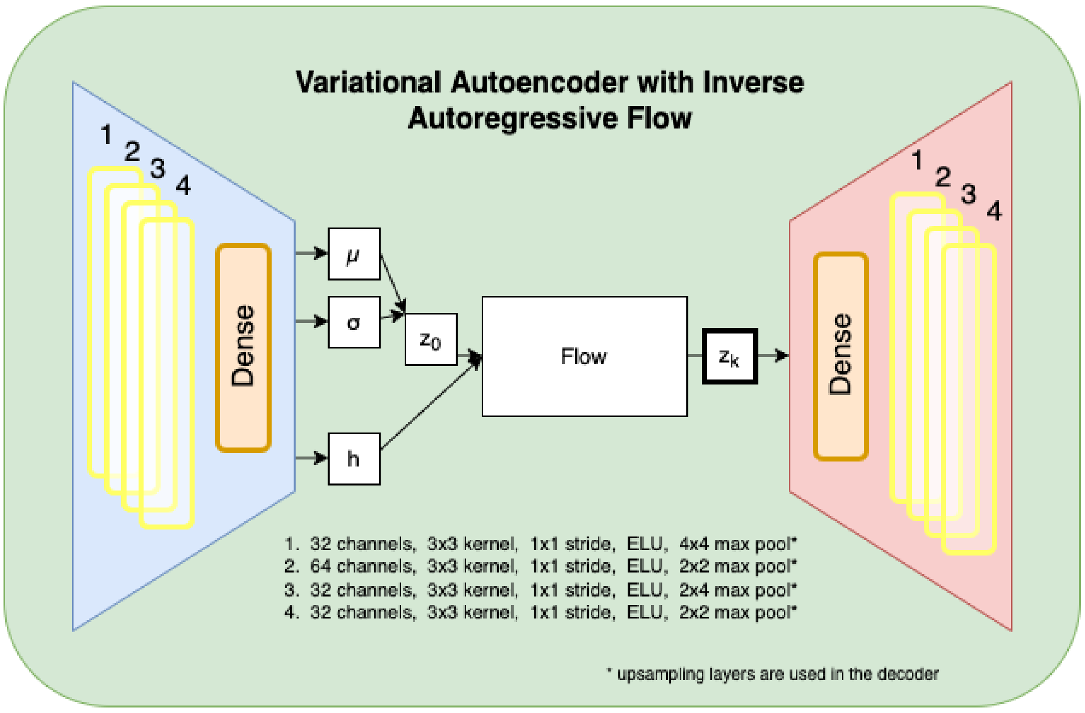

Variational Autoencoder with Inverse Autoregressive Flow
1
2
3
4
Dense
μ
σ
z_0
h
Flow
z_k
Dense
1
2
3
4
1. 32 channels, 3x3 kernel, 1x1 stride, ELU, 4x4 max pool*
2. 64 channels, 3x3 kernel, 1x1 stride, ELU, 2x2 max pool*
3. 32 channels, 3x3 kernel, 1x1 stride, ELU, 2x4 max pool*
4. 32 channels, 3x3 kernel, 1x1 stride, ELU, 2x2 max pool*
* upsampling layers are used in the decoder

Figure 7.3.: VAE architecture used for experimentation.

applied and passed through an activation function, from which the mean and standard deviation was predicted to form the latent vector z_0, concluding the encoder part of the VAE. The hidden output, denoted by h in figure 7.3, was also fed into the following flow layers. Inverse autoregressive flow (IAF) layers shaped the initial latent vector z_0 into a more complex distributed vector z_k. Each of the IAF layers consisted of 64 non-linear transformations. z_k was input to the decoder, which was almost identical to the reversed encoder architecture. A dense layer with dropout was passed through an activation function, just like the encoder. Instead of max pool layers, the decoder used unpooling layers and transposed convolutions to upsample the latent vector into a 125x250 size spectrogram. All activation functions were set to ELU according to recommendations by (Frenzel, 2019).

For the VAE to fulfill condition **C5**, it should be generating a latent space that maintains both the similarity of sounds locally by making clusters, as well as a global structure that ensures there are no sudden gaps between the clusters. The generated spectrograms also needed to be very similar to the spectrograms in the dataset to ensure that the system output audio of high quality and fulfills condition **C4**. Several models were set up to investigate which configurations of the VAE would lead to the best performance to meet these conditions. An overview of the models and the parameters that were experimented with is shown in table 7.2. All parameters were kept the same as the default setup in the open-source repository except the following:

- **Latent Dimensions** The number of latent dimensions used in the open-source repository was 64, but the author did not justify this number. Latent Timbre Synthesis (Tatar et al., 2020) used its two VAEs, the first one with a latent space of 256 dimensions, while the second VAE used only 8 dimensions. Three models with 8, 64, and 256 latent dimensions were set up.

- **Flow Layers** The VAE architectures from the literature review managed to generate spectrograms both with and without the use of flow layers. As the models investigating the latent dimensions use 10 flow layers, two additional models with 0 and 20 flow layers were set up.

- **Beta** The original VAE paper (Kingma and Welling, 2014) describes the Kullback-Leibler divergence (KL) term in the loss function as a regularizing term for the model. The ELBO loss has two pulling forces; the reconstruction loss is responsible for creating local clusters, while the KL term attracts the clusters to the center of the latent space. The paper suggests multiplying the KL term by $\beta = \frac{N}{M}$, where N is the size of the dataset and M is the batch size. A model named "Beta" in table 7.2 was set up to investigate the effects of scaling the KL term. For a training set with $N = 8220$ and a batch size of $M = 64$, β was set to 128.

- **Dropout** To avoid overfitting, dropout was applied to the dense layers by default. The model named "No Dropout" in table 7.2 was set up to investigate performance without using dropout. Training this model was difficult as the loss frequently exploded. As the model was rolled back to stable checkpoints upon divergence several times, it was decided to train it for 10 thousand steps.

Model Name	Latent Dimensions	Flow Layers	Trained Steps
8 dim	8	10	20k
64 dim	64	10	20k
256 dim	256	10	20k
0 flow	64	0	20k
20 flow	64	20	20k
Beta	64	10	20k
No Dropout	64	10	10k
Underfit	64	10	2k
Overfit	64	10	55k

Table 7.2.: Variational Autoencoder Hyperparameter Experiment Setup.

Training The models were trained with the default parameters from the open-source repository. These parameters included a learning rate of 0.001, batch size of 64, and validation of every 100th step. Initially, training was performed with a dynamic learning rate. If the validation loss did not improve within the next five validation steps, the learning rate was divided by five until stopping if it reached a minimum learning rate set to 0.00001. The generated spectrograms were blurry as the initial models converged at approximately 2000 steps with a dynamic learning rate. Based on these initial results and recommendations from the SampleVAE author to overfit the models to the training spectrograms, the other models were trained for 20 thousand steps. After evaluating the "64 dim" model at 20 thousand steps, it was further trained for 35 thousand steps to investigate the results of extensively overfitting the training data. The setup and results from the initial underfit model and the extensively overfit model are included in table 7.2 and table 7.3 under the names "Underfit" and "Overfit", respectively. The training times were effectively 1 hour for the underfit model and 13 hours for the overfit model when trained on two Intel Xeon Gold 6132 processors with a total of 28 cores on the IDUN HPC cluster (Själander et al., 2019). However, the training times were longer as exploding losses occasionally required the training to be rolled back to the last valid checkpoint.

Evaluation Metrics Evaluating generative models is problematic because it can be hard to differentiate between how well the model performs and how good the quality of the generated content is. Several metrics tend to be applied to measure different aspects of a system. A set of metrics was set up to focus on different qualities of the model performance to ensure adequate evaluation of the system. Each metric evaluated all models, and the results are shown in table 7.3. The evaluation metrics were as follows:

- **Frechet Inception Distance (FID)** To measure how similar two groups of images are, FID (Heusel et al., 2017) computes feature vectors for each set of images based on how similar the computer vision features are. The metric summarizes the distance between feature vectors from the Inception v3 model (Szegedy et al., 2015) meaning that lower scores indicate that the groups are similar. Lower scores are shown to correlate well with higher-quality images. For each model in table 7.2, the FID score was calculated.

- **Number of Statistically-Different Bins (NDB)** This metric was proposed by Richardson and Weiss (2018) as a way of measuring the diversity of generated examples. The training and generated examples can be compared by clustering the examples into $k = 50$ clusters, called bins, with the k-means algorithm. NDB is then calculated as the number of clusters of the training examples significantly different from the clusters of the generated examples by a two-sample Binomial test. Even though NDB is mostly used to measure mode collapse in GANs, a problem that should not occur for VAEs, this metric is included to assess the difference between the real and reconstructed spectrograms. An open-source implementation of NDB was setup. The spectrograms were resampled from 128x250 pixels to 16x32 pixels due to a limit of 1000 features in the open-source implementation.

- **PEAQ** To measure how the spectrogram generating step affected the audio quality, PEAQ-scores were used. This metric was used in the audio quality experimentation with the outcome that the Mel spectrogram format reconstructed with the GLA was selected as the spectrogram representation and reconstruction method. The spectrograms in the test set were reconstructed by the VAE models and then reconstructed to audio with the GLA to evaluate the system's audio quality end-to-end. The maximum amplitude was recorded for each sample so the reconstructed samples could be correctly scaled to account for the differences in loudness. ODG scores were then calculated between the original and reconstructed audio files. The total average ODG scores were included in table 7.3 while both the average and best ODG scores for each of the drum hits in the dataset are shown in table 7.4 for the "Overfit" and "Beta" models. As the gstPEAQ implementation needed a certain quantity of information to evaluate audio quality[2], each audio sample was concatenated with itself, that is, doubled in length, for the gstPEAQ algorithm to output valid numbers.

- **Audio Fingerprinting** As a means of measuring the similarity of the original samples and the generated samples in the test set, the average similarity between the fingerprints was calculated. The fingerprinting algorithm needed a certain amount of audio information, as with gstPEAQ, and the same double-length samples were used when estimating the ODG scores above.

Experiment Results The results in table 7.3 show that the images in the test set were similar to the train set in terms of NDB. All models except "Beta" generated image spectrograms with five or fewer statistically significant bins compared to the original spectrograms in the train set. This behavior was expected since the inductive bias of VAEs is to reconstruct all the classes in the training data. However, figure 7.4 shows how "Beta" failed to produce some of the classes in the dataset entirely, as the higher KL term seems to prohibit the latent space from being mapped out.

The FID scores seem to correlate with how overfit the models were. However, the models with larger latent dimensions seemed to generate spectrograms with similar features to the train set with less training. Another observation from the results was that a lower FID score points toward a slightly higher ODG score. In terms of audio quality, the average ODG scores place between *annoying* and *very annoying*, meaning that the level of audio degradation was very high. Even when overfit, the models seemed to average over the training examples. The reconstructed samples possessed essential audio characteristics but lacked the specificity and variations unique to every sample. The average fingerprint scores indicate that the reconstructed samples were not very similar to the samples in the original samples from the test set.

The models with the highest and lowest average ODG scores are compared in table 7.4 and shows the number of samples from each drum category that could have ODG

[2]https://github.com/HSU-ANT/gstpeaq/issues/6

Model	**NDB**	**FID**	**Avg ODG**	**Avg Fingerprint Similarity**
Test set	1	43.5	-	-
8 dim	0	141.6	-3.742	47.1
64 dim	4	133.7	-3.687	47.0
256 dim	4	121.8	-3.682	47.2
0 flow	5	123.6	-3.688	47.1
20 flow	3	121.0	-3.668	46.9
Beta	11	163.0	-3.836	46.4
No Dropout	3	140.0	-3.713	46.9
Underfit	4	167.3	-3.801	46.8
Overfit	2	122.5	-3.645	47.2

Table 7.3.: Hyperparameter Experiment Results.

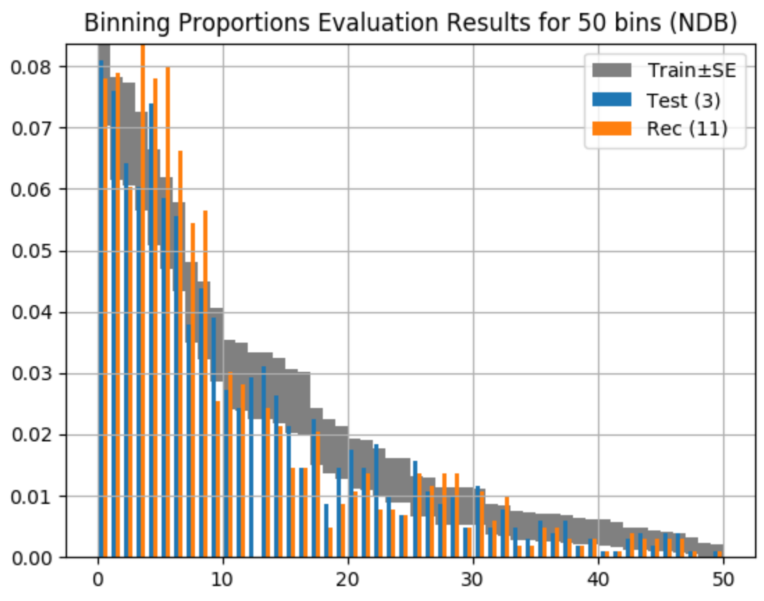


Figure 7.4.: NDB scores for the image spectrograms in the train set, test set, and the set of examples reconstructed by the model "Beta", the latter indicated by orange bars. The model failed to reconstruct several bins, shown by the orange bars falling outside the standard error of the train set.

scores calculated by gstPEAQ, along with the average and best ODG scores. Despite low average ODG scores for all models, some drum hits were more frequently reconstructed with a higher score. An interesting observation for the “Overfit” model is that hi-hats have a slightly higher average ODG score. The drum categories that had the best ODG score placing within *perceptible but annoying* seem to be mid-range sounds with pitched characteristics, which is the case for some snares, mallet drums, and metallic sounds. Regardless of the occasional generation of good quality sounds, the system performed poorly at fulfilling condition **C4** as the overall audio quality, as measured with PEAQ, was low. Since the “Overfit” model achieved slightly better ODG scores, it was used in the subsequent experiments. The drum hit-specific ODG scores for the remaining VAE models are shown in appendix A.

Model	**Overfit**			**Beta**		
Drum Hit	**n**	**Avg ODG**	**Best ODG**	**n**	**Avg ODG**	**Best ODG**
Clap	23	-3.778	-3.481	10	-3.903	-3.871
Combo	1	-3.801	-3.801	2	-3.886	-3.884
Cymbal	68	-3.623	-2.747	63	-3.734	-3.146
Hand Drum	18	-3.705	-3.393	15	-3.885	-3.804
HiHat	154	-3.468	-1.575	158	-3.779	-2.202
Kick	133	-3.814	-3.506	130	-3.892	-3.818
Mallet Drum	73	-3.683	-1.381	54	-3.896	-3.818
Metallic	57	-3.604	-2.51	63	-3.822	-2.421
Shaker	16	-3.727	-3.267	21	-3.881	-3.448
Snare	203	-3.655	-1.118	160	-3.87	-3.325
Tom	91	-3.615	-1.333	91	-3.814	-2.632
Wooden	8	-3.714	-3.261	1	-3.906	-3.906
NaNs	182	-	-	259	-	-
Total	845	-3.645	-	768	-3.836	-

Table 7.4.: End-to-End Audio Quality Results.

7.2.2. Dimensionality Reduction and Visualization

Audio should be generated from an interactive visual representation of a sample library to fulfill condition **C5**. Since the previous experiment showed that several of the implemented VAE models could generate distributions of spectrograms similar to the training data, dimensionality reduction (DR) algorithms could be applied to visualize all the samples in the dataset. This experiment investigated which DR method would generate the most meaningful local clusters while still preserving a smooth global structure. Additionally, since the developed system was restricted to generating samples from an interactive visualization, the possible errors from applying a transform and inverting it was measured.

Experiment Setup Based on the results from the VAE experimentation, the "Overfit" model was used due to the generated spectrograms having the most similar features as the original spectrograms. Each sample in the dataset was encoded into 64-dimensional latent vectors, as denoted by z_k in figure 7.3. The respective drum hit labels were also saved for later plotting. The following functions were used to reduce the high-dimensional vectors to two dimensions:

- **Principal Component Analysis (PCA)** The scikit-learn function sklearn.decomposition.PCA was used with the default parameters and the number of components set to two.
- **t-distributed Stochastic Neighbor Embedding (t-SNE)** The scikit-learn function sklearn.manifold.TSNE was used with most of the default parameters. Based on recommended values from the scikit-learn library, 2, 30, and 100 were used as values for the perplexity parameter, setting the size of the neighborhoods. For each of the values of perplexities, t-SNE embeddings were optimized for 250, 500, and 5000 iterations, generating nine embeddings.
- **UMAP** DR with UMAP was performed with similar values for the neighbor parameter, this time using 10, 50, and 100 based on recommendations from the UMAP documentation and the fact that the sample size of the NI Battery 4 Drums dataset is 10275. The minimum distances between the data points were set to 0.1, 0.5, and 0.8 for each neighbor parameter value, resulting in nine embeddings.

For better readability the resulting two dimensional embeddings were scaled with the scikit-learn function sklearn.preprocessing.MinMaxScaler to fit the outermost points between 0 and 1 on the x- and y-axis. The scaled plots are shown below, with PCA in figure 7.5, t-SNE in figure 7.6 and UMAP in figure 7.7.

Experiment Results Prior knowledge about the transformed data is needed to determine which DR method provided the most meaningful local clustering while still preserving a smooth global structure. Four of the most prominent clusters formed with all DR methods were kick drums, toms, hi-hats, and cymbals. Kick drums and toms typically contain low-frequency content, with toms usually having a longer amplitude decay. Hi-hats and cymbals are high-frequency sounds typically with short, and long amplitude decays. The PCA embedding seemed to encode the amplitude decay of the sounds on the x-axis. At the same time, the y-axis appeared to show the frequency content, from $y = 0$ indicating low frequencies to $y = 1$ indicating high frequencies. As the other drum hits were grouped in the center in terms of frequency and amplitude decay, there were few other distinct clusters in the PCA plot.

Both t-SNE and UMAP were able to cluster a fair amount of the other drum hits, such as the categories for metallic, hand drum, shakers, and wooden sounds. t-SNE focused more on preserving local structure even when perplexity was set to 100 than the

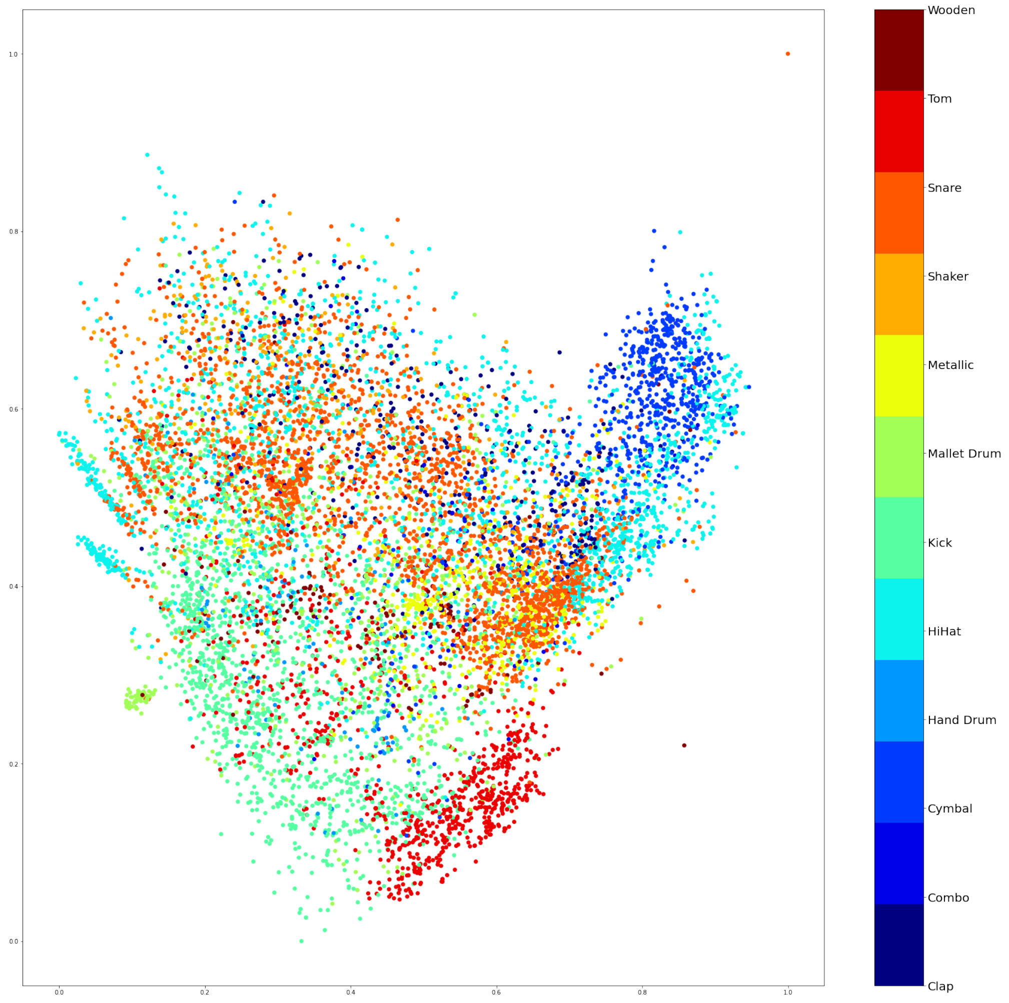


Figure 7.5.: PCA showing the two components with the highest covariance.

UMAP plots. Of the three DR methods, UMAP with neighbors set to 100 and minimum distance set to 0.8 seemed superior for embedding accurate local clustering and keeping a smooth global structure.

Since condition **C6** constrained the system to generate samples from an interactive visualization, the DR methods needed to have some support for inverse transforms. Since t-SNE produces local embeddings, finding a good inverse mapping is not feasible[3]. On the other hand, the PCA and UMAP functions used in the experiment setup facilitate inverse transforms. A reconstructed embedding was made by applying the DR transforms, and their respective inverse transforms to quantify the accuracy of the DR methods. The mean squared distance was calculated between the original and reconstructed spectrogram

[3]https://datascience.stackexchange.com/questions/34352/reconstructing-original-data-points-from-t-sne-output

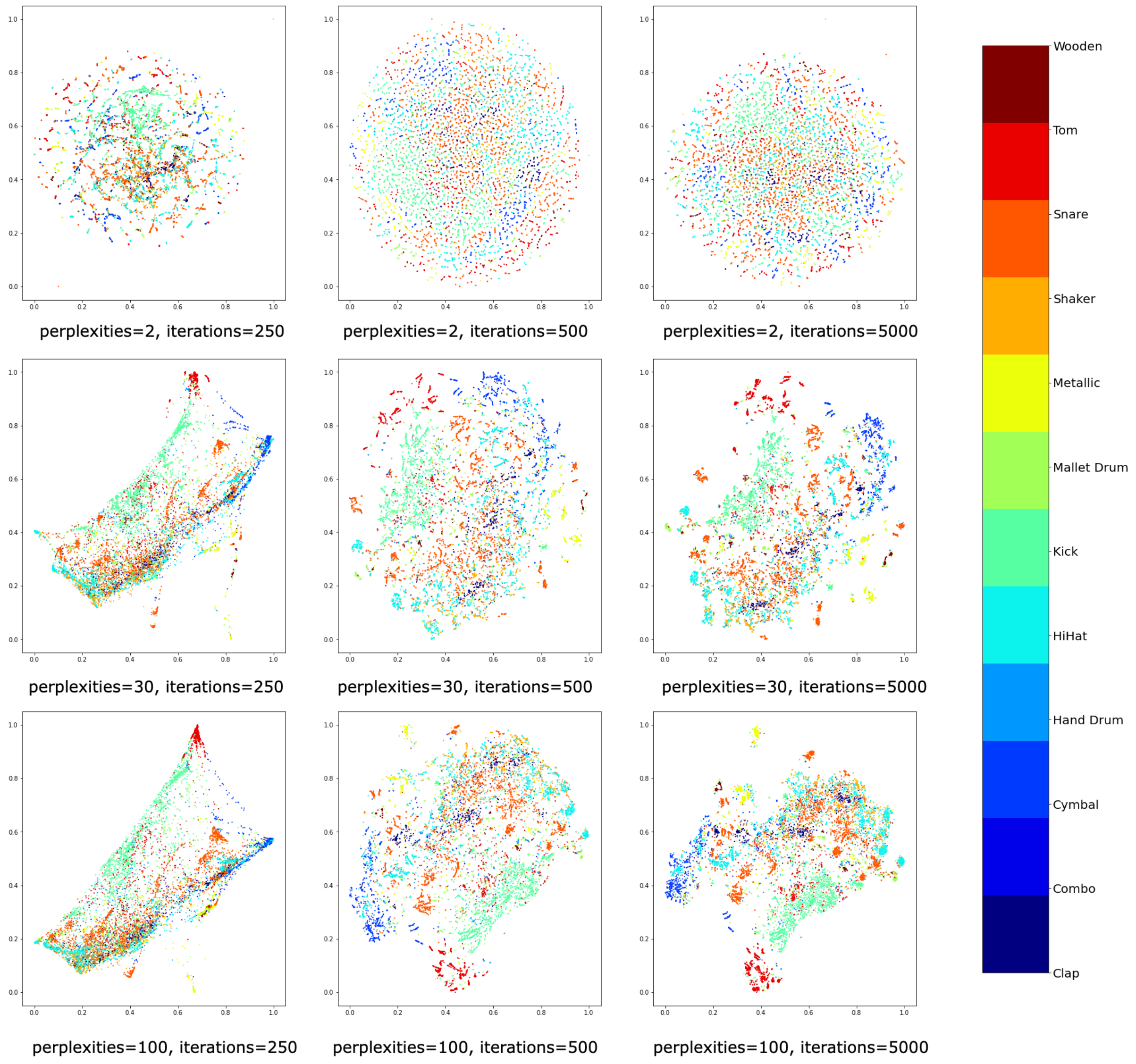


Figure 7.6.: Embeddings generated by t-SNE.

embeddings. For PCA and UMAP, the mean squared distance was 0.22 and 0.59. For perspective, the values of the original embeddings had a mean of 0 and ranged from -7.68 to 4.85.

The lack of invertibility discarded t-SNE as a DR method the system could use. Regardless of this, it was included in the experimentation to show that the local clustering capabilities of UMAP performed similarly. Despite having a higher reconstruction error than PCA, UMAP was chosen as the DR method for the system due to its superior performance in creating meaningful local clusters while still preserving a smooth global structure. The following experiment employed UMAP as the latent space visualization

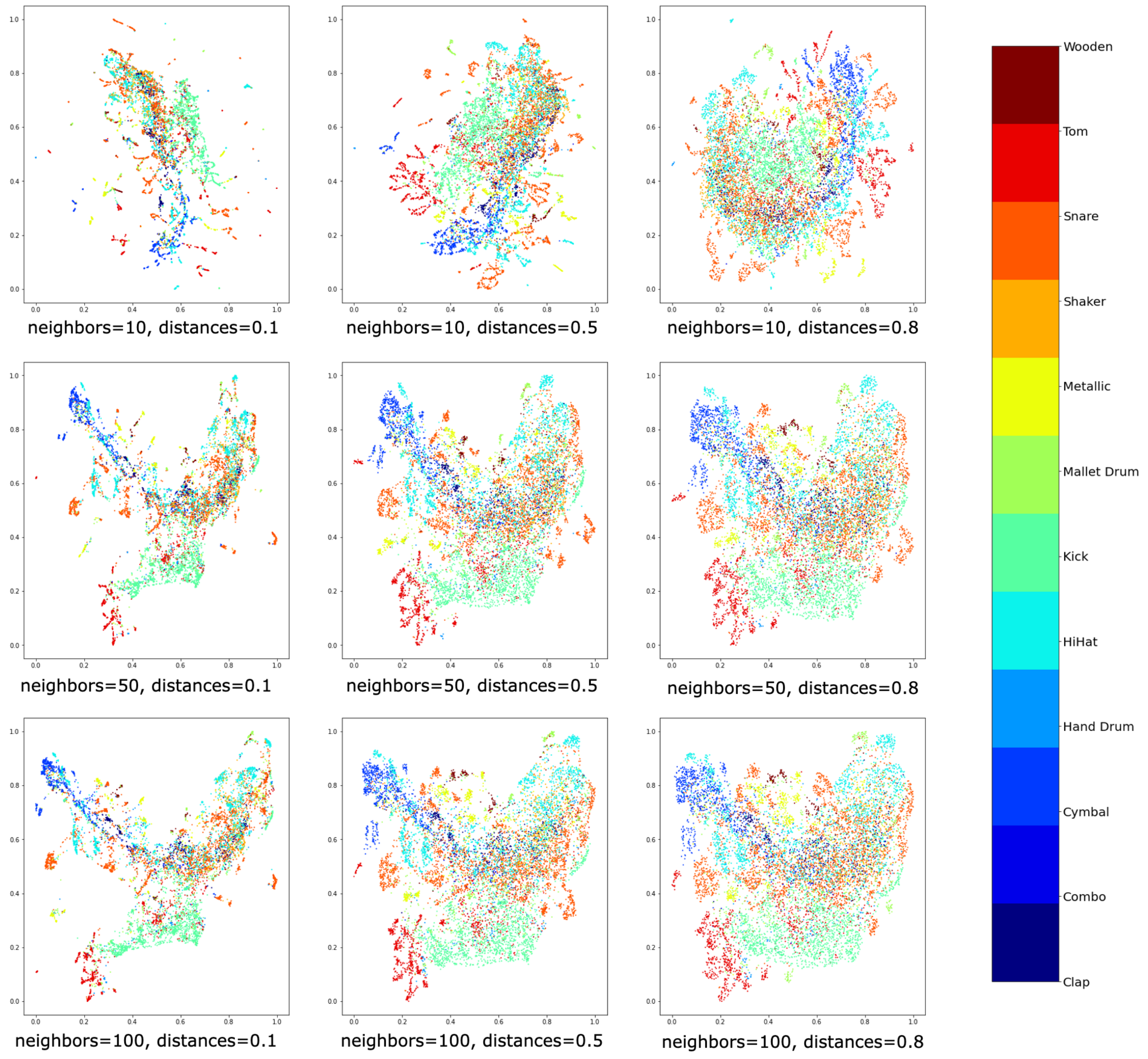


Figure 7.7.: Embeddings generated by UMAP.

sub-module in the completed system pipeline.

7.3. Completed System Experiments

The experiments conducted in section 7.1 and section 7.2.2 investigated the performance of the individual modules in the system pipeline. The audio to spectrogram module was represented by the Librosa library creating Mel spectrograms in the completed system pipeline. The spectrograms were then fed to the spectrogram generating system, which

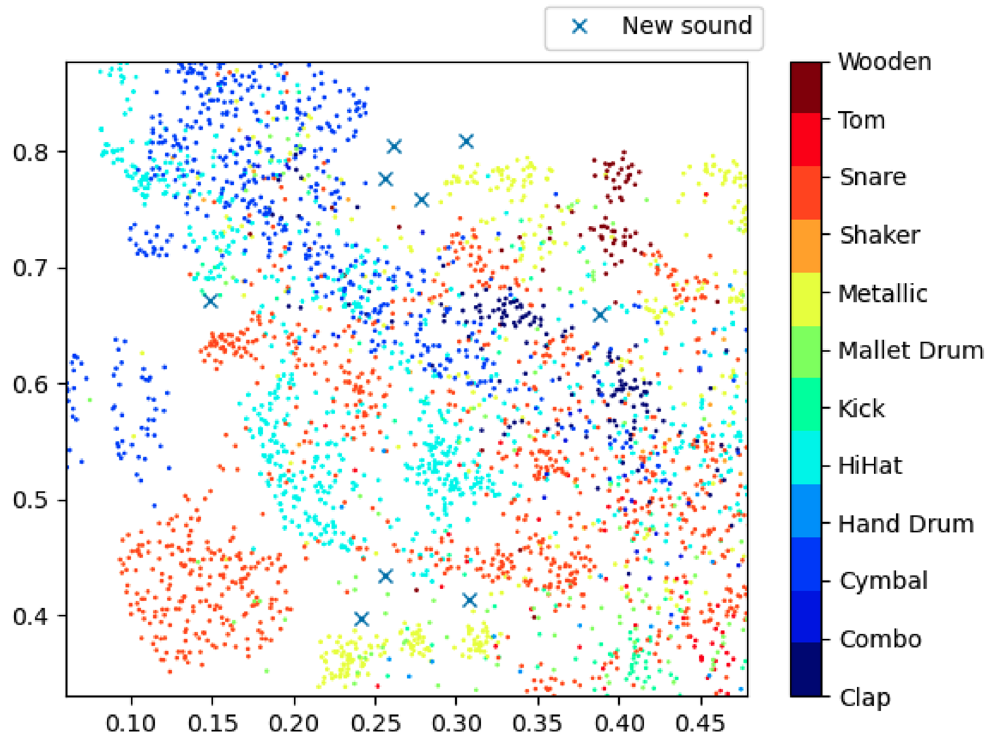


Figure 7.8.: Visualization used to generate samples.

consisted of the VAE and UMAP transform in tandem. The generated spectrograms were then converted to audio with the GLA. In this section, experimentation using the completed system pipeline was conducted to qualitatively investigate how effectively the system met condition **C5** by generating diverse samples from an interactive visualization.

7.3.1. Experiment Setup

Interactivity was added to test the generative abilities of the system according to the project goal and condition **C5**. The same UMAP plot from the previous experiment was set up, with a neighborhood size of 100 and a minimum distance of 0.8 between the points. To be able to select a point in the visualization of the sample library, the Matplotlib (Hunter, 2007) extension mpl_point_clicker was used. The system could record the coordinates of the clicked points using the clicker object in the library. The user-generated points, such as the ones denoted by "x" marks in figure 7.8, were inverted with the UMAP transform and decoded by the VAE decoder and output as spectrograms. The resulting spectrograms were then reconstructed with the GLA to audio.

A grid with 100 points was spread across the map to test the limits of the UMAP visualizations. Each corner point of the grid was estimated to cover most of the samples in the visualization but occasionally fell outside, as seen for the upper and lower right corners in figure 7.9. All of the 100 points were then converted to latent vectors by the inverse UMAP transform, decoded into spectrograms, and reconstructed to audio. As a means to visualize the diversity learned by the system, the generated spectrograms from each point across the grid are shown in a corresponding plot in figure 7.9. This process was repeated for all the VAE models that were set up in section 7.2.1. The spectrograms were resized only to show the first second of the sounds to better show details due to the high number of sounds with a short duration across the latent grid.

7.3.2. Experiment Results

The latent grids with corresponding spectrograms are shown for the models "Overfit", "Underfit", and "Beta" in figure 7.9, 7.10, and 7.11, respectively. A notable observation from the figures of the latent grids is that there is a significant difference between the local clustering of samples. The "Overfit" model enabled UMAP to create more clusters. The global structure also seemed smoother, with few white gaps between the clusters, as it seemed that the model managed to reconstruct a higher amount of examples for each type of drum hit. The "Underfit" model enabled UMAP to create almost the same amount of clusters, except for each cluster being slightly less spread out. The "Beta" model, on the other hand, showed less prominent clustering and poor global structure. The model also produced strange spectrograms with few variations. The spectrograms created by the "Underfit" model were blurry, and the differences between each spectrogram were not very drastic. However, the model did map out a diverse set of spectrograms across the latent space that seemed to average over the typical drum hit for each class. The "Overfit" model had a similar latent space, but the spectrograms were much more detailed, as shown by the horizontal lines created by the "Metallic" cluster points. The latent grid visualizations of the remaining models are shown in appendix B.

According to the spectrogram visualizations of the latent space, the models were capable of learning a diversity of spectrogram types but struggled with learning the differences within each class. The VAE seemed to average over most of the examples within each type of drum hit. The effect of averaging over the training examples were more apparent for the "Underfit" model, as each spectrogram in figure 7.10 transitions smoothly into the surrounding spectrograms. The overfit model appeared to create a more extensive palette of spectrograms with finer details. This is noticable in figure 7.9 as there are more sudden transitions between some spectrograms. The finer-detailed spectrograms generated by the overfit model appeared to be a contributing factor to the slight increase in audio quality in the experiment conducted in section 7.2.1. The diversity shown by the "Overfit" model seemed more effective in meeting condition **C5**.

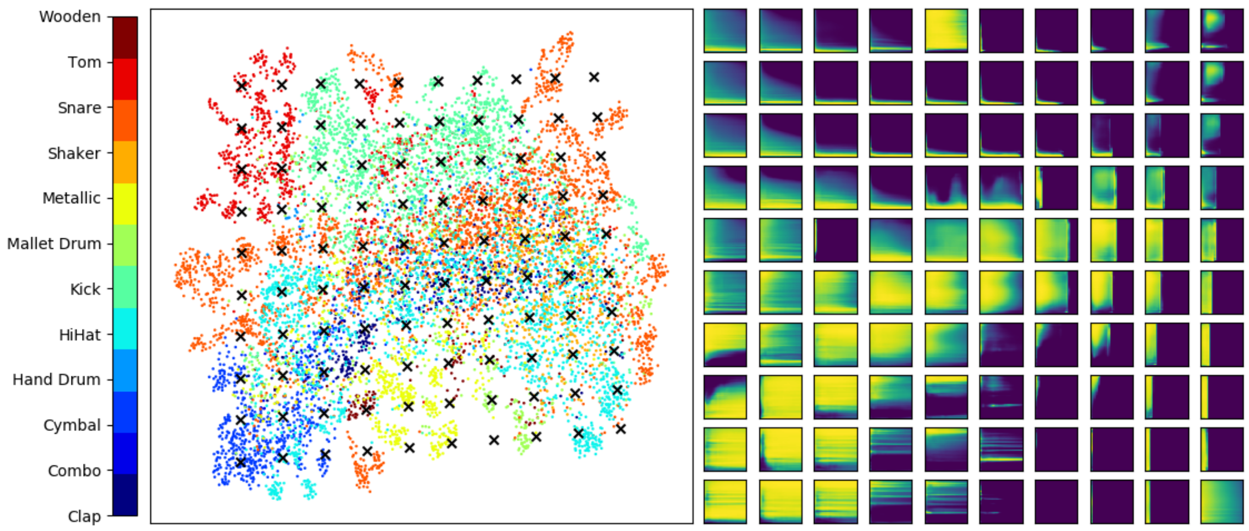


Figure 7.9.: Generated spectrograms from a latent grid for the “Overfit” model.

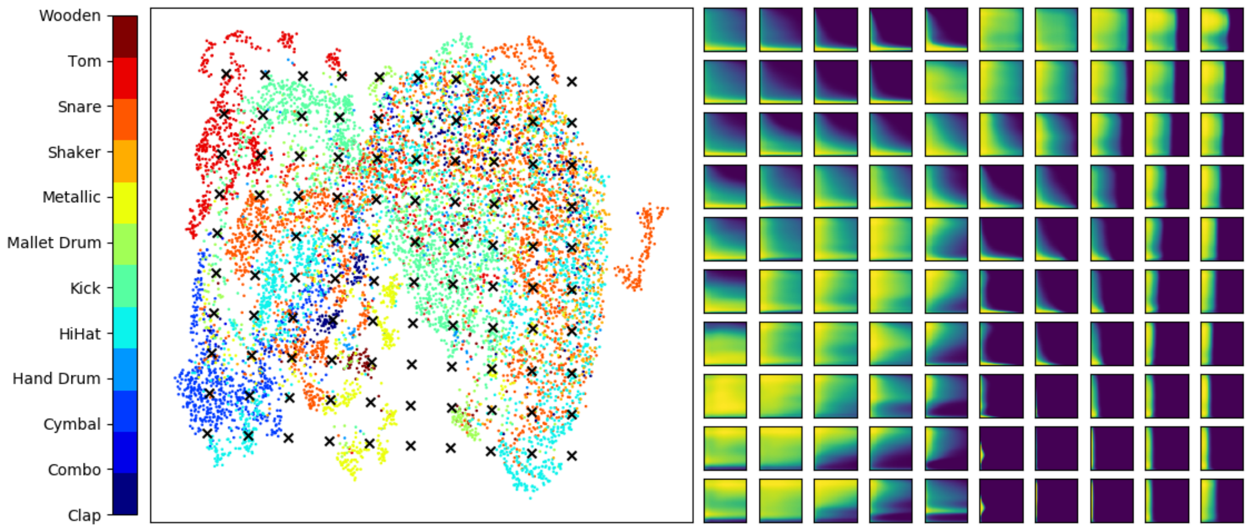


Figure 7.10.: Generated spectrograms from a latent grid for the “Underfit” model.

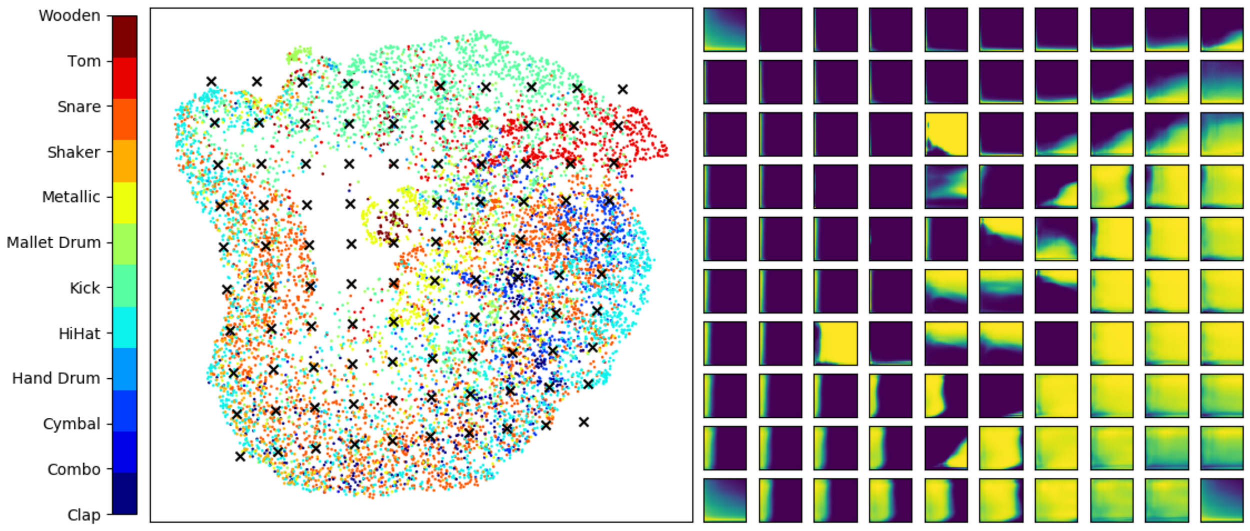


Figure 7.11.: Generated spectrograms from a latent grid for the “Beta” model.

8. Evaluation and Discussion

Based on the results of the experiments, the system was evaluated with regard to the research questions that were made and the conditions that were formulated as a part of reaching the project goal. This chapter evaluates specific aspects of the system in the following section. Section 8.2 assesses if the research questions have been answered, while section 8.3 analyses whether the system reached the project goal or not.

8.1. System Evaluation and Discussion

The two aspects that have been fundamental throughout the development of the system were audio quality and sample diversity, addressed in subsection 8.1.1, and 8.1.2, respectively. Each aspect was discussed based on system performance during experimentation and compared to the literature review results. The system's limitations were analyzed, and the limitations of the employed evaluation metrics.

8.1.1. Audio quality

Section 7.1 investigated the audio quality of spectrograms reconstructed with the Griffin-Lim Algorithm (GLA), Deep Griffin-Lim Iteration, DeGLI (Masuyama et al., 2019), and a WaveNet (van den Oord et al., 2016). The initial experiment showed that Mel spectrograms with the GLA resulted in the least degraded audio quality among the logarithmically scaled spectrogram reconstructions.

The motivation for including DeGLI as a reconstruction method in the initial experiment was due to the better PESQ (Rix et al., 2001) scores compared to the GLA. PESQ is different from PEAQ, and focuses on audio sharpness, background noise, and variable latency in the audio, whereas PEAQ focuses more on distortions and the error in harmonic structure. It was evident that DeGLI did not achieve the same results on PEAQ. The intention of using the NSynth dataset with a limited amount of pitches was to force DeGLI and WaveNet to learn the concept of timbre. Additionally, the NSynth dataset facilitates for convenient reproducibility of the experiment. The choice of dataset might have biased the ODG scores in favor of the GLA as only a limited range of frequencies was tested. The Native Instruments dataset includes sounds with frequency ranges across the entire human hearing range instead of the filtered NSynth dataset, with no frequency lower than 261 Hz. Therefore, the limitations of Mel spectrograms to represent information in the low frequencies compared to CQT spectrograms might not have been sufficiently demonstrated in the initial audio quality experiment. The outcome resulted in the Mel spectrograms and the GLA being used for the remaining experiments. The

GLA was a limiting factor for the audio quality when reconstructing spectrograms from the VAE models.

Neither Latent Timbre Synthesis (Tatar et al., 2020) or SampleVAE (Frenzel, 2019) conducted any audio quality evaluation. MOS scores based on a Likert scale from 1 to 5 were used by the WaveGAN and SpecGAN (Donahue et al., 2019) authors to evaluate the Speech Commands Zero Through Nine (SC09) dataset, with the results being 2.3 ± 0.9 and 1.9 ± 0.8, respectively. The authors stated that the poor qualitative ratings of SpecGAN were primarily due to the lossy Griffin-Lim inversion and not the generative procedure itself. One second samples were evaluated with MOS. Presenting listeners with samples of such a short duration is not necessarily enough to let them get a broad perspective of the quality. The short sample duration was also problematic for the PEAQ implementation in this thesis because the generated samples had to be doubled in length for the algorithm to give an ODG score. The developed system achieved ODG scores not far from the SpecGAN results. The reduction in audio quality compared to SpecGAN might be caused by the VAE averaging over training examples rather than favoring pixel realism typical for GANs. The WaveGAN and SpecGAN authors evaluated only the reconstructed output with MOS, while the ODG scores in this thesis compared a reference with a reconstruction. Additionally, the frequency range of the audio evaluated in this thesis is larger than the range in the SC09 dataset used by WaveGAN and SpecGAN.

GANSynth (Engel et al., 2019) implemented WaveGAN as one of many baselines for evaluation. The authors presented human judges with two four-second long samples of audio corresponding to the same pitch and asked them which sample had better audio quality. Their best model achieved a similar amount of "wins" as the original audio samples. WaveGAN performed worst among all the models in the comparison. It was problematic to compare the audio quality results from the experiments in this thesis compared to GANSynth and WaveGAN. GANSynth was restricted to only generating pitched audio, and the MOS scores from WaveGAN were based on speech audio. Thus, it is not that meaningful to compare the ODG scores of the audio generated by the system developed in this thesis.

It is important to stress that the usage of PEAQ was not ideal for measuring audio reconstruction quality for short audio samples, as the recommended input audio length is between 10 and 20 seconds. The difference grade between an audio signal that a generative model has reconstructed will be very different from a signal that has only undergone some form of perceptual coding. Other alternative evaluation methods, like PEMO-Q (Huber and Kollmeier, 2006), have been proposed instead of PEAQ. However, their approach also provides a metric score to evaluate some possibly distorted test signals against a high-quality reference signal. Additionally, the gstPEAQ implementation (Holters and Zolzer, 2015) had to be given a certain amount of data to calculate a score. Since only four-second audio samples were used during the PEAQ evaluation of the VAE models, the algorithm propagated several NaN values, making the results less reliable.

In the past, identifying electronic sounds was difficult because of the digital audio formats themselves. The same audio file would be opened, stored, or cataloged differently depending on whether it was from an MP3 file or a .wav file. Audio fingerprinting was made to identify the sound's attributes and create a virtual overview of the peaks and points for these attributes. Using audio fingerprinting to measure audio quality in terms of audio degradation can lead to misleading results unless the audio degradation drastically alters and distorts the audio features and their timing. In other words, using audio fingerprinting to evaluate the reconstruction quality of a series of short and monophonic samples could lead to ambiguous results. Comparing the PEAQ results with the similarity results shows that the latter metric fails to describe the audio quality for short excerpts of monophonic audio.

8.1.2. Sample Diversity

Evaluating generative models in the waveform domain is challenging. Neither Latent Timbre Synthesis (Tatar et al., 2020) nor SampleVAE (Frenzel, 2019) conducted any metric-based evaluation of the sample diversity of Latent Timbre Synthesis or SampleVAE. These models were not conditioned on pitch and are the most similar systems from the literature review. Measuring sample diversity by comparing the generated samples between models is difficult. The generated examples depend on the datasets used. The lack of established benchmark datasets for sample libraries makes for less comparability for these models. Instead, metrics like the Number of Statistically-Different Bins (NDB) and Frechet Inception Distance (FID) measures how different the original and generated distributions of samples are.

WaveGAN and SpecGAN (Donahue et al., 2019) measured the speaker diversity with MOS. The training data used by the WaveGAN authors was also the most diverse among the studies described in the literature review, ranging from single-word speech recordings, bird vocalizations, individual drum hits, and short excerpts of piano music. WaveGAN and SpecGAN achieved speaker diversity MOS scores of 3.2 ± 0.9 and 2.6 ± 1.0, respectively. Unfortunately, these MOS scores can not be compared to the NDB and FID scores used to measure the sample diversity of the system developed in this thesis.

GANsynth (Engel et al., 2019) evaluated model performance with several metrics, including NDB and FID. The best GANsynth (Engel et al., 2019) models achieved NDB scores of only 29.3 out of 50 bins, indicating that more than half of the clusters of generated examples were statistically different from the original examples. Mode collapse is a known problem for GANs, and the VAE models that were experimented with in section 7.2.1 had low NDB scores as expected from the mode-covering behavior of VAEs. Even the "Underfit" model achieved a low NDB score after training for only 1600 steps. The latent grids shown in the completed system experiments in section 7.3 qualitatively confirmed that the VAE models mapped out most of the drum hits to distinct areas of the latent space. The completed system experiments also showed a difference in the spectrogram fidelity, with the overfit models having finer details than the less overfit models.

The FID metric captures the feature similarity of two distributions of samples indicating that the overfit models lean towards lower FID scores. The superior GANSynth models had FID scores of 104 and 167, while all the VAE models tested in spectrogram generating experiment scored between 167 and 121. The metric appeared to correlate with how overfit the models were. A high FID score for the real examples in the test could result from a substantial variance of audio characteristics for each drum hit. Compared to the results from the evaluation of GANSynth (Engel et al., 2019), the test set used for experimentation with the VAE was four times less similar to the train set in terms of FID. Drawing a more similar test and train set would probably lower FID scores. Both NDB, FID, and the qualitative analysis of the latent grids generated by the VAE models indicate that the developed system could generate diverse samples.

8.1.3. Limitations

Several limitations were present throughout the development and experimentation with the system. One of the constraints was the audio duration and the fact that the system only works for audio samples with a length of two seconds. The limited sample length was due to the fixed spectrogram size set in the dimensions in the input and convolutional layers. The system would output errors upon a different input than 128x250 pixel spectrograms. Despite the two-second audio duration limit, longer samples could be generated by splitting them up and concatenating the generated results. Considering that the mean length was less than two seconds for most of the drum hits in the dataset, the impact of this limitation was less significant.

The system is also limited by depending on a labeled dataset for the visualizations to make sense. If the plotted points of the UMAP visualizations all had the same color, it would be impossible to determine which audio characteristics are present in each cluster. The developed system demands the user to possess knowledge about the characteristics of each drum hit in order to be able to navigate the visualization. Without some labeled data points, visualizing the sample library has no purpose. An idea to cope with this limitation is presented as future work in section 9.2.

Choosing only drum hits as training data limited the evaluation options, as the drum samples' quality was challenging to determine during experimentation. At the same time, the typical sample library used by audio content creators consists mainly of short samples. If the system had been modified to work with longer samples, the ODG and similarity scores would be more reliable but at the cost of using samples that might not represent the typical sample library used by audio producers.

Another limitation is the possible errors caused by the DR method employed by the system. The UMAP and inverse UMAP transforms are stochastic. Applying the UMAP transform on some high-dimensional vector followed by the inverse UMAP transform is unlikely to reconstruct the original vector. The more data used to compute the parameters for the initial transform, the less the resulting inverse will deviate on a qualitative level. The error size is also dependent on the density of data in different regions. If a point is inverted from a sparse region, it is more likely to have a significant round trip error. The grid of spectrograms in the figure on page 66 covers points that fall within the UMAP embedding space. If some of these points were to fall outside the bounds of clusters of this space, the inverse transform could operate poorly, sometimes generating strange results. An example of such results from points falling outside the UMAP embedding space is shown in figure 8.1.

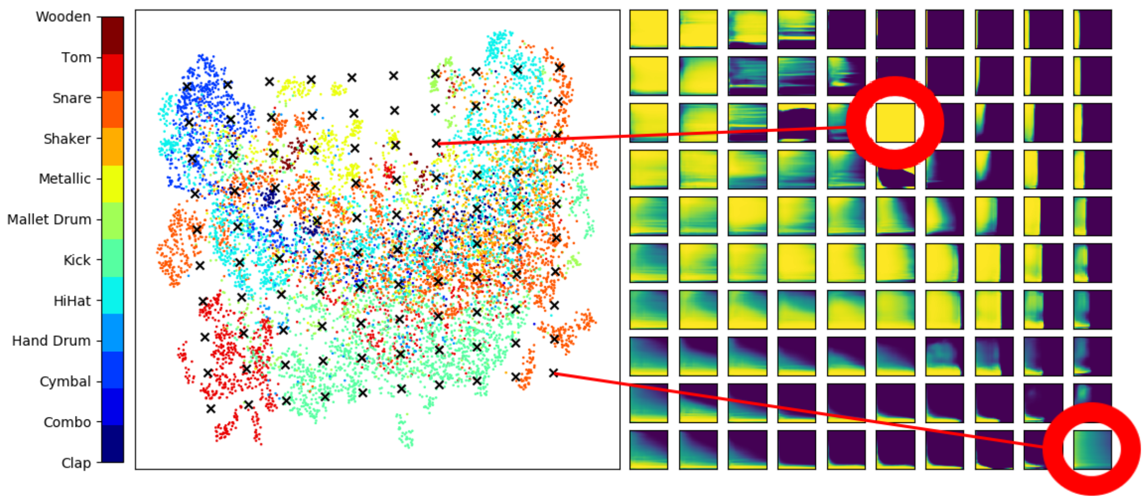


Figure 8.1.: Limitations with inverse UMAP transform when trying to reconstruct points that fall outside the embedding space. The generated spectrograms are very different from the surrounding spectrograms.

8.2. Research Questions

After evaluating the system, the research questions were reviewed to assess how well they were answered.

Research Question R1

How well can audio be generated in the waveform domain with deep learning?

Section 4.1 of the literature review described state-of-the-art within the field of deep generative audio modeling. Several distinct deep learning techniques for generating audio in the waveform domain were described. WaveNet (van den Oord et al., 2016) and SampleRNN (Mehri et al., 2017) were mentioned as initial attempts motivated by progress within Text-To-Speech (TTS) systems. Limitations for the methods were short receptive fields resulting in poor audio quality and high computational cost during training and sample generation. WaveGAN and SpecGAN (Donahue et al., 2019) were mentioned as methods reducing the computational cost but still suffered from inadequate audio quality.

Different approaches using intermediate spectrogram representations of the audio were described. GANSynth (Engel et al., 2019) was introduced with an alternative spectrogram representation unwrapping the phase onto the magnitude spectrograms and achieved better audio quality scores. MelGAN (Kumar et al., 2019) achieved decent audio quality results and fast inference times being trained on Mel spectrograms of audio. Tacotron 2 conditioned a WaveNet on Mel spectrograms and achieved exceeded state-of-the-art results within TTS. Additionally, Deep Griffin-Lim Iteration, DeGLI (Masuyama et al., 2019), was proposed as a new method to reconstruct phase from magnitude spectrograms better. The flexibility of using spectrograms as an intermediate audio representation (supporting **R2** along with the promising results of the conditioned WaveNet and DeGLI caused them to be included in the audio quality experimentation in section 7.1. The audio quality was measured by calculating Objective Difference Grade (ODG) scores and fingerprint similarities to answer the research question precisely. Due to insufficiency in the training data and DeGLI not performing as well with music as with speech, the Griffin-Lim Algorithm was used in the pursuit of fulfilling conditions **C2**, **C3**, and **C4**. Despite not using deep learning-based reconstruction methods in the final system, deep learning was used to generate spectrograms inspired by Latent Timbre Synthesis (Tatar et al., 2020) and SampleVAE (Frenzel, 2019) from the literature. In total, **R1** was considered to be answered while also contributing to the conditions above.

Research Question R2

How well can new sounds and timbres be generated from a visual representation of a sample library?
Answering this research question was done in two parts. The first part considered how new sounds and timbres were generated and was described in the literature review in section 4.2. State-of-the-art deep learning-based systems learning the concept of timbre were presented. Jukebox (Dhariwal et al., 2020) was mentioned as a system capable of generating complete compositions of music but was discarded as an option for the system due to high computational costs. DDSP (Engel et al., 2020) was presented as a novel approach utilizing synthesizer components controlled by neural networks to learn the timbre of monophonic pitched instruments. DDSP was not used in the developed system due to its limitation of only generating pitched audio. LTS and SampleVAE were described as flexible VAE-based systems capable of learning sounds with any characteristics. They were both spectrogram-based and intended to be used with sounds from sample libraries and were therefore investigated further during experimentation.

The second part of the research question included a review of dimensionality reduction (DR) methods used in sample library visualization systems such as The Infinite Drum Machine (McDonald et al., 2017). Three of the most common DR methods were presented in section 3.5. The performance of the DR methods was further investigated during experimentation in section 7.2.2. The round-trip error for the invertible DR methods was measured along with how well the DR method could cluster data points while maintaining a smooth global structure. From the experimentation, Uniform Manifold Approximation and Projection (UMAP) was chosen as the DR method for the system. Answering the research question was important for conditions **C5** and **C6** to be fulfilled. Based on the above, **R2** was considered answered and also contributing to the conditions above.

Research Question R3

How can the quality of generated samples be evaluated?
The literature review discussed various evaluation methods as a part of answering this research question in section 4.3. Mean Opinion Scores (MOS) were presented as an essential metric to include during audio quality evaluation. Perceptible Evaluation Metrics for Audio Quality (PEAQ) was mentioned as an alternative method trying to simulate the results of human rankings. Using fewer resources than MOS, ODG scores were measured with PEAQ throughout the conducted experiments. Further, subsection 4.3.3 presented factors that affect subjective perceptions of audio quality in music productions.

Additionally, audio fingerprinting was used to determine the similarity of the original and reconstructed samples. Despite using multiple metrics during experimentation, the audio samples were problematic to evaluate due to their short length. Regardless, **R3** was considered partly answered due to the shortcomings of the audio quality metrics on the samples used during experimentation.

8.3. Project Goal and Conditions

Create a deep learning system with the capability of generating a diverse set of high-quality audio samples from an interactive visual representation of a sample library.

In addition to the three research questions, six conditions were formulated as requirements that needed to be fulfilled for the project goal to be considered achieved. Five out of the six conditions were considered to be met. Regardless of the system not being able to generate audio of high quality, it still generated a diverse set of audio samples from an interactive visual representation of a sample library. The insufficient audio quality meant that the project goal was not fully met.

Condition C1

Deep learning based models are the basis for the developed system.
The completed system pipeline used a deep generative model in the form of a VAE with Inverse Autoregressive Flow to generate spectrograms. By the definition of deep learning formulated by Deng and Yu:

> “A class of machine learning techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation and for pattern analysis and classification.”

Despite discarding the deep learning-based spectrogram reconstruction methods based on the results from the audio quality experiment in section 7.1, the core of the system is the VAE model. Thus, the condition was considered to be met.

Condition C2

Audio in the waveform domain will be output by the system.
Spectrograms were chosen as an intermediate representation of audio. The spectrogram format allowed for a flexible and modular pipeline to be built. A variety of audio to spectrogram and spectrogram reconstruction modules could be experimented with without affecting the spectrogram generating system. The developed system outputs audio with a sample rate of 16 kHz and a bit rate of 16 bits after reconstructing spectrograms with the GLA. Despite the system being limited to generating audio with a fixed length of two seconds, this condition was considered met.

Condition C3

The system should work for any audio sample regardless of the characteristics.
Similar to condition **C2**, the use of spectrograms as an intermediate representation of the audio in combination with not requiring pitch information for the VAE models allowed the developed system to be used with any audio regardless of the characteristics. This condition more explicitly constrains the system to accept pitched sounds with fundamental frequencies and more experimental noise-based electronic samples typically found in the sample libraries of audio creators. The condition was met considering this.

Condition C4

The generated audio must be of high quality.
The audio quality experiment conducted in section 7.1 limited the system to use Mel spectrograms and the GLA for audio reconstruction. Evaluating the reconstructed audio from the spectrograms generated by the system resulted in ODG scores indicating an audio degradation between “annoying” and “very annoying”. Despite considering the results from the PEAQ analysis less reliable due to the short length of the samples, the audio quality could not be considered high. Due to this, the condition was not met.

Condition C5

The system should be able to generate a variety of samples that are not already in the sample library.
This condition was intended to specify that the system should generate any sample not already in the sample library and not necessarily samples with entirely new characteristics. As the completed system experiment showed that the system produced audio with desired characteristics, it was also evident that the system seemed to average over most training examples, limiting the number of new samples generated. The system generated more specific audio features by overfitting the VAE models. This condition was considered met as the system could generate samples from clusters formed for almost all of the drum hits in the sample library, and even express variations within most clusters.

Condition C6

The new samples should be generated from an interactive visual representation of the existing sample library.
By using UMAP as a DR method to visualize the latent vectors of a sample library in two dimensions, the system used this visualization as an interactive map that could generate samples. The completed system experiment described in section 7.3 showed how samples were generated by selecting any point within the limits of the UMAP embedding. Therefore, this condition was considered met by the system.

9. Conclusion and Future Work

This chapter presents the conclusion of the thesis and future work. Most of the future work suggested for the developed system is based on its current limitations.

9.1. Conclusion

A goal of creating a deep learning system with the capability of generating a diverse set of high-quality audio samples from an interactive visual representation of a sample library was proposed in the introduction of this thesis.

A literature review was conducted in chapter 4 to investigate state-of-the-art generative models in the waveform domain and deep learning-based systems attempting to learn the concept of timbre. The literature review emphasized spectrograms as a flexible intermediate representation of audio. Processing audio in terms of image spectrograms facilitated separate systems converting audio to spectrograms and reconstructing audio from spectrograms. The literature review described Variational Autoencoders (VAE) as promising generative models for processing audio from sample libraries in terms of spectrograms. An architecture using dimensionality reduction (DR) methods was described to reduce the latent space of the VAE into two dimensions to realize an interactive visualization of a sample library.

Mel spectrograms reconstructed with the Griffin-Lim Algorithm (GLA) were chosen as the spectrogram representation and reconstruction method after initial experimentation. The implemented system utilized a pipeline architecture of four processes. The first process used Librosa (McFee et al., 2022) to generate spectrograms. A VAE with Inverse Autoregressive Flow based on SampleVAE (Frenzel, 2019) was set up as the generative model of the system. The DR method Uniform Manifold Approximation and Projection (UMAP) was chosen due to being invertible and superior at clustering data locally while preserving a smooth global structure. The VAE model and UMAP were set up as the second and third processes in the system, while the fourth process consisted of the Librosa implementation of the GLA.

The system was trained on a sample library converted to Mel spectrograms and used the VAE encoder to create latent vectors for each spectrogram in the library. The high-dimensional latent vectors were reduced to two dimensions with UMAP for visualization. The system can generate a latent vector from any point on the visualization with the inverse UMAP transform and use the VAE decoder to generate a new spectrogram. The GLA is then used to reconstruct the generated spectrogram to audio.

Experiments were conducted to evaluate the diversity and quality of the generated audio. Different combinations of hyperparameters were investigated for the VAE model as the model was evaluated by its ability to reconstruct unseen samples and construct a diverse latent space. The audio quality of the reconstructed samples was measured in terms of audio similarity and Perceptible Evaluation Metrics for Audio Quality (PEAQ). The PEAQ metric measures the audio degradation of a reference signal and the corresponding degraded signal. It is intended to be used with 10 to 20 seconds of audio, but a length of only four seconds was used during experimentation, decreasing the reliability of the metric. The diversity of the generated examples was evaluated with Number of Statistically-Different Bins (NDB), Frechet Inception Distance (FID), and by qualitatively evaluating the generated spectrograms from a grid across the latent space.

The system generated a wide range of timbres from the training data but seemed to average over most of the samples within each class. Overfitting the VAE to the training data resulted in more intra-class diversity and higher fidelity spectrograms, generating audio with slightly better audio quality. Despite this, the overall audio quality was considered low. The reduction in audio quality was considered a result of the limitations of spectrogram reconstruction with the GLA. Additionally, the model seemed to average over the training examples such that the reconstructed spectrograms obtained the essential audio characteristics but not the per-sample uniqueness. Based on the poor audio quality of the generated samples, the goal of the thesis was not considered fully met as generating high-quality audio was specified as a condition for reaching the goal.

A review of the most notable contributions of this thesis include the following:

- The creation of a system generating diverse audio samples from an interactive visual representation of a sample library.
  This thesis describes the implementation background, conceptualization, and implementation of a deep learning-based system generating audio in terms of spectrograms. The implemented system creates latent embeddings for the spectrograms of the sounds in the sample library and visualizes them with dimensionality reduction with UMAP. The system generates diverse spectrograms that are converted to audio with the GLA.

- An overview and evaluation of state-of-the-art deep generative models in the waveform domain.
  This thesis conducted a literature review of state-of-the-art generative models in the waveform domain and systems learning the concept of timbre. The most promising methods were adopted and experimented with based on the project goal. Evaluation of the models included measuring the diversity and audio quality of the generated samples with several evaluation metrics.

- An understanding and evaluation of how different metrics can be used to measure the audio quality of generated samples.
  PEAQ and audio similarity were used as metrics to measure the audio quality of the generated samples throughout experimentation with the developed system. The discussion of the results included an assessment of the shortcomings of these metrics. Insufficient sample length was considered to make the results less reliable.

- An evaluation of dimension reduction algorithms for sample library visualization. A two-dimensional visualization of a sample library was a part of the developed system. Three dimensionality reduction algorithms were experimented with and evaluated according to their ability to cluster data locally while preserving a smooth global structure. The methods were also evaluated based on their invertibility and reconstruction error.

9.2. Future Work

Additional spectrogram reconstruction methods could be explored in future work to enhance the audio quality of the system. MelGAN (Kumar et al., 2019) was mentioned in the literature review as a method to reconstruct audio from Mel spectrograms but was not included in the audio quality experiment due to reporting lower audio quality than WaveNet. Damen (2021) outperformed WaveGAN (Donahue et al., 2019) at generating kick drums with a Progressively Growing GAN. Creating drum-specific generative models might be an option for future work if audio quality is more important than the diversity of the generated output.

One of the limitations present in the system is the need for labeled data to give meaningful visualizations. A solution to using the developed system with an unlabeled sample library would be to use an existing labeled sample library, such as Native Instruments Battery 4, to display the name of each drum category by calculating the center of gravity for each drum category. The clusters formed by the unlabeled sample library would then have descriptors such as “Kick” or “Snare” in the places these would occur for the labeled sample library.

Enhancing the user experience further by showing the sample names for some points in the sample library would be helpful. The Infinite Drum Machine (McDonald et al., 2017) is an excellent example of how the audio characteristics of a cluster are explained by describing the sound. The pitfall of using DR on high-dimensional audio features for visualization is that the axes are less informative if no labeling or information about the samples is provided.

This thesis evaluated the generated samples to be diverse based on training on a sample library with 12 drum hit categories. To further explore the diversity of the system, it should be trained on larger datasets with a broader range of samples. A typical sample library contains more than the 12 categories of drum hits used by the developed system. Training on a dataset of a larger scale would allow for more diversity in the generated output.

Another way of using the developed system to generate new samples more experimentally could be to select multiple points on the map and perform arithmetic operations with the latent vectors before converting them into spectrograms and audio. Such a feature would allow for taking the mean of the latent vectors of a kick and a hi-hat and end up with a completely different result. The difference between latent vectors could also be calculated. Implementing an option for combining the selected points on the sample library visualization would extend the sound design options of the system.

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Anaconda Inc. Anaconda software distribution, 2020. URL https://docs.anaconda.com/.

M.A. Bartsch and G.H. Wakefield. Audio thumbnailing of popular music using chroma-based representations. *Institute of Electrical and Electronics Engineers Transactions on Multimedia*, 7(1):96–104, 2005.

B. Benward and M. Saker. *Music in Theory and Practice Eight Edition.* McGraw-Hill, 2009.

M. Bosi and R. E. Goldberg. *Introduction to Digital Audio Coding and Standards.* Kluwer Academic Publishers, 2002.

K. Brandenburg and H. Popp. An introduction to MPEG Layer-3. *EBU Technical Review*, 2000.

Judith C. Brown. Calculation of a constant Q spectral transform. *Journal of the Acoustical Society of America*, 89:425–434, 1991.

Kin Wai Cheuk, Kat Agres, and Dorien Herremans. The impact of audio input representations on neural network based music transcription. *International Joint Conference on Neural Networks*, 2020. URL https://arxiv.org/abs/2001.09989.

Perry R. Cook. *Computer Music.* Springer Science+Business Media, LLc New York, 2007.

Wouter Damen. Generating kick drum samples with neural networks. Technical report, Radboud University, 2021. URL https://www.cs.ru.nl/bachelorstheses/2021/Wouter_Damen___1028002___Generating_Kick_Drum_Samples_with_Neural_Networks.pdf. Accessed: 09.06.2022.

Roger Dannenberg. A brief survey of music representation issues, techniques, and systems. *Computer Music Journal*, 17, 1998.

Li Deng and Dong Yu. Deep learning: Methods and applications. Technical Report MSR-TR-2014-21, Microsoft, 2014. URL https://www.microsoft.com/en-us/research/publication/deep-learning-methods-and-applications/.

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *abs/2005.00341*, 2020. URL https://arxiv.org/abs/2005.00341.

Christian Dittmar and Meinard Müller. Towards transient restoration in score-informed audio decomposition. *Proceedings of the International Conference on Digital Audio Effects*, 2015.

Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. *abs/1802.04208*, 2019. URL https://arxiv.org/abs/1802.04208.

Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders. *abs/1704.01279*, 2017. URL https://arxiv.org/abs/1704.01279.

Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. GANSynth: Adversarial neural audio synthesis. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1xQVn09FX.

Jesse Engel, Lamtharn (Hanoi) Hantrakul, Chenjie Gu, and Adam Roberts. Ddsp: Differentiable digital signal processing. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1x1ma4tDr.

Russ H Epworth-Sawyer, Jay Hodgson, and Mark Marrington. *Producing Music*. Routledge, 2019.

Derry Fitzgerald, Matt Cranitch, and Marcin Cychowski. Towards an inverse constant Q transform. *Audio Engineering Society - 120th Convention Spring Preprints*, 1, 2006.

Frederic Font, Gerard Roma, and Xavier Serra. Freesound technical demo. *Proceedings of the 21st Association for Computing Machinery international conference on Multimedia*, 2013.

Free Software Foundation. Gnu Bash version 4.4.20, 2020. URL https://www.gnu.org/software/bash/manual/bash.html.

Max Frenzel. SampleVAE - A Multi Purpose AI Tool for Music Producers and Sound Designers, 2019. medium.com [Online; posted 8-November-2019].

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *abs/1406.2661*, 2014. URL `https://arxiv.org/abs/1406.2661`.

D. Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

Michael Heideman, Don Johnson, and Charles Burrus. Gauss and the history of the fast fourier transform. *Archive for History of Exact Sciences*, 34:265–277, 1985.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems*, 2017. URL `https://arxiv.org/abs/1706.08500`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 1997.

Martin Holters and Udo Zolzer. GstPEAQ - an open source implementation of the PEAQ algorithm. In *18th International Conference on Digital Audio Effects*, 2015.

Harold Hotelling. *Analysis of a complex of statistical variables into principal components.*, volume 24. Journal of Educational Psychology, 1933.

R. Huber and B. Kollmeier. PEMO-Q-A new method for objective audio quality assessment using a model of auditory perception. *Transactions on Audio, Speech, and Language Processing*, 14(6):1902–1911, 2006.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *Computing Research Repository*, abs/1710.10196, 2017. URL `http://arxiv.org/abs/1710.10196`.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *abs/1312.6114*, 2014. URL `https://arxiv.org/abs/1312.6114`.

Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational inference with inverse autoregressive flow. *abs/1606.04934v2*, 2016. URL `https://arxiv.org/abs/1606.04934`.

Solomon Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brebisson, Yoshua Bengio, and Aaron Courville. MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis. *abs/1910.06711*, 2019.

Jongpil Lee, Jordi Pons, and Sander Dieleman. ISMIR 2019 tutorial: waveform-based music processing with deep learning, 2019. URL https://doi.org/10.5281/zenodo.3529714.

V. Levenshtein. *Binary Codes Capable of Correcting Deletions, Insertions and Reversals.* Soviet Physics Doklady, 1966.

Yoshiki Masuyama, Kohei Yatabe, Yuma Koizumi, Yasuhiro Oikawa, and Noboru Harada. Deep griffin-lim iteration. *Computing Research Repository*, abs/1903.03971, 2019. URL http://arxiv.org/abs/1903.03971.

W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, pages 115–133, 1943.

Kyle McDonald, Manny Tan, Yotam Mann, and Google Creative Lab. The infinite drum machine, 2017. URL https://experiments.withgoogle.com/ai/drum-machine/view/. Online; posted May 2017.

Brian McFee, Alexandros Metsai, Matt McVicar, Stefan Balke, Carl Thomé, Colin Raffel, Frank Zalkow, Ayoub Malek, Dana, Kyungyun Lee, Oriol Nieto, Dan Ellis, Jack Mason, Eric Battenberg, Scott Seyfarth, Ryuichi Yamamoto, viktorandreevich-morozov, Keunwoo Choi, Josh Moore, Rachel Bittner, Shunsuke Hidaka, Ziyao Wei, nullmightybofo, Adam Weiss, Darío Hereñú, Fabian-Robert Stöter, Pius Friesch, Matt Vollrath, Taewoon Kim, and Thassilo. Librosa: version 0.9.1, 2022. URL https://doi.org/10.5281/zenodo.6097378.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018. URL https://arxiv.org/abs/1802.03426.

Muhammad Huzaifah Md Shahrin. Comparison of time-frequency representations for environmental sound classification using convolutional neural networks. *abs/1706.07156*, 2017. URL https://arxiv.org/abs/1706.07156.

Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. *abs/1612.07837*, 2017. URL https://arxiv.org/abs/1612.07837.

Bob Moog. Midi: Musical instrument digital interface. *Journal of the Audio Engineering Society*, 34(5), 1986.

Vinod Nair and Geoffrey Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of International Conference on Machine Learning*, 27:807–814, 2010.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *abs/1511.06434*, 2016. URL `https://arxiv.org/abs/1511.06434`.

Eivind Rebnord. Generating Audio in the Waveform Domain. Technical report, Norwegian University of Science and Technology, 2021. Specialization Project.

Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. *abs/1505.05770*, 2015. URL `https://arxiv.org/abs/1505.05770`.

Eitan Richardson and Yair Weiss. On GANs and GMMs. *abs/1805.12462*, 2018. URL `https://arxiv.org/abs/1805.12462`.

Jean-Claude Risset and David Wessel. Indagine sul timbro mediante analisi e sintesi. *Bollettino del Laboratorio di Informatica Musicale della Biennale di Venezia*, 1982.

A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra. Perceptual evaluation of speech quality (PESQ) — a new method for speech quality assessment of telephone networks and codecs. *International Conference on Acoustics, Speech and Signal Processing*, 19:2125–2136, 2001.

David E. Rumelhart and James L. McClelland. *Learning Internal Representations by Error Propagation*, pages 318–362. MIT Press, 1987.

M. Salovarda, I. Bolkovac, and H. Domitrovic. Estimating perceptual audio system quality using peaq algorithm. In *2005 18th International Conference on Applied Electromagnetics and Communications*, pages 1–4, 2005.

Arnold Schoenberg. *Harmonielehre*, page 503. Leipzig and Vienna: Verlagseigentum der Universal-Edition, 1911.

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. *abs/1712.05884*, 2018. URL `https://arxiv.org/abs/1712.05884`.

Magnus Själander, Magnus Jahre, Gunnar Tufte, and Nico Reissmann. EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure. *arXiv:1912.05848 [cs]*, 2019.

S. S. Stevens, John E. Volkmann, and Edwin B. Newman. A scale for the measurement of the psychological magnitude pitch. *Journal of the Acoustical Society of America*, 8: 185–190, 1937.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *abs/1512.00567*, 2015. URL https://arxiv.org/abs/1512.00567.

Kıvanç Tatar, Daniel Bisig, and Philippe Pasquier. Latent timbre synthesis. *Neural Computing and Applications*, 33(1):67–84, 2020. URL https://doi.org/10.1007%2Fs00521-020-05424-2.

Wim Taymans, Steve Baker, Andy Wingo, Ronald Bultje, and Stefan Kost. *Gstreamer Application Development Manual (1.10.1)*. 12th Media Services, 2018.

Thilo Thiede, William Treurniet, Roland Bitto, Christian Schmidmer, Thomas Sporer, John Beerends, Catherine Colomes, Michael Keyhl, Gerhard Stoll, Karlheinz Brandenburg, and Bernhard Feiten. PEAQ—the ITU Standard for Objective Measurement of Perceived Audio Quality. *Journal of the Audio Engineering Society*, 48:3–29, 2000.

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *ArXiv*, abs/1609.03499, 2016.

Laurens van der Maaten and Geoffrey E. Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Gino Angelo Velasco, Nicki Holighaus, Monika Doerfler, and Thomas Grill. Constructing an invertible constant-q transform with nonstationary gabor frames. *Proceedings of the 14th International Conference on Digital Audio Effects*, 2011.

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. *abs/1703.10135*, 2017. URL https://arxiv.org/abs/1703.10135.

Alex Wilson and Bruno Fazenda. Perception & evaluation of audio quality in music production. In *Proceedings of the 16th International Conference on Digital Audio Effects*. Digital Audio Effects, 2013.

Alex Wilson and Bruno Fazenda. Perception of audio quality in productions of popular music. *Journal of the Audio Engineering Society*, 64:23–34, 2016.

A. Objective Difference Grade Scores

The following table shows the drum hit-specific ODG scores for the VAE models not included in the experimentation results in section 7.2.1. The average scores in this table vary between the scores of the best model ("Overfit") and the worst model ("Beta").

Model	8 dim			64 dim			256 dim		
Drum Hit	**n**	**Avg ODG**	**Best ODG**	**n**	**Avg ODG**	**Best ODG**	**n**	**Avg ODG**	**Best ODG**
Clap	24	-3.848	-3.652	22	-3.8	-3.541	21	-3.786	-3.555
Combo	2	-3.877	-3.877	2	-3.821	-3.768	1	-3.747	-3.747
Cymbal	68	-3.659	-2.724	71	-3.588	-2.539	70	-3.625	-2.551
Hand Drum	18	-3.805	-3.478	18	-3.769	-3.449	18	-3.758	-3.42
HiHat	153	-3.597	-1.388	163	-3.518	-1.933	156	-3.56	-1.98
Kick	132	-3.869	-3.698	133	-3.855	-3.698	133	-3.832	-3.602
Mallet Drum	75	-3.798	-1.71	72	-3.774	-1.472	72	-3.738	-1.535
Metallic	61	-3.716	-2.876	57	-3.685	-3.171	57	-3.711	-3.094
Shaker	18	-3.833	-3.615	16	-3.778	-3.578	14	-3.809	-3.697
Snare	206	-3.776	-1.871	208	-3.703	-1.407	205	-3.661	-1.256
Tom	91	-3.693	-1.649	91	-3.641	-1.286	91	-3.633	-1.71
Wooden	7	-3.843	-3.764	8	-3.797	-3.696	7	-3.747	-3.369
NaN	172	-	-	166	-	-	182	-	-
Total	855	-3.742	-	861	-3.687	-	865	-3.682	-
Model	**0 flow**			**20 flow**			**No Dropout**		
Drum Hit	**n**	**Avg ODG**	**Best ODG**	**n**	**Avg ODG**	**Best ODG**	**n**	**Avg ODG**	**Best ODG**
Clap	18	-3.814	-3.64	23	-3.797	-3.51	21	-3.84	-3.73
Combo	1	-3.827	-3.827	1	-3.78	-3.78	2	-3.861	-3.838
Cymbal	68	-3.585	-2.779	67	-3.598	-2.501	68	-3.618	-2.729
Hand Drum	18	-3.777	-3.437	18	-3.746	-3.462	18	-3.825	-3.681
HiHat	158	-3.558	-1.799	162	-3.514	-1.988	162	-3.577	-2.223
Kick	133	-3.836	-2.984	133	-3.828	-3.559	133	-3.864	-3.582
Mallet Drum	74	-3.71	-2.235	72	-3.724	-1.364	75	-3.667	-1.827
Metallic	59	-3.698	-2.84	58	-3.637	-2.648	60	-3.72	-2.589
Shaker	16	-3.81	-3.653	19	-3.779	-3.425	21	-3.78	-3.449
Snare	206	-3.699	-1.554	207	-3.673	-1.17	209	-3.749	-1.307
Tom	91	-3.646	-1.468	91	-3.641	-1.409	91	-3.672	-1.446
Wooden	8	-3.789	-3.558	7	-3.79	-3.62	8	-3.836	-3.761
NaN	177	-	-	169	-	-	159	-	-
Total	850	-3.742	-	858	-3.687	-	868	-3.713	-

Table A.1.: ODG scores for the VAE models for experimentation in section 7.2.1

B. Latent Grid Visualizations

The following figures show the UMAP visualizations from the VAE models not included in the results in the completed system experiments in section 7.3. The figures show the spectrograms generated from the points in the latent grid mapped across each visualization.

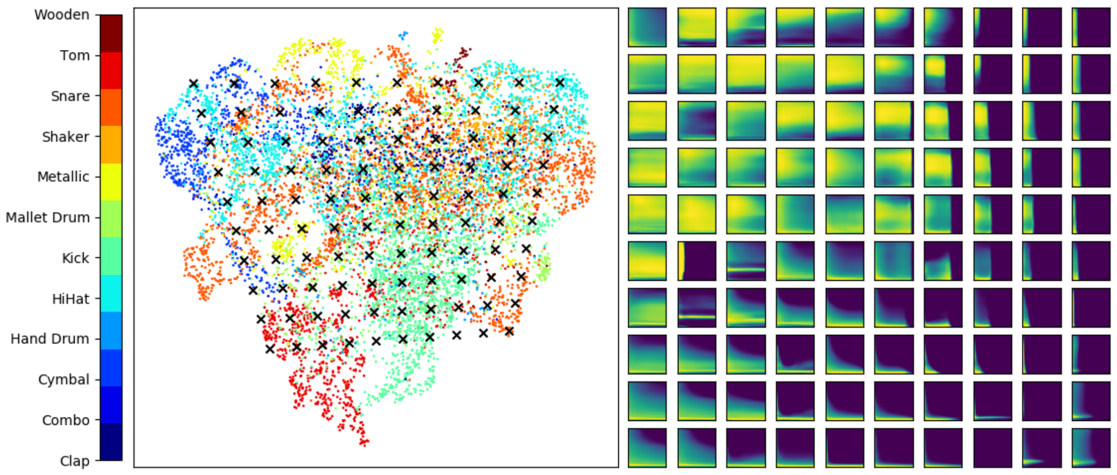


Figure B.1.: Generated spectrograms from a latent grid for the “8 dim” model.

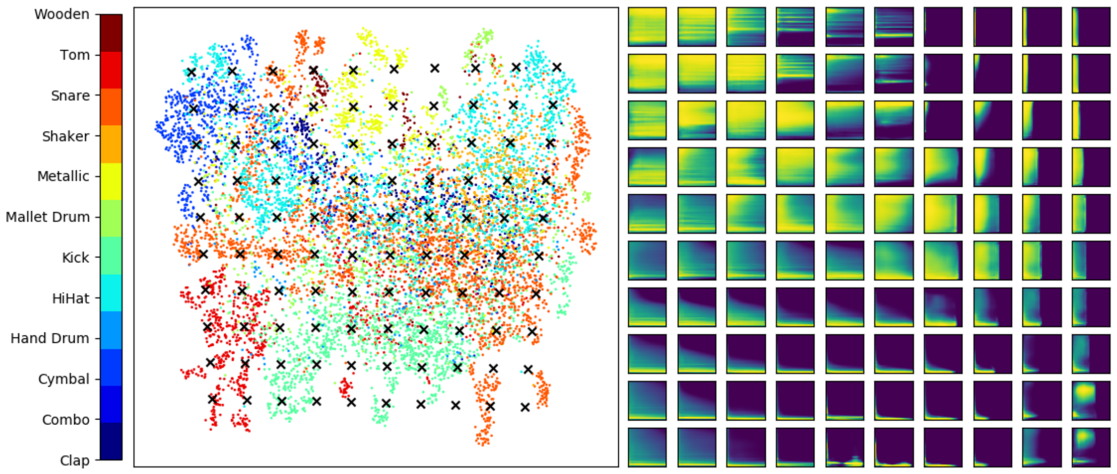


Figure B.2.: Generated spectrograms from a latent grid for the “64 dim” model.

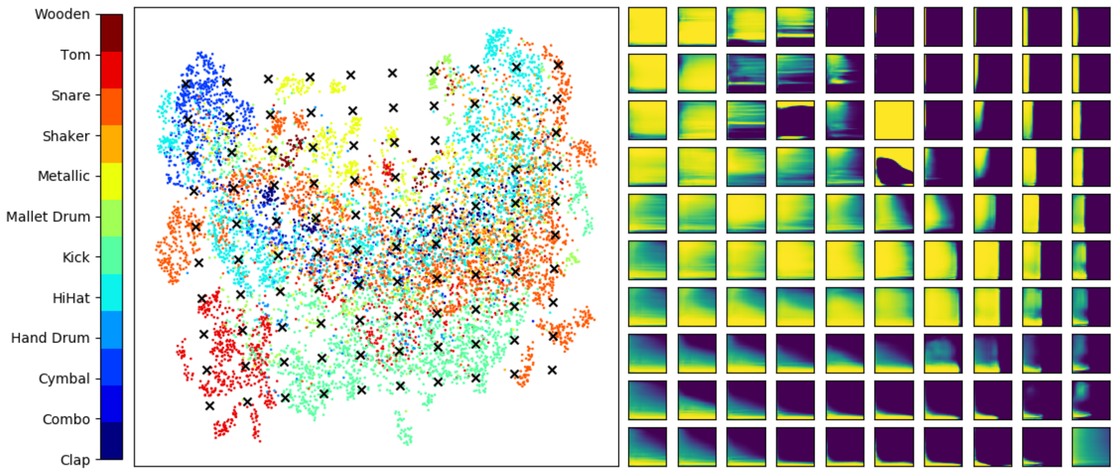


Figure B.3.: Generated spectrograms from a latent grid for the “256 dim” model.

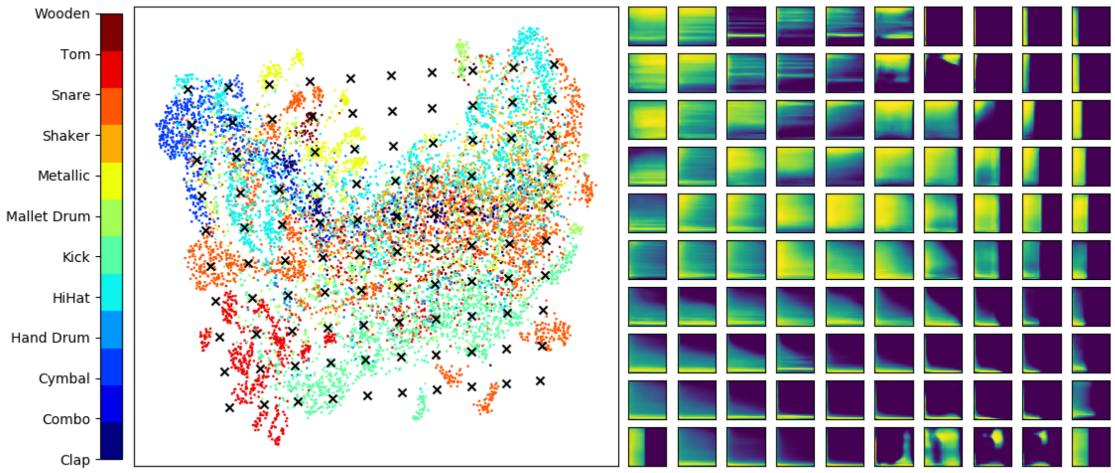


Figure B.4.: Generated spectrograms from a latent grid for the “0 flow” model.

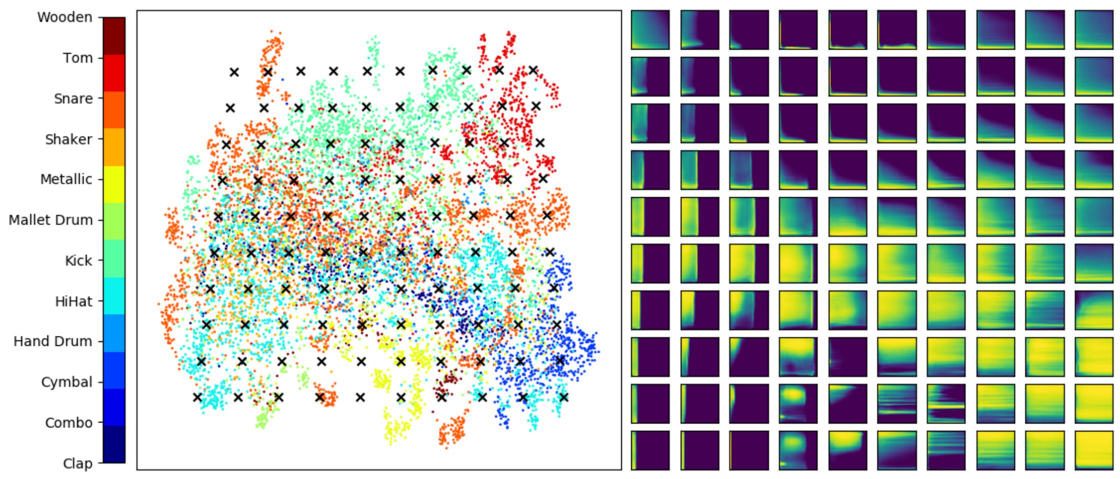


Figure B.5.: Generated spectrograms from a latent grid for the “20 flow” model.

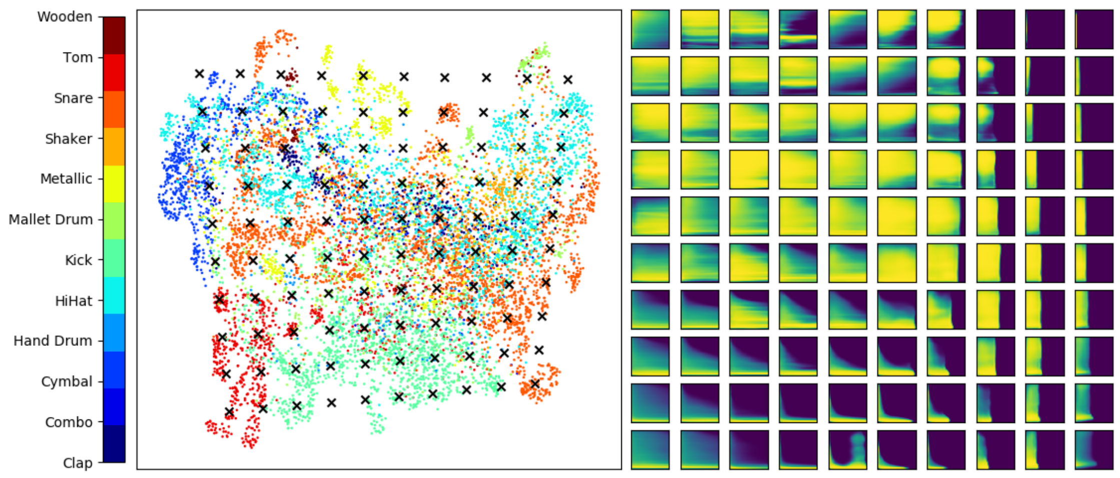


Figure B.6.: Generated spectrograms from a latent grid for the “No Dropout” model.

Eivind Aksnes Rebnord

Master's thesis

NTNU
Norwegian University of
Science and Technology