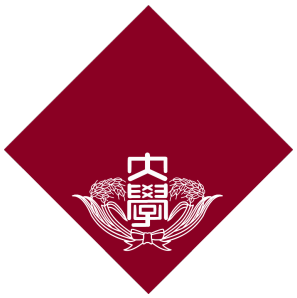
Drum Sample Synthesis with Variational Autoencoders



Ahmad Yunis Moussa

5119FG10

A Thesis Submitted to the Department of Computer Science and Communications Engineering, the Graduate School of Fundamental Science and Engineering of Waseda University in Partial Fulfillment of the Requirements for the Degree of Master of Engineering

Advisor: Prof. Watanabe Hiroshi

Research Guidance: Research on Audio-visual Information Processing

Submission Date: January 25th, 2021

List of Publications

- A. Moussa and H. Watanabe, "Audio Translation with Conditional Generative Adversarial Networks," 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Fukuoka, Japan, 2020, pp. 438-442, doi: 10.1109/ICAIIC48513.2020.9065067.
- A. Moussa and H. Watanabe, "Generation and Extraction of Color Palettes with Adversarial Variational Auto Encoders" 2021 6th International Congress on Information and Communication Technology (ICICT), accepted paper

Acknowledgements

Firstly, from the bottom of my heart, I want to thank my parents, without whom this thesis would not have seen the light of day. You have continuously encouraged me to pursue my ambitions, for which I am endlessly grateful.

Secondly, I would like to thank my advisor Professor Hiroshi Watanabe who has helped me on several occasions, for which I am very grateful.

Thirdly, I would probably not have reached this point in life without the help and encouragement I have received from my best friend Wiem.

Additionally, I want to thank my friends in Lebanon Mohammad, Abed and Rajab with whom I could always have a chat when I needed it, especially during these odd times. As well as the friendships that I have made here in Japan, Mehdi, Michael and Philip with whom I could always have interesting conversations.

Abstract

The task of synthesizing novel audio samples is a relevant and challenging problem, that requires a generative system to be aware of information beyond what is contained in floating point arrays that abstracts real audio, such as frequency and phase information. Modern generative neural network architectures can synthesize such realistic audio by training them on carefully constructed, large scale datasets, such that they learn the inherent characteristics of the data and become able to replicate by synthesis of novel samples. This replicative power generally is enabled with smart perceptual losses, powerful learned similarity metrics as well as the incredible generative power of multi-scale hierarchical models and hardware accelerated paralleled training procedures.

This thesis analyses the important components of modern neural networks for audio synthesis as well as other techniques that are required for the construction of specific datasets, specifically with neural classifiers and manifold data propagation methods. For the most part, techniques are discussed with a focus on short drum samples as they offer a challenging subset of sounds that is quite difficult to model. The thesis's main contribution harnesses the power of the variational autoencoder trained with perceptual losses to synthesize novel drum samples.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 On sound in the real world

Sound in the real world consists of intricate signals that are heavily loaded with humanly intelligible semantic information. From speech to music, the human ear is continuously consuming an incoming stream of different auditory signals which is decoded into illustrative information.

One can close their eyes and listen to the sounds propagating from outside the window. The brain can then know that there are certain events occurring, such as a car passing by, or a number of kids playing in some playground. But not only can we tell what events are occurring merely by listening, but one can also understand the particularities of these events, such as what type of car is passing by, approximately how fast it is passing by and equivalently how many children are currently rummaging around in the playground.

In addition to this objective information one can also attribute subjective connotations, for example, hearing the music blaring through the speakers of the passing car, one can decide whether that music sounds good or bad, and in turn one could also decide what type of person the listener is based on the choice of their music. This is an astounding level of inference that machines have yet to achieve.

For the human ear and brain these tasks are trivial and are accomplished multiple

times every single day. For intelligent systems these tasks have only become tractable very recently with the rise of trainable, hardware accelerated neural algorithms.

1.2 Problem Statement

Recent years have seen a surge in machine learning models devised for two main purposes: classification and generation of data samples. The purpose of this thesis is to offer a summary and analysis of these different methods with a special focus on synthesis of short audio waveforms. More specifically we explore the components that allow the composition of a dataset and the construction of a capable generative neural network model.

Synthesis and/or generation of audio is extremely useful in the current world of digital mediums such as music and video. Most Video has to be accompanied by audio, which is usually what makes the video feel 'alive'. Before the wide-spread use of modern computers and digital audio workstations, special artists were required to accompany and record the sounds of a movie in real time, they were called foley artists. Nowadays this can be done, to some degree by simply browsing a large bank of pre-recorded sounds and inserting them at the correct position alongside the video stream.

Recently, subscription based online marketplaces for these sounds are becoming increasingly popular amongst professional as well as independent music producers. These websites consist of large browsable databases of sounds that producers can utilize for the creation of a musical product. In most cases, a large amount of time is spent scouring these databases for the right sounds by artists and producers. Searching for specific sounds is primarily based on entering specific keywords and tags which will point to a certain subset of sounds. This prompts the exploration of models that are capable of the synthesis of such sounds and inspection of their inner workings.

1.3 Contributions

The contributions of this thesis are as follows:

1. An exploratory audio-data analysis with manifold learning techniques, as well as the proposal of an automatic label propagation that combines manifold learning algorithms with neural classifiers.
2. A comparison of neural classifiers for the multi-class classification of drum sounds.
3. An analysis of neural generative and discriminative models that have been utilized for the synthesis of audio signals in recent years.
4. A system that is capable of synthesizing short drum sound waveforms in the time domain, and learning a meaningful interpolatable latent space.

Part I

The Shape of Sound

Chapter 2

Digital Audio

Before delving into the specifics of machine learning models that are capable of generating realistic audio signals, it is beneficial to inspect how audio data is stored on modern computers, in addition to the different forms that audio data can be represented as.

2.1 Sampling Rate, Bit Depth and Nyquist Theorem

Sound in the real world is a vibration that travels through the air to the human ear. More specifically this vibration is a compression of particles in the space near and around the ear, which causes a thin stretch of skin inside our ear, the 'eardrum', to vibrate. This vibration triggers a signal to the brain which will be decoded into semantic information. To capture an abstraction of these vibrations we have designed microphones to operate in similar manner. A microphone generally consists of a diaphragm that emits an electrical current via transduction when it is met with a vibration from a sound wave. Intuitively, we can think about a microphone as taking 'snapshots' of the amplitude of a signal at each point in time. And to record sound in a faithful manner, these snapshots have to be taken at an incredibly high rate. The rate at which these snapshots are taken is what we refer to as the 'sample rate' of the ultimately recorded audio signal. Which is essentially it's temporal resolution. Higher sample rates allow more dense signals, that contain more information. Generally, the most popular sampling is 44.1khz (44100 samples

taken per second of audio), which historically was the original sample rate that was chosen for the Compact Disc (CD) [33], and is still used until today in modern audio workstation softwares. Different sampling rates are however possible. Another reason for the choice of this sampling rate as a standard is that it satisfies the Nyquist–Shannon sampling theorem [21], which specifies that the sampling rate of a signal has to twice as much as the highest frequency component contained in that signal to be able to accurately reconstruct it. Since human hearing generally only reaches 20khz, the 44.1Khz standard is completely sufficient. Another important parameter that dictates the quality of digital audio is bit depth. When sampling rate is the temporal resolution of an audio signal along the x axis, bit depth is the resolution of an audio signal along the y axis. A higher bit depth allows us to capture the sound with a higher fidelity as it essentially specifies the resolution of each individual sample taken, and how precisely we can place it on the y axis.

2.2 The Frequency Domain, Fourier Transform and Spectrograms

Now, we have established a compact digital representation for audio signals that is incredibly powerful. Real life sound can be stored as an array of floating point numbers and played back to us by interpreting it with two parameters: sampling rate and bit depth. However convenient this abstraction is, sound is not a simple as this. Audio typically signals consist of a combination of different overlapping vibrations at different intensities at different points in time, this makes an audio signal far from just a 1 dimensional abstraction. The analysis of the energies present in a signal could be incredibly helpful and indicative for downstream tasks such as classification of different sound events. However, it seems impossible to retrieve these different frequencies from a signal once it is stored as an sequence of numbers. It is similar to trying to reconstruct the different fruits that went into the process of making a smoothie.

The fourier transform [7] however makes this possible. It is an ingenious algorithm that allows the conversion between time domain and frequency domain:

$$X_{2\pi}(\omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-i\omega n} \tag{2.1}$$

The above formula describes the discrete-time fourier transform. Intuitively we can think of this formula as a machine that attempts to wrap a sequence of numbers (that is our audio signal), around the origin of the complex plane. Winding the sequence around the graph at different rates and simultaneously observing the center of mass, we can observe that at certain frequencies this center of mass will deviate significantly from the origin. These deviations are essentially the composite frequencies in our signal. Generally, for convenience and speed, we utilize the short term fourier transform (STFT) [13] rather than the regular fast fourier transform (FFT), which differs in that it is applied to consecutive slices of the input signal and not on the signal in it's entirety. This has the advantage that it allows us to visualize the change of frequencies with time.

$$\mathrm{STFT}\{x[n]\}(m,\omega) \equiv X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n} \tag{2.2}$$

A spectrogram thus, visualizes the intesity of a given frequency at any given point in time via the color component of the image. The brighter a certain spot the more of that specific frequency is present. Generally we represent time on the x axis, frequency on the y axis in log scale, and the color axis in Decibels (log scale of amplitude).

2.3 Mel Spectrogram

Studies have shown that the perception of human hearing is much closer to a logarithmic scale than it is to a linear scale [38, 40]. For example, we can barely tell the difference between the two frequencies 15000Hz and 15500Hz whereas it is very

easy to distinguish between 500Hz and 1000Hz, even though the distance is the same in both cases. The paper “A scale for the measurement of the psychological magnitude pitch” introduces a scale such that equal distances in pitch sound perceptually equally distant to the listener. This is done by applying a non-linear transformation of the frequency scale (there are several popular formulas for this purpose). One popular formula for this purpose is shown below:

$$m = 2595 \log_{10} \left(1 + \frac{f}{700}\right) \tag{2.3}$$

However, there are different variations of this formula.

2.4 Other Representations

There are a number of other spectrogram representations that are worth mentioning here, one of them being the Rainbowgram. The rainbowgram was first introduced in [6], where it represents "a CQT spectrogram with intensity of lines proportional to the log magnitude of the power spectrum and color given by the derivative of the phase" which results in a colorful and useful spectrogram representation suitable for the representation of musical notes.

Another heavily used representation are Mel Frequency Cepstral Coefficients [27], in short MFCCs. Cepstral coefficients, an anagram of the word spectral, are obtained by applying the DFT to the DFT of a signal. The obtained information shows the rate of change in the spectral bands, and is essentially a highly compressed version of a mel spectrogram. It is generally useful for audio classification and recognition task.

Chapter 3

Spectrogram Inversion Methods

We will also briefly touch on methods to invert back from the frequency domain to the time domain. This will become relevant later on throughout the thesis. Because of incredible advances in image synthesis with neural networks, a number of models have been devised for the synthesis of spectrogram images rather than the raw audio waveform. This comes with a caveat: reconstruction of the exact waveform might not always be possible depending on the chosen type of spectrogram representation. When creating magnitude spectrograms we are able to visualize the different frequencies that occur in a given signal, but with partial or complete loss of phase information. Phase information indicates how frequencies are aligned in a given signal, and are necessary for exact reconstruction. This is generally known as the phase reconstruction problem. A commonality with all currently existing algorithms that solve this inversion problem is that there is no one optimal solution.

3.1 Griffin-Lim Algorithm (GLA)

Generally, if we were to have the phase information we could simply utilize the inverse STFT to convert back to the time domain. However as we have already mentioned, the result of the STFT consists of two components: a real valued component and a complex component. The complex component represents phase information. In simpler terms, phase information indicates the overall alignment of frequencies with time. Phase information being complex valued makes it difficult to work with in modern neural network frameworks that can only deal with real numbers. This

amounts to discarding phase information in most cases and attempting to reconstruct it in some manner after the frequency component has been modeled by the neural network. One of the most popular algorithms for phase reconstruction is the Griffin-Lim algorithm [9]. It is an iterative procedure that continually converts between frequency and time domain until it converges towards some kind of phase information. The reconstruction quality is generally bad, and reconstruction speed is slow.

The Griffin-Lim algorithm begins by assuming a random phase component and applying the inverse STFT to the initial frequency component and therandom phase component. This ISTFT conversion results in a very bad signal in the time domain, however, due to the redundancy of the STFT and it's overlapping windowing function, if we apply the STFT on this signal we obtain a phase component that is better than the random phase which we started with. If we replace the frequency component with the initial frequencies and repeat this procedure of converting between frequency and time domain, we usually end up with a relatively useful phase component.

3.2 Neural Network Based Waveform Estimation

A number of neural network based approaches have been devised, not specifically for phase reconstruction, but estimation of the time domain signal from the magnitude spectrogram. It's a little bit different from phase reconstruction, since the resulting neural network will be very domain specific depending on the training data, but generally this approach is much faster, as computations can be done in parallel and it is not based on a iterative procedure, merely a single forward pass through the network.

These neural networks are essentially convolutional neural network that downsample, reshape and flatten an input spectrogram to a waveform. One interesting architecture, is based on applying multiple convolutional heads in parallel to the spectrogram for the waveform synthesis [2]. Such that each convolutional head observes the spectrogram in a slightly different manner, and they aggregate the final waveform. We

will encounter this type of multi-scale parallelism for waveform synthesis in more detail in later sections.

Part II

Related Work

Chapter 4

Generative Architectures

Modern neural networks have allowed the synthesis of sounds with an unimaginable quality in comparison to just a decade prior. Hardware acceleration with GPUs and TPUs have additionally permitted a fast paced advancement in this domain, considering that models are already deployed in live applications, and not just ongoing research projects. Before we dive into the specifics of our proposed method we should first examine related works that touch upon the same topic. We examine a number of related neural architectures, split into two sections. First we will have a look at modern generative neural architectures, that enable audio synthesis, followed by discriminative architectures (that usually make up the second half of GANs). We make the assumption that the reader is well acquainted with the inner workings of the fundamental building blocks of these architectures.

"Generative architectures" refer to a type of algorithm that is capable of creating some arbitrary type of data (digital audio in our case). The term "architecture" is generally adopted to describe the shape of the neural network that acts as the generative model, similarly to an architect who designs buildings on a blueprint, the modern machine learning engineer has full control over the number of parameters that will ultimately constitute the final neural network model. Recent years have seen a surge in different types of these generative neural network architectures, especially in the image processing domain, but also to some extent in the audio processing domain. Both fields share a great deal of similarities in that regard. Hence, in the following section we examine some important neural network architectures that have

emerged in recent years and their contributions towards the task of audio synthesis. Each of the following sections discusses a specific family of similar methodologies.

4.1 Auto-Regressive Models

Auto-regressive models [34, 36, 39] tackle generative modeling at the finest scale. The waveform is sequentially generated, one sample at a time, where each newly generated sample depends on all previously generated samples. In this manner a waveform can be represented as the product of the conditional probability of each sample:

$$p(\mathbf{x}) = \prod_{t=1}^{T} p\left(x_t \mid x_1, \ldots, x_{t-1}\right) \tag{4.1}$$

These sequential generative models have achieved state of the art results in several domains (refer to wavernn paper for citations here), yet their serial sampling significantly slows down the inference process, as the time resolution of audio is incerdibly large (thousands of samples per second). Much research has therefore gone into their speed up as well as the exploration of alternative strategies.

4.1.1 Wavenet

One of the first models that attempted the generation of raw audio with inarguably astonishing results was Wavenet [34]. The model is 'fully probabilistic and autoregressive' such that each predicted sample is conditioned on all previous samples. Wavent's architecture is intricately designed and was based by a preceding architecture called PixelCNN [35] for the image generation task, which generates images one pixel at a time.

Wavenet consists of multiple stacks of dilated causal convolutions. Causal convolutions are equivalent to masked convolutions in the image domain [26], which simply means that the kernel of the convolution does not have access to all of its inputs. A causal convolution however masks all future time steps to the input convolution,

this is done to prevent having the model access to the samples that it has yet to generate. Dilated convolutions [42] are convolutions that have a large stretched kernel that has the advantage of having a much larger receptive field, without observing all inputs that lay within it. Stacking these convolutions allows for an exponentially large receptive field, which is necessary for dense temporal data like audio, without increasing the number of operations required. The dilated convolution mechanism is re-used in several other papers that attempt the synthesis of audio.

4.1.2 SampleRNN

In a similar manner to wavenet, SampleRNN models raw audio in an autoregressive manner and generates one sample at a time during inference [24], but instead of utilizing stacked convolutional layers it does so by utilizing layered recurrent neural networks, which it calls modules. Each layer of SampleRNN performs at a different level of abstraction of the waveform: higher layer modules operate on large frames of the waveform, whereas layers down the stack operate on incrementally smaller frames, down to the sample level. In this regard high level modules create conditioning vectors for lower level modules. Like wavenet this hierarchical architecture works very well for capturing long term as well as short term temporal dependencies and structure of the waveform.

4.1.3 MelNet

Melnet also generates audio in an auto-regressive manner, but this time in the frequency domain rather than the time domain [39]. This is done by generating high resolution spectrograms in a similar manner to PixelCNN [35], but with a fully recurrent architecture rather than a convolutional network, which is due to spectrograms not being invariant to translation. Crudely, the generation of the spectrogram is done via two recurrent stacks: a time-delayed stack and a frequency delayed stack. These two stacks essentially allow the model to capture information on global scale as well as on a local scale. The time-delayed stack consists of three RNNs that make

passes over the spectrogram by observing consecutive frequency frames (a slice of the spectrogram) from left to right, top to bottom and bottom up, and their outputs will be combined and passed on to the frequency-delayed stack, which in turn generates the next frequency frames element by element and is conditioned on the output of the time-delayed stack. In this manner the frequency-delayed stack models local frequency details whereas the time-delayed stack assists the former by feeding it prior frequency information.

4.1.4 WaveRNN

WaveRNN improves the slow sampling speed of wavenet by exploiting hardware parallelism and proposes batched sample prediction via means of subscaling [14]. Subscaling essentially consists of re-ordering the waveform, by batching each B-th element of the waveform into B substensors. During the computation of a new sample, the model has then access to all samples in the current sub-tensor as well as all the already computed samples in the previous subtensors, which essentially creates a trade off between the causality of the model and it's parallelism.

4.2 Non-Autoregressive and Flow Based Models

Normalizing flows have recently made become a popular alternative to existing generative models, and can be considered as a different category of generative neural networks [31, 30, 36]. The goal of these models is the conversion of simple probability distributions into more complex distributions by means of invertible and differentiable mappings. In the case of audio, flow based models will generally synthesize sound from a simple distribution like white noise.

4.2.1 Parallel Wavenet

An alternative method to speeding up the original wavenet model without loss of quality. Parallel Wavenet attempts probability density distillation with a teacher-student framework [36], which simply means that a fully trained wavenet model will

be used to teach a simpler and faster model to generate results with similar quality. During training, the student network, which is a flow based model and can synthesize the waveform in parallel, is given as input a signal that consists of white noise and is tasked with the generation of the target signal. This generated waveform will then be passed to the teacher wavenet which is tasked with scoring it and signaling the student network how well it performed. We then iteratively improve the student network by reducing the Kullback leibler divergence between the distribution of the student network and the teacher network. Even though inference is sped up tremendously, training this type of model has to be done in two stages (individual training for the teacher and the student) which is incredibly time consuming.

4.2.2 WaveGlow and WaveFlow

WaveGlow and WaveFlow are another type of normalizing flow based neural networks that synthesize realistic speech, but do so without utilizing a teacher network and show that the auto-regressive flows are unnecessary [31, 30]. Both show that it is sufficient to train the flow based model with maximum likelihood to capture all modes of the data which makes the training procedure much simpler

4.3 GAN based Models

GANs [8] have seen incredible results in image synthesis and have also made their way to enhance audio synthesis. Training a Generative Adversarial Networks [9] demands two neural networks to be trained concurrently. Where the two networks consist of a generative model that attempts to generate new data by up-sampling a latent code x=Gen(z), and a discriminator that attempts to classify the generated samples as real or fake, such that y=Dis(x)[0,1]. This objective can be formalized as follows:

$$\mathcal{L}_{\text{GAN}} = \log(Dis(x)) + \log(1 - Dis(\text{Gen}(z))) \tag{4.2}$$

GANs train the discriminator and generator separately, such that the discriminator is trained to classify real and fake data each training step, and will be frozen during the gradient update of the generator network. Training GANs is notoriously difficult and unstable such that many modifications have been proposed for the stabilization of this procedure [32]. One famous problem of GANs is mode collapse, where the generator memorizes a number of examples from the training dataset, which it cycles through to fool the discriminator. The discriminator is essentially powerless and does not have the capacity to mitigate this problem as it is not explicitly trained to maximize the variety of generated samples. This is still an ongoing problem.

4.3.1 WaveGAN

One of the first papers to attempt unsupervised audio synthesis in an adversarial manner [5]. It adapts the famous DCGAN architecture to generate short audio clips of one second in length. They propose the generation of spectrograms with a model they call 'SpecGAN', as well as raw audio waveforms by flattening the DCGAN into a one dimensional equivalent architecture coined 'WaveGAN'. Another novelty of this paper is the introduction of the 'Phase Shuffling' technique which is implemented in the discriminator of their architecture. We will cover this in the discriminative models section.

4.3.2 MelGAN

Current state of the art techniques decompose the synthesis of human speech into two stages: synthesis of an intermediate representation, conversion of that intermediate representation into raw audio. Generally these intermediate representations consist of low resolution spectrogram representations. MelGAN's [19] generator focuses on the inversion of an input representation into raw audio. It does so by upsampling the representation through transposed convolutions and stacks of residual convolutional blocks. One important novelty of MelGAN is it's multi-scale discriminator architecture, which will be discussed in detail in a later section.

4.3.3 Parallel WaveGAN

Parallel WaveGAN, similarly to MelGAN is purposefully designed for speech synthesis from speech features [41]. Parallel WaveGAN trains it's wavenet generator via an adversarial objective and an auxiliary multi-scale STFT loss. This is again to sidestep the distillation process proposed in parallel wavenet. The novel component of this model is the loss function that is utilized and also follows a multi-scale scheme. We talk about this auxiliary STFT loss in detail in the next chapter.

4.3.4 HiFi-GAN

HiFiGAN, similarly to MelGAN attempts the generation of speech from an intermediate representation [17]. It does so by introducing a novel component called MRF module. The Multi-Receptive Field Fusion module improves on the MelGAN architecture by running the input through several residual blocks in parallel, each of which has different kernel sizes and dilation rates.

4.4 SING (Symbol-to-Instrument-Neural-Synthesizer)

SING tackles the synthesis of musical notes with a simple convolutional auto-encoder that is trained in two stages [3]. In the first stage the autoencoder is trained to reconstruct musical notes from NSynth dataset, then in the second stage we train a recurrent unit to imitate the latent code generated by the encoder when given a number of parameters that describe the encoded waveform. The RNN will be plugged in front of the decoder, in place of the encoder, which can be discarded. In this manner we can convert a number of high level symbols into a waveform. This shows that a carefully designed simple convolutional decoder can be powerful enough to generate high quality waveforms of short length waveforms, when trained with perceptual losses. This is impressive and shows that for the synthesis of certain types of audio signals, like musical notes, it might in fact be sufficient to utilize a simpler model. Because of these impressive results and the simple architecture, we

choose to adopt it's architecture for our own proposed model architecture, with a number of modifications.

Chapter 5

Loss Functions for Audio

5.1 Loss Functions

In this section we examine different types of loss functions that can be utilized to train models for the generation of waveforms. We briefly discuss why strict element-wise losses , like L1 and L2 loss, are quickly becoming obsolete in light of perceptual losses which are tailored to simulate human perception to some degree.

5.2 Element wise losses are not just bad but detrimental

Utilizing an element wise absolute distance metric as a loss function discards many aspects that constitute sound in the real world. Audio stored as an array of floating point numbers is an abstraction of real sound, and applying an element wise loss to this vector generally yields a loss that is sufficient for exact reconstruction, but is agnostic to other information such as frequency. For example, element wise losses penalize small shifts of the waveform tremendously, and are agnostic to presence or absence of specific frequencies. A convenient ablation study can be found in [19] on the effect of different loss functions.

5.3 Perceptual Losses

Since absolute distance metrics might be detrimental to the optimization problem, these problematic aspects of element wise losses led to the utilization of more intricate and perceptually informative losses in recent generative audio models. Ideally we would want to utilize a loss function that can mimic the way that the human ear can perceive sound in the real world. One way of doing so, would be the inspection of the spectral components of a generated waveform and comparing them to the target waveform and computing an error based on the difference of present spectral energy. We have already covered how to convert from the time domain to the frequency domain, in this section we cover a number of methods that attempt to extract an informative loss from the comparison of generated and target waveform.

5.3.1 Spectral Convergence Loss

This loss is used in [2, 41]. Intuitively this loss penalizes spectral components at a large scale due to the nature of the frobenius norm. This can be critical in early stages of training when the generated waveforms are still crude. This loss can be simply calculated as follows:

$$\| |\mathrm{STFT}(s)| - |\mathrm{STFT}(\hat{s})| \|_F / \| |\mathrm{STFT}(s)| \|_F \tag{5.1}$$

The exact origin of this loss's first application in machine learning is a bit difficult to pinpoint.

5.3.2 Log-scale STFT-magnitude Loss

Implemented in [3, 2]. Computing the difference of the spectral components on a log scale permits this loss to shift it's attention to small scale differences. This allows the carving out of fine details of the waveform in later stages of training. It's computation is as follows:

$$\| \log(|\operatorname{STFT}(s)| + \epsilon) - \log(|\operatorname{STFT}(\hat{s})| + \epsilon)\|_1 \tag{5.2}$$

Additionally, since the STFT is a parametric function, where the most critical parameters are the length of the FFT, the window size and the hop size of the window. Different values for these parameters will yield different spectrograms, albeit representing the same signal. Applying the Spectral convergence loss and the log-scale STFT-magnitude loss with different parameters for the STFT allows us to construct a multi scale loss that observes and compares the frequency components in the generated and target waveform at different scales [41], which might overall enhance the quality of the generated waveform.

5.4 Problems with generated waveforms

When we closely inspect images generated by neural networks, we often see repeating patterns that occur on a pixel level of the image, such that every other pixel has brighter and dimmer color values, similar to the pattern we see on a checkerboard. These artifacts arise from the nature of the operations in a deconvolutional neural network that employs transposed convolutions. These artifacts are not desirable in digital images and several solutions have been designed to alleviate this problem (need citations here). Discriminators in GANs can generally reject images presenting these artifacts, as they do not occur in regular images. When generating audio, these artifacts appear as recurring periodicities that manifest themselves as pitched noise [5], that can theoretically occur in real audio recordings. The discriminator might not be able to single out such frequencies and discriminate against, or alternatively might be able to notice that all generated fake waveforms do present a specific high pitched frequency, rendering the discriminators task trivial. Which in turn leads to a GAN failure mode.

Chapter 6

Discriminative Architectures

In addition to perceptual losses, and in light of the problems of generated waveforms with deconvolutional networks, a number of different discriminative architectures have been developed for the improved discrimination of generated waveforms in GAN based models [5, 19, 17].

6.1 Phase Shuffle

Because of the occurrence of pitched noise artifacts in generated waveforms, the discriminator can learn a trivial policy to reject them, by simply observing the presence of said noise. This inhibits the overall training of the GAN, as the discriminator stops yielding an informative criteria to the generator. WaveGAN proposes to solve this by implementing a 'phase shuffling' operation in the discriminator [5], which simply randomly shifts the entire output feature map in the intermediate layers of the discriminator, and pads them with reflection padding. This essentially manifests as a perturbation to the phase of the input signal, such that discriminator essentially starts ignoring phase information and finds other informative and discriminative features. This naturally makes the discriminator's task more difficult, but encourages it to give the same result for the same waveform even if it has a different phase (even if it is shifted by some amount of samples).

6.2 Multi-Scale Discriminator

Rather than utilising one discriminator that is fed the entire generated waveform, we utilize multiple discriminators each of which accepts a slightly different version of the same waveform [19]. For example, if we were to utilize 3 discriminators, the second and third discriminator could be fed downsampled and smoothed versions of the generated waveform. Which makes it such that these discriminators only have access to low frequency components of the waveform and are biased to learn discriminative features based only on these frequencies. This encourages an inductive bias and pushes the discriminators to learn features beyond what is present in the target data.

6.3 Multi-Period Discriminator

Similarly to the multi-scale discriminator scheme, HiFi-GAN devises a group of discriminators to inspect the waveform in different ways [17]. We construct these discriminators such that they accept as input arrays of equally spaced samples from the input waveform at a given period p. This is achieved by reshaping the input waveform (a 1 dimensional array), and cutting it up into p sized chunks. We take each chunk, transpose it and stack them into a 2dimensional tensor, with dimension p x L/3 where L is the original length of the waveform. The discriminator in this case will consist of 2 dimensional convolutional layers, and convolutional kernels will be restricted to a width of 1 such that they can inspect each pattern by itself. This again encourages an inductive bias to detect different periodic patterns in the input waveform. Conveniently HiFiGAN makes an ablation study on the effectiveness of MSD and MPD.

6.4 Markovian Discriminator and Window Based Objective

Instead of outputting a binary value, we truncate the discriminator's tail and output a matrix of probabilities. In the case of images, such a discriminator would classify NxN patches of the input image as real or fake, and would be run convolutional across the image [12, 19]. This essentially models the image as a random markov field. In the case of audio this NxN patch becomes a slice of length N and has been shown to be equally successful in the audio domain (melnet). A regular discriminator would learn to distinguish between distributions of complete waveforms, the patch discriminator would learn to distinguish between small chunks of the waveform, additionally we can construct the discriminator in such a way that it observes large, overlapping chunks of the waveform and in that way observes coherency and high frequency structure over the waveform.

6.5 Feature Matching Loss

This technique is introduced as a 'regularizing objective' for the generator [32]. Rather than training the generator to fool the discriminator in a binary manner, we train the generator to produce data samples that induce discriminator activations that match the activations of the discriminator for real data samples.

Part III

Dataset

Chapter 7

Audio Classifiers

For our proposed label propagation technique we require a separate classification model that can run in the background. Depending on the type of representation we choose for the audio data, we can approach this classification task in several different ways:

1. **Raw Audio:** training a convolutional neural network from scratch or fine tuning PANNs (pre-trained audio neural networks) [18].
2. **Spectrogram Representation:** treating the spectrogram as an image we can fine tune state of the art pre-trained image classification networks such as ResNeSt50 [10].
3. **MFCC:** for compressed representations we can utilize traditional classification methods and are maybe the go-to feature representation when it comes to audio classification.

The purpose of this classifier is to train it on a small dataset that is annotated by hand and make it capable of generalizing to new examples with a high accuracy such that we can annotate the rest of the data dump that we have collected. Which also makes it important to choose a number of different classes that we will split our data into. Most audio samples that we collected have already been named after the specific drum sound that they represent, which allows us to create a small dataset that constitutes of 8 distinct classes:

1. Kick
2. Snare
3. Hat
4. Crash
5. Ride
6. Clap
7. Shaker
8. Tom

In this sense, the classifier will output a one hot vector that indicates the type of the sound that we are dealing with making it a multi-class classification task. The problem of classifying drum sounds specifically has been studied in depth in a number of publications (cite paper) on which we also base our chosen classes on.

7.1 Baseline Models

We propose three simple models as base line that we train from scratch:

1. Convolutional Neural Network that accepts the raw waveform
2. CNN that accepts the spectrogram representation
3. CNN that accepts both

7.2 Transfer Learning

Transfer learning is reusing a neural network that has been trained on some task for a new similar task [29]. For example, if we're creating a classifier to recognize different types of vehicles, it might be beneficial to utilize a neural network that has already been trained to recognize various real world objects to a certain degree of accuracy, and train it for the new task. In this manner, a pre-trained neural network

already has some understanding of different types objects and can generalize easier to the new task rather than training it from scratch. In many cases the new task is less difficult than the original task. Popular neural network architectures in this regard are pre-trained imagenets and resnets, trained on the imagenet dataset [4] containing 1000 different classes. To apply transfer learning, we generally append a new trainable layer to the pre-trained neural network because most often the output dimensions and specifications differ from the original task. Here we have the choice of freezing the original weights of the pretrained model or allowing their gradient updates. Choosing to freeze those weights, the only trainable weights become those of the appended layer/s. This is called fine-tuning a neural network and in many cases works very well. Since Audio signals can be represented in a plethora of ways, this opens up many different neural networks that we can utilize for transfer learning. We list and evaluate some of the architectures in this section.

7.2.1 Pre-trained Image Classifiers

Repurposing image classification models for audio classification tasks is not an entirely new concept, but hasn't been studied in depth yet. One recent paper explores this venue: "Rethinking CNN models for audio classification". Thus, since audio signals can be represented as spectrogram images, these image classifiers are a perfectly viable method for their classification [28]. Which at first thought might be problematic as these models are trained on natural images where the main purpose is the detection, localization and classification of specific objects in the image whereas they are being applied to a completely different type of representation. However, it has been shown that these pretrained models do generalize well to spectrograms, as they can recognize and locate specific frequency slopes and shapes that indicate the occurrence of some audio events. We explore how well this can be applied to the classification of drum sounds.

7.2.2 Pre-trained ResNeSt50 (ResNet architecture with split attention)

Attention mechanisms have become increasingly popular, leading to tremendous breakthroughs in the NLP field most notably utilized within the transformer architecture. Attention mechanisms have also made their way into other fields like computer vision such that already powerful models like Resnet have been redesigned with these mechanisms incorporated. We also apply one such model to the drum sound classification task [43].

7.2.3 PANN (Pre-Trained Audio Neural Networks)

PANNs are also CNNs that have been specifically pre-trained on audio tasks and are an equivalently strong contender to be used for new audio classification tasks [18]. One small modification that we apply is the reduction of dropout rate in the last embedding layer from 0.5 to 0.2. Since our dataset is already quite small we do not require any additional amount of regularization, and reducing the dropout rate allows for better performance.

7.3 Comparison

For this comparative experiment we create a small handmade dataset consisting of roughly 4400 drum sounds split evenly among the aforementioned number of classes. We also set apart a small test set of 160 drum sounds (20 for each class). The following table summarizes the results.

Model	Accuracy	Precision	Recall	F1-Score
CNN Raw	0.52	0.47	0.52	0.49
CNN Spec	0.61	0.73	0.61	0.59
CNN Raw + Spec	0.74	0.74	0.74	0.74
ResNet50	0.79	0.80	0.79	0.78
ResNeSt50	0.86	0.86	0.86	0.86
PANN CNN14	0.86	0.87	0.86	0.86

Table 7.1: We compare the performance of different types of classifiers on the drum sounds recognition task. We can observe that the pre-trained models perform much better in general.

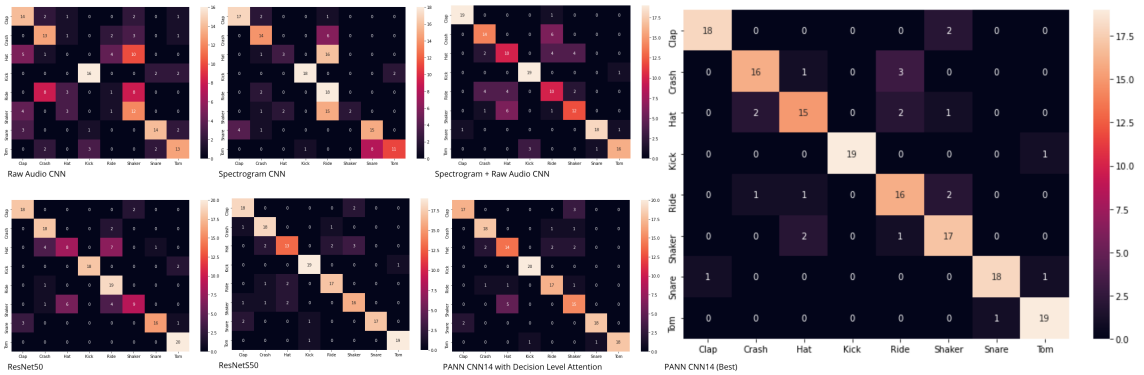


Figure 7.1: Confusion matrices of the different classification models. Pretrained models are much more precise than other alternatives.

Chapter 8

Manifold Learning for Audio Data

8.1 Assisted Data Annotation Method

Before we delve into the training specifications of our neural network, we describe how we construct our dataset and we propose a set of steps for a semi supervised and automatic label propagation technique to new data samples. The purpose of this algorithm is the annotation of the unlabeled data samples in the large data dump that we scraped from the internet, from a smaller subset of already annotated data. The entire dataset consists of 44000 data samples, of which only one tenth is annotated by hand. Large amounts of good quality training data are a cornerstone of modern deep learning architectures and are becoming more and more necessary to train large scale neural models. This is why AI assisted data annotation and exploration tools can be extremely powerful. And in our case, the objective task revolves around constructing a meaningful latent space and a strong decoder that can interpret latent codes into raw audio. In the next couple of subsections we detail the necessary techniques involved that build the backbone of our algorithm.

8.2 Manifold Learning

Manifold learning can be described as non-linear dimensionality reduction that projects high dimensional data onto some lower dimensional surface (a manifold). Generally it is applied to visualize and represent datasets that consist of high dimensional

data samples in a lower dimensional space. Dimensionality in this case, refers to the number of attributes (variables/coordinates) that are required to represent a certain kind of data. For example, as discussed earlier, audio is stored as an array of floating point numbers on disk, and depending on the sampling rate and the length of this audio file, we can determine its dimensionality. However, it is impossible to represent said audio file on a plot, we can hardly imagine things in 4 dimensions. Often though, this high dimensionality happens to be very redundant, where the set of defining features is generally much smaller than the overall dimensionality. A manifold learning algorithm solves this by representing our high dimensional data sample simply by a 2D or 3D point on a plot. Additionally, it reflects and captures the differences and the structure that arises in our high dimensional dataset by distance on the low dimensional point plot. More simply, the more different two items in the dataset are, the further apart they will be on the point plot, and vice versa, the more similar they are, the closer they will end up. How this high dimensional similarity and likeness is determined depends on the algorithm utilized as well as the features that we compute it on, in addition to how well the method is at preserving local and global structure. We inspect two such algorithms in the remainder of this section: t-distributed Stochastic Neighbour Embedding and Uniform Manifold Approximation and Projection

8.3 t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE [37] operates by creating two distributions, one over the high dimensional dataset and another in a lower dimension and then attempts to make both a close as possible. Constructing these probability distributions is done by converting the Euclidian distances between high dimensional data samples into conditional probabilities. In simpler terms, we span a distribution over each data sample and find all samples that fall under this distribution In comparison to other techniques such as PCA, t-SNE greatly preserves local structure, and can also incidentally learn some

of the global structure of the data, however without any guarantee that it represents the actual global structure in the real data.

8.4 Uniform Manifold Approximation and Projection (UMAP)

UMAP [23] can be considered as t-SNE's more powerful successor. And similarly to t-SNE, it arranges high dimensional data on a low dimensional graph and is generally preferred over t-SNE due to it's faster computation speed with larger datasets. It is also speculated to retain more global structure. UMAP in comparison to t-SNE also allows control over the shape of the graph with several parameters as opposed to the single perplexity parameter in t-SNE. Additionally, UMAP also accepts labels and can perform semi-supervised clustering of samples.

8.5 Manifold Comparison on Audio Dataset

Both t-SNE and UMAP are equivalently viable for our purposes (generally we prefer the faster implementation). The shape of the constructed neighbour graph is important for the remainder of this shape. We combine the previously discussed neural network classifiers with manifold learning techniques for the automatic propagation of annotations and assisted annotation on partially annotated datasets. By creating the UMAP of a dataset of partially labeled points we can observe the shape of the dataset from a low dimensional point of view. When done for audio files, we can hear that the closeby data points are generally grouped by similar perceptual qualities. In our case, kick drum sounds will generally be grouped tightly together and slowly transition into a different class. Naturally some overlap can occur in many cases when it is not very clear what type of sound it is. Even human listeners might struggle to classify such data well. Since neighbouring points are perceptually similar, it could be useful to use the labels of nearby labeled points for the labeling of unlabeled points. In this manner we would aggregate the labels of a number of nearby points and obtain a tentative classification. In the previous section we have

already discussed modern neural network classifiers and achieved a top accuracy of 87% on our small dataset, which will be very difficult to surpass. The essential difference between this section and the previous, is that we are using the topology of the neighbour graph to infer information about a data sample, rather than classifying the data sample itself.

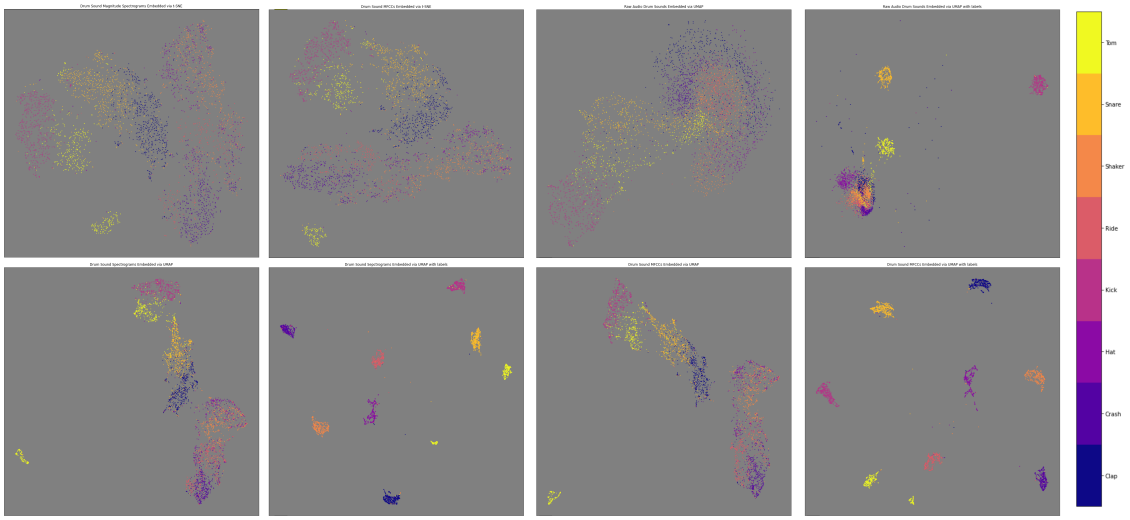


Figure 8.1: Comparison of Neighbour Graphs of our audio dataset constructed with UMAP and t-SNE for different feature representations.

8.6 Automatic Label Propagation

Our approach of propagating the labels to unlabeled examples consists of querying a point from the neighbour graph, querying the k nearest points with a KDTree, finding the already labeled points amongst them, summing the result and applying a softmax to normalize the result. This opens up a number of questions for experimentation:

1. **What ratio of labeled to unlabeled data works best:** The more neighboring labeled data is around a data point, the better the propagated label should be.

2. **Which graph has the best topology for this purpose?:** UMAP or TSNE? 2D or 3D? Which features should you compute it on?

3. **Can we use unlabeled points for inference?:** Since we have a very strong

classifier, maybe we can increase the performance further by running it on nearby unlabeled points and also factoring them into the result. In this case, what weight should be assigned to ground truth and predicted labels?

Also mention proximity bubble.

Neighbours	3					6					12				
Split Ratio	0.5	0.6	0.7	0.8	0.9	0.5	0.6	0.7	0.8	0.9	0.5	0.6	0.7	0.8	0.9
MFCC	0.1451	0.1560	0.1424	0.1401	0.1218	0.1261	0.1514	0.1281	0.1412	0.1376	0.1234	0.1175	0.1273	0.1231	0.1241
Spec	0.1383	0.1384	0.1371	0.1355	0.1444	0.1352	0.1232	0.1507	0.1299	0.1173	0.1334	0.1237	0.1122	0.1299	0.1489

Table 8.1: Results for the proposed data propagation method with different data splits and different number of considered neighbours. Overall the numbers indicate that the method does not work very well, regardless of the parameter configuration used. It also shows that the ratio of labeled to unlabeled data does have an effect and that less extreme splits work better overall.

We conduct a number of experiments, where we split the dataset with different ratios and then compute the UMAP based on MFCCs and Spectrograms. Then for each unlabeled point we fetch the k nearest labeled neighbours and average their labels. As the table shows this naive method yields more than disappointing results. This might be due to the fact that some classes are very intertwined in low dimensional space. For example, if we refer to the earlier UMAP visualizations (where we didn't provide labels) we can see that hats and shakers always have overlapping areas. Maybe better results could be achieved if we utilize a dimensionality reduction technique that achieves a better inter class separation. One interesting observation though is that the ratio of labeled to unlabeled points affects the propagation accuracy.

Neighbours	3					6					12				
Split Ratio	0.5	0.6	0.7	0.8	0.9	0.5	0.6	0.7	0.8	0.9	0.5	0.6	0.7	0.8	0.9
Spec	0.1307	0.1209	0.1371	0.1276	0.1557	0.1320	0.126	0.1454	0.1638	0.1286	0.1279	0.1147	0.125	0.1288	0.1444

Table 8.2: Results for the proposed data propagation method with a backend classification model. Overall the performance is not improved and neighbour information seems to generally not be very useful for the classification of new points. In this experiment we only use spectrogram features as the ResNeSt50 can not be used on MFCCs.

We perform another experiment where we split the dataset by a certain ratio into labeled points and unlabeled points and train the ResNeSt50 classifier on the labeled data points. In this manner, for each unlabeled data point we fetch the nearby labeled

data points as well as the unlabeled ones, then run the classifier on the unlabeled points and average all results. This does not improve the results and is most likely due to the aforementioned problems. Overall this indicates that we are better off by simply using the classification model to annotate new data samples and that the topology of the neighbor graph has a drastic effect on the accuracy of propagated labels. Maybe there is something else that needs to be considered to improve and make this method succesful.

Part IV

Proposed Method

Chapter 9

Proposed Neural Architecture

9.1 Variational Auto Encoders - VAEs

In this section we describe our proposed neural network architecture in detail. The main difference in our model in comparison to other research that has gone into the synthesis of drum samples, is that current approaches generally attempt to model spectrograms rather raw audio [1, 25]. Modeling spectrograms entails a waveform estimation procedure which we side-step by directly generating the waveform.

9.1.1 VAE Objective

Our proposed architecture is essentially a Variational Auto Encoder [15]. Regular autoencoders generally compress data into a lower dimensional representation in an unsupervised manner. This compression is achieved with an encoding neural network that transforms the input data into a latent representation. Meaningful latent codes are learnt by iteratively compressing data then reconstructing the original data with a decoding architecture (which accepts the latent code as an input). Albeit, actual data compression often not being the main goal but rather the reduction of the dimensions of the input space.

Conventional auto-encoders do not guarantee the construction of a meaningful and interpolatable latent space. This means that decoding a randomly created latent vector will most likely produce a garbage data sample. This is the key difference

between Auto-Encoders and Variational Auto Encoders. VAEs attempt to learn the input data space as a probability distribution rather than a mapping into a lower dimension. Therefore VAEs are trained differently from regular auto-encoders, with an additional regularization term that enforces structure in the latent space beside the reconstruction term. The regularization term attempts to cluster similar encoded data samples into neighbouring regions in the latent sapce. In this manner, the VAE's decoder guarantees that any randomly sampled latent vector will produce a meaningful data sample. In this regard, VAEs have been proven to be powerful generative deep learning models for the construction of meaningful and rich interpolatable latent spaces.

More pragmatically, the input to the encoder can be nominated as x, the intermediate latent vector by z and it's weights and biases by θ. The encoder can then be represented by $q_\theta(z|x)$ and the decoder by $p_\phi(x|z)$, where it's input is the latent vector z with weights and biases ϕ. To measure how accurately the decoder has reconstructed the expected output we utilize the log-likelihood $logp\phi$ as well as the regularizing term that specifies how much information was lost between $q_\theta(z|x)$ and p(z). This loss function is called the evidence lower bound function in variational inference and can be formalized as follows:

$$- E_{z\sim q_\theta(z|x)} \left[\log p_\phi(x \mid z)\right] + KL\left(q_\theta(z \mid x) \| p(z)\right) \tag{9.1}$$

Due to the sampling operation in the latent dimension, obtaining the gradient through the ELBO is not feasible. This can be side-stepped by using a re-parametrization trick where p(z) is a standard normal distribution with mean zero and variance one:

$$\epsilon \sim \mathcal{N}(0, I), z = \mu + \sigma \odot \epsilon \tag{9.2}$$

Much research has gone into the development of variations and different versions

of the ELBO loss function [16], however for the purposes of our model we find the vanilla one sufficient.

9.1.2 Proposed VAE Architecture

We experiment with different encoding and decoding architectures, and ultimately settle for an architecture that is based on the SING auto-encoder architecture. We have already discussed the SING architecture earlier. The quality of SING's generated samples is astounding, for the relatively simple architecture and decide to implement the model with a number of modifications. We insert a sampling layer in between the encoder and decoder such that it becomes a VAE. Additionally, to train the model, we adapt the multi-scale STFT loss as proposed in Parallel WaveGAN, which is comprised of three differently parametrized STFT operations. However we do not use the Spectral Convergence, solely the formula suggested in the SING paper. The keras framework makes it easy to implement, as the STFT function has to be fully differentiable for the gradient to travel backwards and update the weights of the VAE. Naturally, to train the variational auto-encoder we also require a regularization term which is the Kullback Leibler Divergence. The overall objective function can be observed in (reference formula here). The overall detailed architecture can be observed in figure 9.1.

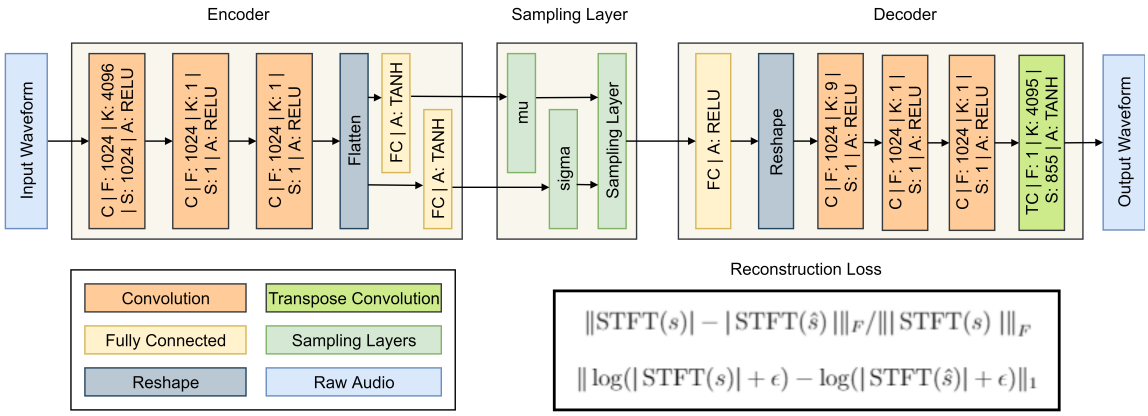


Figure 9.1: The overall Architectur

9.1.3 VAE Results

1. **Random Sampling:** we sample random vectors from a gaussian distribution and run them through the decoder to observe the generated result.
2. **Interpolating Latent Vectors:** we sample two random vectors from a gaussian distributions and create intermediary vectors via linear interpolation.
3. **Reconstruction of Waveforms:** we observe the reconstruction quality of encoded samples.

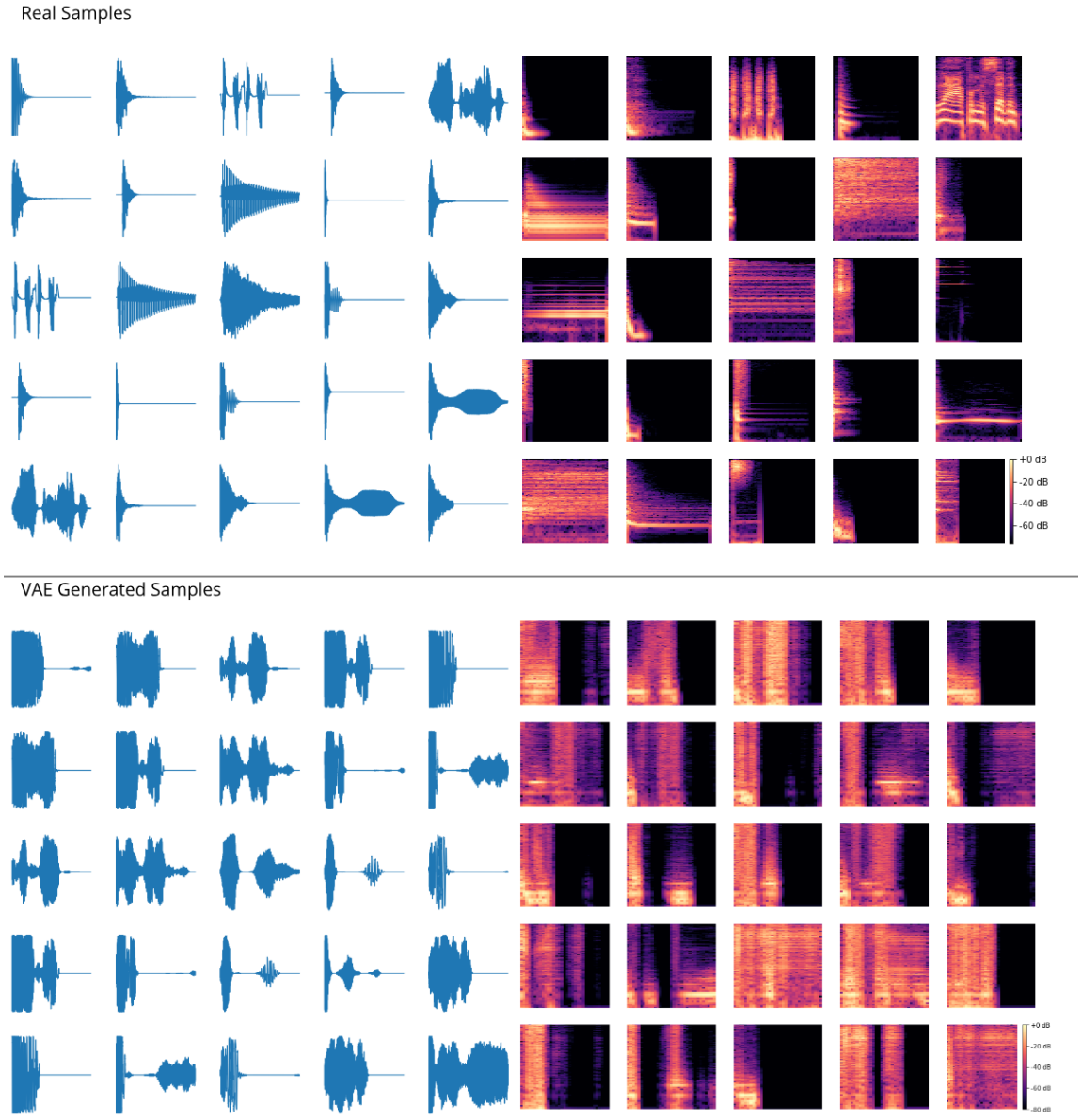


Figure 9.2: Real Samples vs Samples generated from the VAE. Generally, it seems that the sounds the VAE generates are different interpolations between the training samples but lack in clarity and detail, and are noisier than real samples when we observe the spectrogram representations. However, the STFT loss enables the model to capture important high and low frequency components.

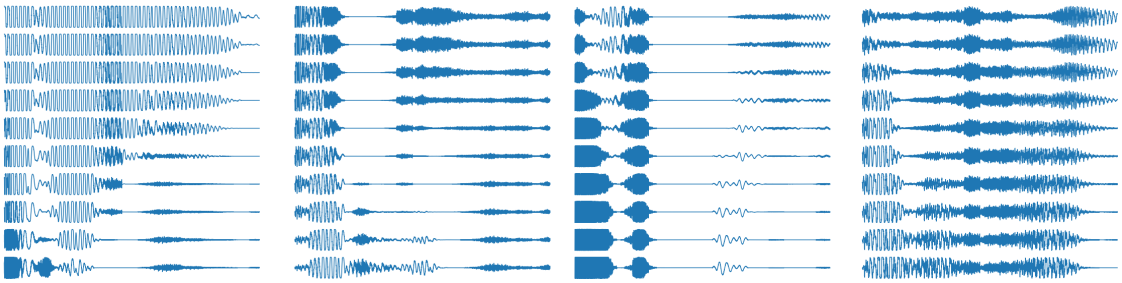


Figure 9.3: Interpolations generated by the VAE model. Generally the interpolations are smooth.

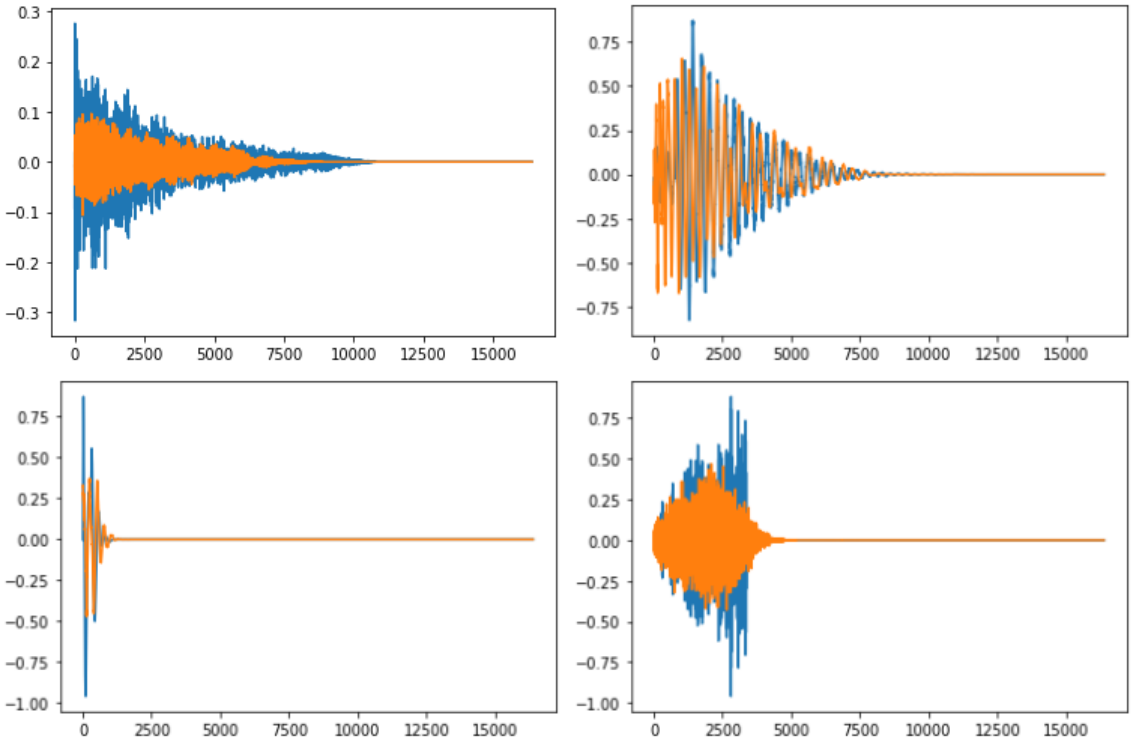


Figure 9.4: Reconstructed drum hits. We suspect that blanacing the reconstruction vs regularization term has a significant effect on the tradeoff between quality of the samples and the smoothness of the latent space. Maybe some modification to the ELBO loss function could yield better results.

9.2 VAE + GAN experiment

9.2.1 GAN Objective

Initially we tried to utilize a model and training strategy similar to the one proposed in [20]. However we have failed to reach convergence with that architecture on our dataset. We assume that this is due to the discriminator not being able to discern between the initial noise that is generated by the generator from the drum samples in the dataset. The discriminator quickly collapses to zero loss and the GAN is in failure mode. To alleviate this we propose the use of an auxiliary discriminator, however, it is unclear if it is actually contributing to the improvement of the overall model.

In our architecture we create a generator that shares weights with the decoder of the variational autoencoder and which can be updated by the discriminator. We thus train the GAN and VAE in an alternating manner and present the results.

9.2.2 GAN enhanced Results

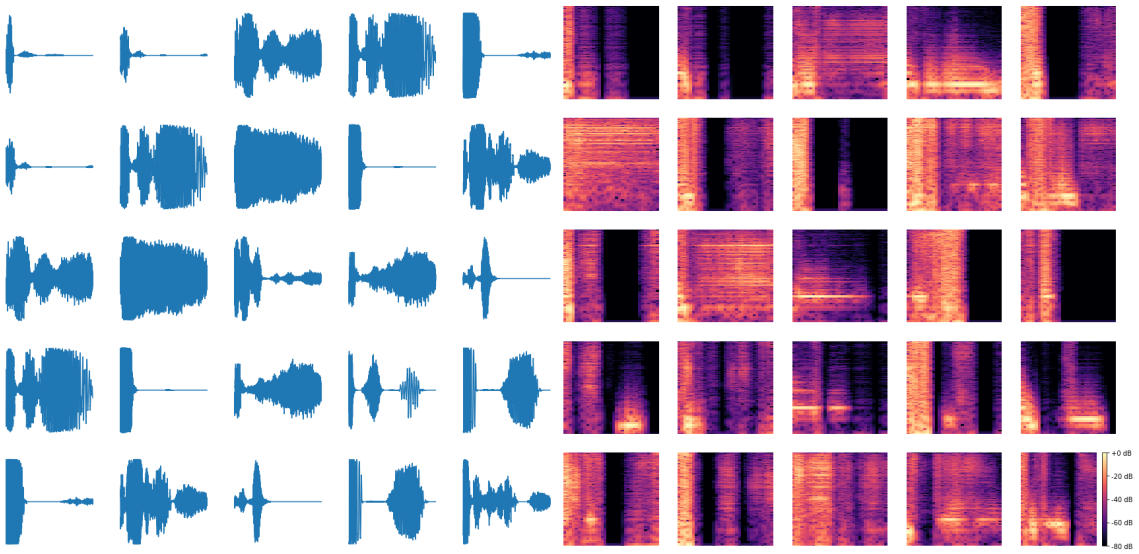


Figure 9.5: Samples generated from the VAE GAN. The samples look quite different from the simple VAE but audibly we don't find much imporvement or difference

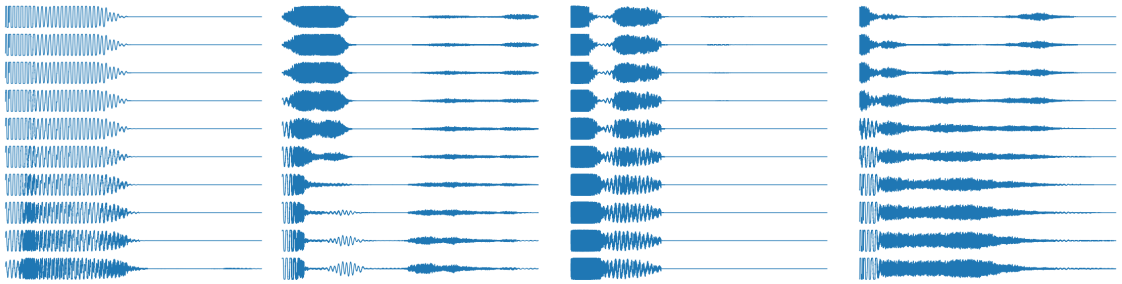


Figure 9.6: Interpolations generated by the VAE GAN model.

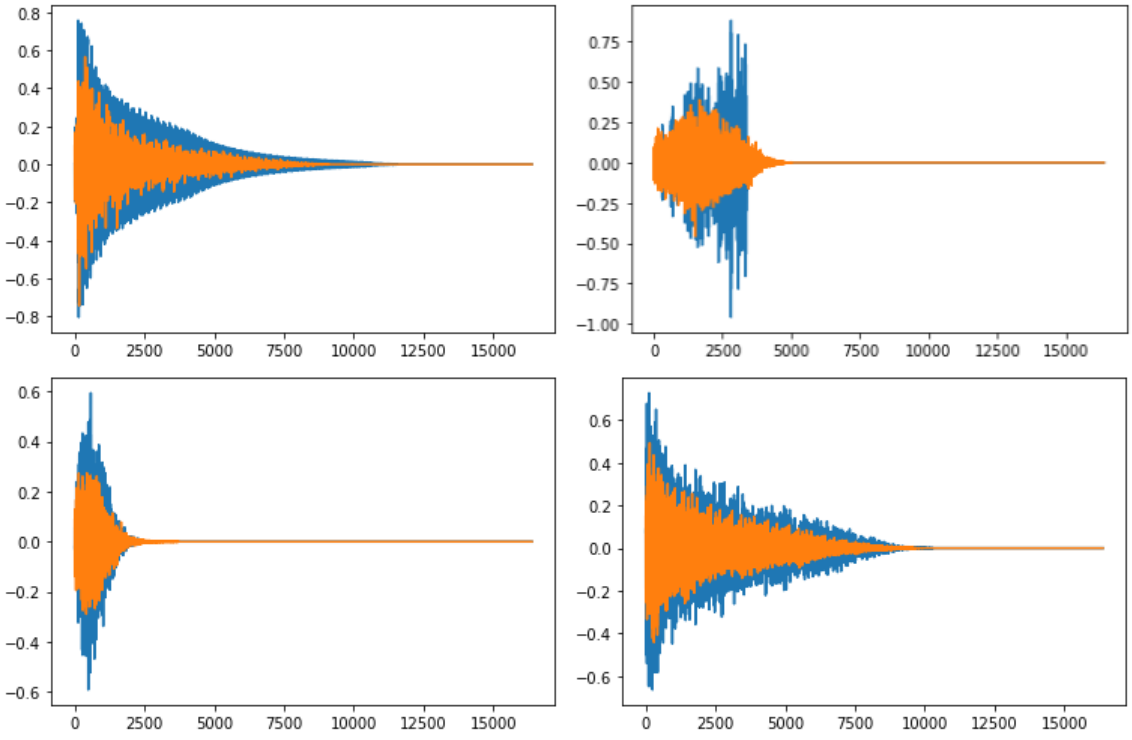


Figure 9.7: Reconstructed drum hits with the VAE GAN model.

9.3 Discussion and Observations

Training GANs is notoriously difficult, and we run into several difficulties. We found that RMSprop shows the best results the soonest into a training session. We used a learning rate of 4e-4 for the generator and discriminator and utilize the least squares gan objective [22]. The largest difficulty we encountered was finding a set up of parameters that actually gave reasonable results early on during training (we don't see any significant difference utilizing [11]). Nonetheless, we fail to obtain convergence on the regular VAEGAN model. The reason for this could be that for the STFT loss to succeed in the GAN setting, there has to be a strong conditioning factor like the mel features in TTS systems [19, 41, 17]. Generally in text to speech systems we condition on these features and try to construct realistic voice, in this manner it is reasonable to utilize the STFT loss, as we are guiding the network to learn a clear 1-to-1 mapping from conditioning feature to raw audio. However, in the case drum sounds that might be very noisy at times, it is not clear if the STFT can contribute when used as auxiliary loss alongside the discriminator. In most cases, we end up with mode collapse when we utilize the STFT loss as an auxiliary loss.

Part V

Conclusion

Chapter 10

Conclusions

In this thesis we present an extensive introduction and analysis of modern techniques that can be utilized for audio processing in a machine learning context. These techniques range from different feature representations, to audio classification, as well as manifold learning and dimensionality reduction of audio samples for representing them on a 2D graph. Finally we conclude with a neural network model that allows the synthesis of novel sounds. We apply these methods with a special focus on drum sounds, as they represent a challenging subset of real world sounds.

Two important deductions that we can draw from the analysis of related work, is that multi-scale architectures are becoming increasingly more popular. Examining the frequency components of a waveform as a spectrogram representation at different scales allows the inspection of long and short term features, which is crucial for classification and also for the synthesis of novel sounds. Modern models therefore adapt hierarchical and multi-scale architectures for the synthesis of raw waveforms as well as spectrogram representations. Synthesis in the frequency domain and inversion to the time domain is also not a problem anymore with neural network powered inversion systems, which opens up many possibilities for the creative synthesis of spectrograms.

The development of these powerful methods also demands the construction of meaningful and large datasets that cover multiple modes and aggregate different types of sounds. The development of tools that allow the construction of such a large

scale dataset is therefore crucial for the advancement of future endeavors in audio synthesis. We believe that manifold learning techniques, in combination with intelligent classifiers as well as human judgement could be an intelligent step in the right direction.

As far as the synthesis of drum samples, it is a relevant and challenging starting point to test the performance of different methods, but is a problem that still requires further investigation. Because of their noisy nature certain models might struggle more than others when trained to synthesize these kinds of sounds. However, even though the existence of perceptual losses that involve a conversion to the frequency domain, and their alleviation of some of the past problems of waveform synthesis with convolutional neural networks, we think that it is possible to further improve loss functions and make them evermore closer to human perception. At some point we hope to fully step away from losses in the traditional sense and fully embrace powerful learned metrics similar to discriminators in GANs, naturally without the drawbacks that they incur.

Bibliography

[1] C. Aouameur, P. Esling, and G. Hadjeres. Neural drum machine : An interactive system for real-time synthesis of drum sounds, 07 2019.

[2] S. Arık, H. Jun, and G. Diamos. Fast spectrogram inversion using multi-head convolutional neural networks. *IEEE Signal Processing Letters*, 26(1):94–98, 2019.

[3] A. Defossez, N. Zeghidour, N. Usunier, L. Bottou, and F. Bach. Sing: Symbol-to-instrument neural generator. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 9041–9051. Curran Associates, Inc., 2018.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[5] C. Donahue, J. McAuley, and M. Puckette. Adversarial audio synthesis. In *International Conference on Learning Representations*, 2019.

[6] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders, 2017.

[7] J. B. J. Fourier. *Théorie Analytique de la Chaleur*. Cambridge Library Collection. Cambridge University Press, 2009.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani,

M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680. Curran Associates, Inc., 2014.

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

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[11] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 6626–6637. Curran Associates, Inc., 2017.

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[13] E. Jacobsen and R. Lyons. The sliding dft. *IEEE Signal Processing Magazine*, 20(2):74–80, 2003.

[14] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu. Efficient neural audio synthesis. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2410–2419, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[15] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[16] D. P. Kingma and M. Welling. An introduction to variational autoencoders. *Found. Trends Mach. Learn.*, 12(4):307–392, 2019.

[17] J. Kong, J. Kim, and J. Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[18] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition, 2019.

[19] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 14910–14921. Curran Associates, Inc., 2019.

[20] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1558–1566. JMLR.org, 2016.

[21] H. D. Luke. The origins of the sampling theorem. *IEEE Communications Magazine*, 37(4):106–108, 1999.

[22] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2813–2821, 2017.

[23] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018. cite arxiv:1802.03426Comment: Reference implementation available at http://github.com/lmcinnes/umap.

[24] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio. Samplernn: An unconditional end-to-end neural audio generation model, 2016. cite arxiv:1612.07837.

[25] J. Nistal, S. Lattner, and G. Richard. Drumgan: Synthesis of drum sounds with timbral feature conditioning using generative adversarial networks, 08 2020.

[26] A. V. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1747–1756, New York, New York, USA, 20–22 Jun 2016. PMLR.

[27] A. V. Oppenheim and R. W. Schafer. From frequency to quefrency: a history of the cepstrum. *IEEE Signal Processing Magazine*, 21(5):95–106, 2004.

[28] K. Palanisamy, D. Singhania, and A. Yao. Rethinking CNN models for audio classification. *CoRR*, abs/2007.11154, 2020.

[29] S. Pan and Q. Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[30] W. Ping, K. Peng, K. Zhao, and Z. Song. Waveflow: A compact flow-based model for raw audio. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 7706–7716. PMLR, 2020.

[31] R. Prenger, R. Valle, and B. Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 3617–3621. IEEE, 2019.

[32] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen. Improved techniques for training gans. In D. Lee, M. Sugiyama,

U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 2234–2242. Curran Associates, Inc., 2016.

[33] K. Schouhamer Immink. Compact disc story. *Journal of the Audio Engineering Society. Audio Engineering Society*, 46:458–460, 462, 464, 05 1998.

[34] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016.

[35] A. van den Oord, N. Kalchbrenner, L. Espeholt, k. kavukcuoglu, O. Vinyals, and A. Graves. Conditional image generation with pixelcnn decoders. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 4790–4798. Curran Associates, Inc., 2016.

[36] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis. Parallel WaveNet: Fast high-fidelity speech synthesis. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3918–3926, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[37] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[38] L. R. Varshney and J. Z. Sun. Why do we perceive logarithmically? *Significance*, 10(1):28–31, 2013.

[39] S. Vasquez and M. Lewis. Melnet: A generative model for audio in the frequency domain. *CoRR*, abs/1906.01083, 2019.

[40] J. Volkmann, S. S. Stevens, and E. B. Newman. A scale for the measurement

of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):208–208, 1937.

[41] R. Yamamoto, E. Song, and J. Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6199–6203, 2020.

[42] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, May 2016.

[43] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Muller, R. Manmatha, M. Li, and A. Smola. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020.