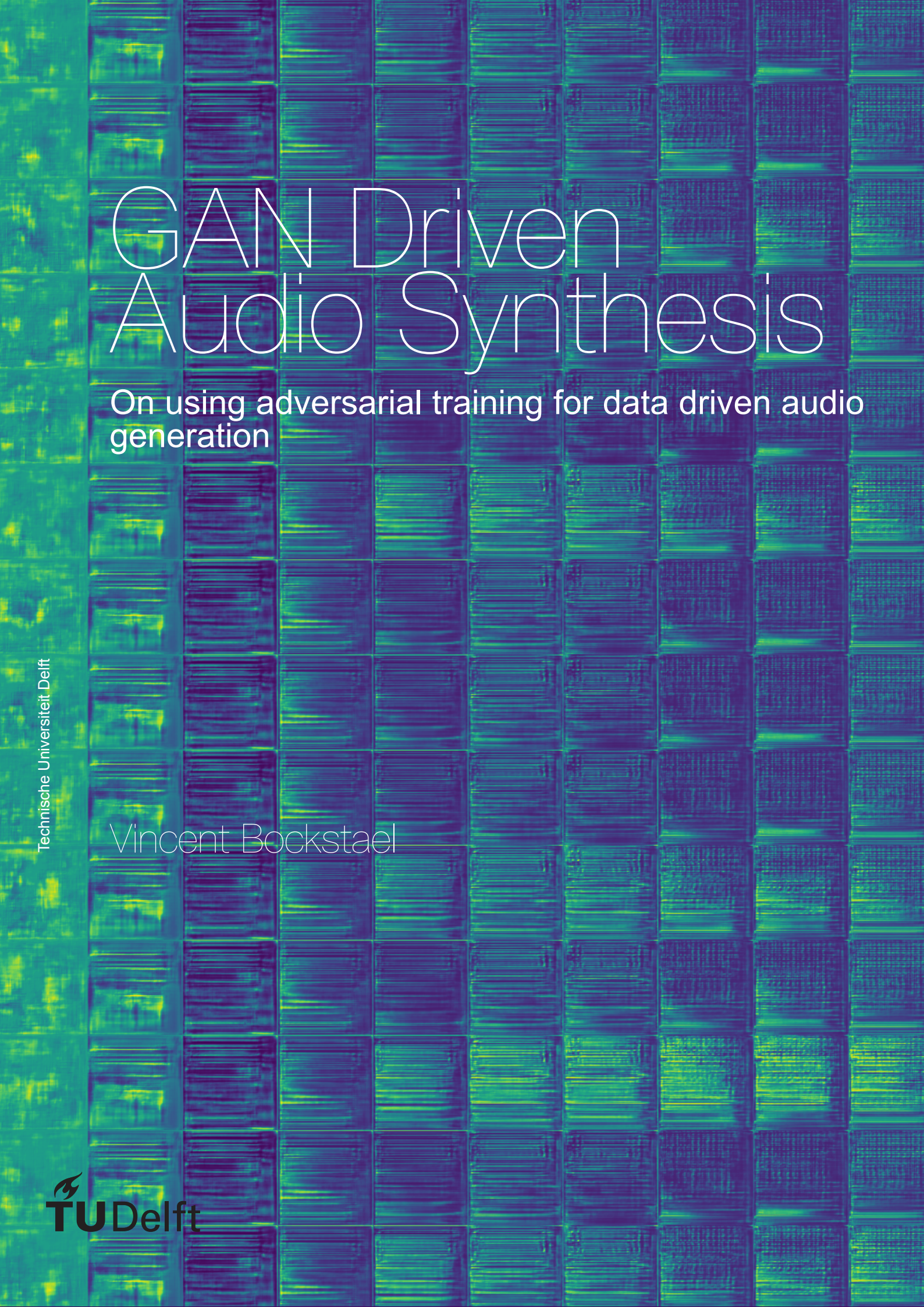

GAN Driven Audio Synthesis

On using adversarial training for data driven audio generation

Vincent Bockstael

Technische Universiteit Delft

TU Delft

GAN Driven Audio Synthesis

On using adversarial training for data driven audio generation

by

Vincent Bockstael

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Wednesday August 25, 2021 at 10:00 AM.

Abstract In this study, we investigate the usage of generative adversarial networks for modelling a collection of sounds. The proposed method incites an interpretation of musical sound synthesis based on audio collections rather than synthesizer component controls. This promises the generation of arbitrarily complex sounds without the restrictions of traditional synthesizer components. Furthermore, the method promises to introduce non-linear interpolations within abritrarily varied collections of sounds. These two elements motivate a new approach in creating musical instruments. Here, we introduce a proof of principle method with qualifications and quantifactions of the results. First, we cover the image-like audio signal representation and neural network architectures that compose a trainable system capable of producing audio signals. Despite some artifacts, the trained system is able to produce structural similarities in the spectral information compared to the training data set. Furthermore, we introduce a metric to quantitatively compare signal characteristics between two sets of signals. The difference between characteritics appears to decline throughout the training of the system.

Student number: 4590694
Project duration: March 1, 2020 – August 25, 2021
Thesis committee: Dr. ir. R. van der Toorn, TU Delft, supervisor
Dr. C. Kraaikamp, TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Contents

1 Introduction

1.1. Synthesizer

A synthesizer, in the context of music, is an electronic music instrument that generates sound signals. Traditional synthesis methods are built up on hand-designed components like oscillators and filters that can be tuned to generate their accompanying characteristic sounds.

Technological breakthroughs with incorporating electronic components into the music instruments, such as the Telharmonium, Trautonium, Ondes Martenot, theremin and Hammond, gave rise to the synthesizer [2]. In 1952, Harry Olson and Herbert Belar of the RCA (Radio Corporation of America) created the first synthesizer capable of artificially creating sound. The synthesizer consisted of a room filled with interconnected sound synthesis components.

Synthesizers were initially regarded as avant-garde instruments, with electronic compositions such as Edgard Varèse: Poème électronique (1958) [1], and Hugh Le Caine: Dripsody (1955) [2]. The 1960s valued the synthesizer in psychedelic and counter-cultural scenes, such as Fifty foot hose: Cauldron (1968) [3], but it promised little commercial potential. However, the unexpected commercial and critical succes of Bach compositions for synthesizer, Switched-On Bach (1968) [4] by Wendy Carlos incited the synthesizer's recognition as a usable musical instrument. The synthesizer was further popularized in the late 1960s and 1970s by electronic acts, pop and rock groups.

With the introduction of software that offered device simulation, synthesizers tended to dematerialize gradually towards digital variants. As a result, computers and synthesizers came together to provide the musician with an extensive instrumental palette. The synthesizer is now used in almost every genre of music and is regarded as one of the most important instruments in the music industry. [5]

1.2. Synthesizer as a Parametric Function

The components within synthesizers are shaped and modulated by envelopes, filters, and oscillators, which creates the characteristic artificial and, "synthetic" perception of the synthesizers [6]. Clearly, controlling the individual components to create precise and complex sounds by hand is a hard task and involves numerous decisions to be made on a microscopic time scale, hence the use of signal controllers such as oscillators and envelopes.

[1] Edgard Varèse: Poème électronique (1958)
[2] Hugh Le Caine: Dripsody (1955)
[3] Fifty foot hose: Cauldron (1968)
[4] No online publication is available, however there is a version by Leonard Bernstein
[5] According to Fact magazine in 2016
[6] Electric Samurai Tomita, Switched On Rock 1974 (vinyl record)

Approaching the synthesizer component controls as input variables, a synthesizer can be represented as the result of a function s of time t and the synthesizer controls z.

$$s(t, z) = x(t)$$

For example, the following function desribes a single sine oscillator synthesizer generator with a being the amplitude of the wave and f the frequency, i.e. $z = \{a, f\}$,

$$s(t, a, f) = a \sin(ft) = x(t)$$

Limiting the output signal to a fixed length, a synthesizer can be written as a fixed length signal generator, omitting time dependency t. In that case, a synthesizer s can be written as a mapping from the synthesizer controls to a fixed length signal, $x = s(z)$, solely dependent on the synthesizer controls z.

1.3. Generative Model as Synthesizer

In mathematics, a generative model can be used to ”generate” instances of a data distribution given an input. Given a collection of audio C, for which the instances are produced by a sound synthesizer s. Then, generating audio signals with the usage of $s(z)$ can be interpreted as the generative model of the collection of audio C.

While the term ”modelling” is used in different contexts with different definitions and can raise some confusion at times, its usage in this study is to conform to the usage in the field of machine learning research [7]. When we consider the term modelling, multiple interpretations can be followed. In this study we define the term modelling as the ”analyzing” and ”generalizing” of a data set. Its behaviour amounts to the interepreting of characteristics and approximating them in reproductions. Although more appropriate terms could be chosen for this behaviour, we favour the conformity with the research field. When we refer to a ”generative model” in particular, we can use the term ”generator” interchangeably.

Let $g_\theta : \mathcal{Z} \to \mathcal{X}$ be a target function parameterized by θ that represents our generator. We wish to obtain a value for θ such that we can generate random instances of our target space $\mathcal{X}$ given an input attribute from $\mathcal{Z}$.

In answering the question of why to use a generator for sounds, we can view the reprehension of a collection of sounds in two paradigms. The first one is a collection of points representing the fixed length audio waveforms directly, i.e. the collection itself, from which we can observe every individual element of that collection. The second one is a generator, or generative model, for that collection of audio waveforms, i.e. a function that is able to generate fixed length audio waveforms similar to the audio waveforms in the collection based on some input attribute.

For example, in context of synthesized sounds, a collection of audio recordings from a synthesizer s can be interpreted as individual fixed length audio waveforms. Clearly, a perfect generator for this collection then is the synthesizer $s(z)$ it self. With the synthesizer we are able to easily generate the audio waveforms that are present in the collection.

However in the case of complex, or natural, sounds, such as speech, the collection would be audio recordings of speech, but the ’synthesizer’ of speech would be your voice. This is inevitably hard to reproduce synthetically with hand-tuned synthesizer components.

Another scenario is where sounds in the data set have limited correlation. That is, the collection of sounds contains different sources, such as a collection of environmental sounds. Providing a traditional synthesizer to produce audio from this collection is inheritely hard as the environmental sounds can be arbitratily different and the scope of the synthesizer is limited to its components. A generator for this collection of audio, however, should be able to produce singals with characteristics from all

[7] In [9], the introduction of GAN, ”generative modelling” is used to define the behaviour of a generator.

individual classes of audio within the collection.

In general, the modelling of a collection of audio waveforms requires an audio synthesis method that is able to produce audio waveforms similar to that of the collection. A generator for a collection of sounds should be able to produce arbitrarily complex sounds with similar charateristics to sounds in the collection. That is, the produced sounds are not limited by the scope of synthesizer components.

1.4. Machine Learned Generative Model

Machine learning is a data analysis technique that allows the creation of models. It is a subfield of artificial intelligence that is based on the idea that systems can recognize patterns and make decisions with limited human intervention. Using machine learning to obtain a generator for audio, allows us to build synthesizers that model collections of audio data with parameters which are too complex, too sensitive or too high in quantity to model, or even define, manually.

Given a large amount of data, the parameters θ of a generative model g_θ can be precisely optimized to represent the data set using machine learning methods. One method to obtain a generative model is the use of a *generative adversarial network*, or GAN [9]. This method is based on an adversarial training process between two competing networks: one is trying to generate material, while the other is classifying the generated material and data set material as 'real', or as 'fake'.

1.5. Research Overview

In this study we design and train such a generative adversarial network, with inspirations from DCGAN [29], GANSynth [6], and GAN [9], to obtain a generator that is able to model sounds from a collection of recordings of electronic pianos [8] playing note middle c. (**Chapter 2, 3, 4**).

With this system we inspect the quality of the generated sounds in comparison to the sounds in the data set by visually comparing the spectral information. Furthermore, we observe the effects of "latent space interpolation" on audio. That is, the inference of a trained generator on interpolated points in the input space of the generator, as is demonstrated in [9] and [29]. **(Chapter 5)**

After this, we perform experiments with the resulting generator network to test its modelling abilities with respect to the original collection of audio. We measure and compare audio signal qualities of material produced by the generator and that of the data set with the help of low level signal descriptors. That is, the signals produced from generator network and signals from the dataset are compared w.r.t. to a set of audio characteristics. At last, we quantify the performance of the generator throughout adversarial training with a perfomance metric based on the previously defined audio characteristics. This specifically aims to quantify the effect of training the network. **(Chapter 6)**

[8] For example a Rhodes Mark I 1975

2

Audio, Image-like Representation and Processing

While this study globally focusses on generating audio signals, the modelling processes actually take place in the image-domain. This is a reasonable choice since audio can be bidirectionally represented with image-like time-frequency information.

The representation illustrates the characteristics of sounds in a conceptually easier way. For example, the fundamental frequency, timbre and loudness of a sound can be seen directly from the image. When representing audio signals in a waveform, the fundamental frequency and timbre are hard to capture directly.

Overview

In this chapter we will descibe the characteristics of sound that can be perceived within the image-like time-frequency information, and the processing that allows us to bidirectionally represent an audio signal in this image.

2.1. Audio

Sound is physically defined as a vibration that propagates through a transmission medium. Typically, this refers to a vibration in air pressure, caused by a sound source. When the vibration reaches the ear, an impulse is created to the brain resulting in an audible perception of the sound source.

2.1.1. Characterizing Sound

Describing sound, without hearing, is a teadious and subjective process. In this study we limit the characterization of the perception of sounds to pitch, loudness and timbre.

Pitch The frequency of a cyclical vibration determines the perception of pitched sound. It is the essential quality that allows distinguishing between low and high sound. Pitch allows musician to use expression in tonality, such as in melodies, scales, and chords. In case of a non cyclical vibration, pitch is indetermined. For example, no pitch is perceived in noise, because there is no cyclical vibration in noise.

Loudness The amplitude of a vibration determines the perceived loudness of a sound. This characteristic defines differences in loud and soft sounds. Controlling loudness allows musicians to utilize expression in the field of dynamics. Dynamics on a micro scale in sound is often referred to as the envelope of a sound. The envelope of a sound categorizes it into a plucky sound like a guitar, and a swelling sound like a violin.

Timbre The timbre of a sound is the perceived quality of a sound or tone. Timbre is determined by the spectral information of the sound. That is, the distribution of amplitudes frequency multiples of the pitch (hamonic and non-harmonic frequencies). Timbre differentiates sounds that have a similar pitch and a similar envelope. For example, the difference between a piano and a harpsichord or a saxophone and a clarinet playing the same note with the same envelope,.

2.1.2. Audio Signal

An audio signal is a representation of sound. As sound in nature is a continuous sequence, in digital audio, the audio signal is a numerical estimate of the sequence. Audio signals typically contain frequencies in the range of 20 to 20,000 Hz. This corresponds to the lower and upper bounds of human hearing.

Discrete-time signals are represented mathematically as a sequence of numbers. Given a sequence of numbers x, the nth number in the sequence is denoted by $x[n]$. Audio signals as entire sequences are refered to as x.

The rate at which samples from the continuous sequence are extracted by the numerical estimation is called the *sample rate*. The sample rate also corresponds to the rate at which samples are converted in order to (re)create a sound. Common choices for sample rates lie in the tens of thousands, e.g. 44.1 kHz.

The methods in this study will be implemented with a collection of fixed length audio signals $\mathfrak{D}$ [1]. Therefore, the choices made in the audio processing are dependent on the audio length and sampling rate of the singals in that collection. The audio signals in $\mathfrak{D}$ have a sample rate of 16 kHz and a length of 4 seconds, corresponding to 64000 samples in total. Let $\mathcal{X}$ refer to the space of signals x with length 64000, than $\mathfrak{D} \subset \mathcal{X}$. According to these constants, choices related to transformations of the signals are made in the upcoming sections.

2.1.3. Frequency Domain

The frequency domain of a signal refers to the analytical space in which a signal is represented in terms of frequency, rather than time.

Fourier Transform The Fourier transform is able to convert a signal from the time domain into the frequency domain. The signal in time domain is transformed into a sum of sine waves of different frequencies, each of which represents a frequency component. The spectrum of frequency components is the frequency-domain representation of the signal.

The discrete Fourier transform [1], relevant as utilize digital signals, transforms a signal x into a sequence of complex numbers X, where $X[k]$ is defined by

$$\begin{aligned} X[k] &= \sum_{n=0}^{N-1} x[n] \cdot e^{-\frac{i2\pi}{N}kn} \\ &= \sum_{n=0}^{N-1} x[n] \cdot \left[\cos\left(\frac{2\pi}{N}kn\right) - i \cdot \sin\left(\frac{2\pi}{N}kn\right)\right] \end{aligned} \tag{2.1}$$

Within the frequency domain, **pitch** can be interepreted by the fundamental frequency (the lowest promiment frequency), **loudness** can be interpreted by the overall magnitudes of frequencies, and at last **timbre** can be interpreted from the amplitude distribution of harmonic frequencies that are present.

[1] Further elaboration will be provided in chapter 5

Figure 2.1: Sine and cosine decomposition of a waveform. Red represents the signal wave form. Blue represents the sines and cosines. (image adapted from [37])

Figure 2.2: Time (red) and frequency domain (blue) representation of a wavefrom signal. (image adapted from [37])

2.2. Processing Image-like Representation of Signals

In order to transform the audio signals to and from the image-like time-frequency representation, appropriate processing is necessary.

In this section the audio signal-to-image transformation Φ and image-to-signal transformation Φ^{-1} will be discussed in detail. In short, the **signal-to-image transformation** of audio signals converts a signal (64000 samples) to 2 stacked 256×256 images containing *mel scale log magnitude spectrograms* and *instantaneous frequency* representations. Whereas, the **image-to-signal transformation** amounts to the inverse: 2 stacked 256×256 images are converted to a 4 second audio signal, again sampled at 16 kHz.

Figures 2.3 and 2.4 visualize the signal-to-image and image-to-signal transformations.

2.2.1. Signal to Image

The signal-to-image transformation in 2.3 can be written as a mapping Φ that maps the signal space of 64000 samples $\mathcal{X}$ to the 2 stacked image space of $256 \times 256 \times 2$ $\mathcal{S}$,

$$\Phi : \mathcal{X} \mapsto \mathcal{S} \tag{2.2}$$

The function Φ is a composition of the following functions

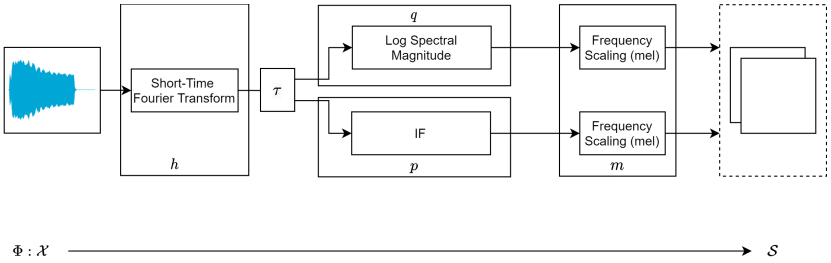

Figure 2.3: Given a signal $x \in \mathcal{X}$, the short-time Fourier transform is calculated (h) and split in to the argument and magnitude components (τ). Then from the magnitude components the log spectral magnitude is calculated (q) and from the argument components the instantaneous frequency is calculated (p). At last, mel frequency scaling is applied on both components (m). The result is a $256 \times 256 \times 2$ image-like structure in $\mathcal{S}$ containing mel frequency scaled log spectrograms and instantaneous frequencies.

$$\Phi = \begin{pmatrix} m \circ q \\ m \circ p \end{pmatrix} \circ \tau \circ h \tag{2.3}$$

h, in section 2.2.3, is the short-time Fourier transform calculation, giving the sinusoidal frequencies and phase content of local sections of a signal. This is a time dependent variation on the discrete Fourier transform from section 2.1.3. A signal in $\mathcal{X}$ is mapped to a complex valued 256×256 array, where the complex values represent information on phase and amplitude at a given frequency and time. One axis represents time, the other represents frequency. τ, section 2.2.3, seperates a complex number into magnitude and argument. A complex valued 256×256 array is split into a real valued 256×256 array, and a 256×256 array containing values in $[0, 2\pi)$. The seperated parts represent magnitudes and phase information respectively. p, in section 2.2.5, is the instantaneous frequency calculation. It maps a 256×256 array of values in $[0, 2\pi)$ to a real valued 256×256 array by taking the temporal rate of unwrapped phase. q, in section 2.2.4, is the spectral magnitude calculation. It maps a real valued 256×256 array to a real valued 256×256 by taking squares and logarithms. m, in section 2.10, is the scaling of the frequency axis to a mel [36] scale, such that they better represent human hearing.

2.2.2. Image to Signal

Inversely, the image-to-signal transformation Φ^{-1} in figure 2.4 converts the stacked-images $s_x \in \mathcal{S}$ to an audio signal $x \in \mathcal{X}$.

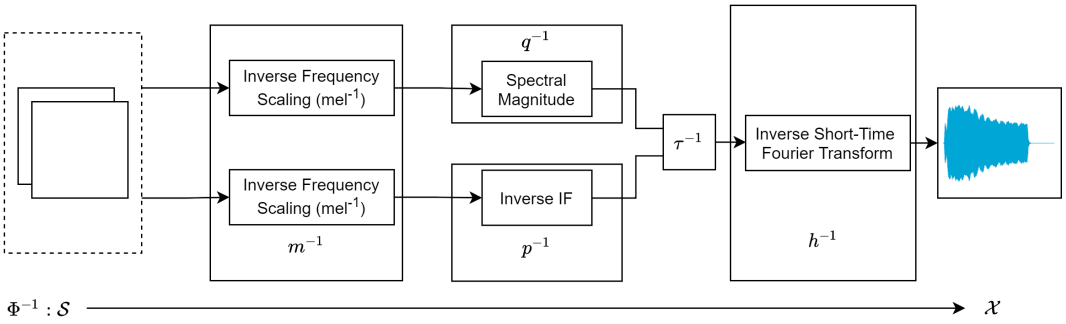

Figure 2.4: Given a $s \in \mathcal{S}$, inverse frequency scaling is applied on both compoments of s (m^{-1}). Then inverse of spectral magnitude calculation is applied on the spectral magnitude component (q^{-1}) and the argument is calculated from the instantaneous frequency (p^{-1}). From the magnitude and argument components a 256×256 complex grid is given (τ^{-1}). At last the inverse short-time Fourier transform is applied (h^{-1}) to obtain signal $x \in \mathcal{X}$.

The inverse operation Φ^{-1} from 2.4 can be written as a mapping $\Phi^{-1} : \mathcal{S} \mapsto \mathcal{X}$.

$$\Phi^{-1} = h^{-1} \circ \tau^{-1} \circ \begin{pmatrix} q^{-1} \circ m^{-1} \\ p^{-1} \circ m^{-1} \end{pmatrix} \tag{2.4}$$

h^{-1}, in section 2.2.3, is the inverse short-time Fourier transform calculation that maps a 256×256 complex valued array to a singal in $\mathcal{X}$. τ^{-1}, in section 2.2.3, gives a complex valued 256×256 array from a real valued magnitude array and a $[0, 2\pi)$ valued argument array of similar dimensions. p^{-1}, in section 2.2.5, is the calculation of phase from instantaneous frequency. q^{-1}, in section 2.2.4, is the inverse of the spectral magnitude calculation. m^{-1}, in section 2.10, is the conversion from mel scale to hz scale.

2.2.3. Short-Time Fourier Transform

The short-time Fourier transform of signal x is a derivation of the Fourier transform used to determine sinusoidal frequency and phase content of local sections of a signal over time.

In practice, this procedure consists of dividing a signal into shorter segments of equal length and then compute the discrete Fourier transform separately on each shorter segment. [1, 16]

Window Function These shorter segments are obtained with the use of a window function w. A window function is non-zero within a specified interval and zero outside of the interval. For w we choose to use a *Hann* window function [10],

$$w[n] = 0.5 \left(1 - \cos\left(\frac{2\pi n}{M-1}\right)\right) \tag{2.5}$$

M is the size of the interval the window function. This corresponds to the length of the resulting windowed signal, as outside of the window inteval the signal is zero.

STFT Calculation The short-time Fourier transform of a signal x, $\mathbf{STFT}\{x\}$, gives a two-dimensional representation of the signal and is calculated as follows, [1]

$$\mathbf{STFT}\{x\}(m, \omega) = \sum_{n=-\infty}^{\infty} x[n] w[n-m] e^{i\omega n} \tag{2.6}$$

Where w is a window function, $x[n]$ is the value of discrete signal x at position n, and m and ω represent the position on the time and frequency axis respectively.

The mapping $h : \mathcal{X} \mapsto \mathbb{C}^{256\times256}$, found in figure 2.3, can be defined by the calculation of the short-time Fourier transform for all time frames, and frequency bins. This amounts to the calculating equation 2.6 for all m, ω, such that $1 \leq m \leq 256, 1 \leq \omega \leq 256$, resuling in the following 256×256 complex valued array.

$$h(x) = \begin{pmatrix} \mathbf{STFT}\{x\}(1,1) & \dots & \mathbf{STFT}\{x\}(256,1) \\ \vdots & \ddots & \vdots \\ \mathbf{STFT}\{x\}(1,256) & \dots & \mathbf{STFT}\{x\}(256,256) \end{pmatrix} \tag{2.7}$$

Inverse STFT Calculation To obtain an inverse for the calculation of the **STFT** (**iSTFT**), we use the following procedure. [1]

Let X be a complex valued array of 256×256, that we interpret as an **STFT** of a signal, then for fixed m,

$$x[n]w[n-m] = \frac{1}{2\pi} \sum_{\omega=1}^{256} X_{m,\omega} e^{i\omega n} \tag{2.8}$$

Here for $m > 1$, with equation 2.8 we calculate a window of signal $x[n]$ at position m. Then since by the choice of the window function w, $\sum_{m=1}^{\infty} w[n-m] = 1$,

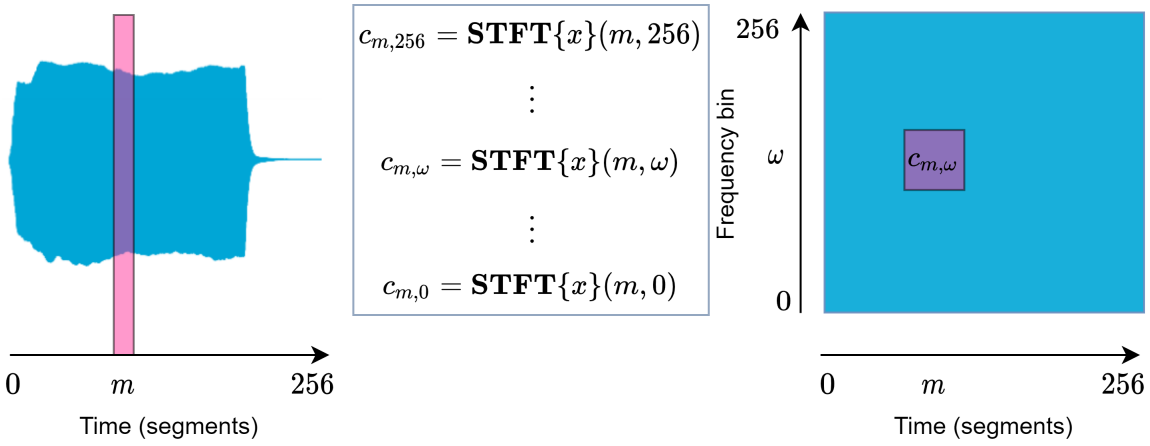

Figure 2.5: A signal x (left) is segmented into 256 pieces. Than, at segment m, $\mathbf{STFT}\{x\}(m, \omega) = c_{m,\omega}$, where ω is the frequency bin. This results into a 256×256 grid of complex values $c_{m,\omega}$. Each value represents the amplitude and phase of the frequency bin ω in the signal at time segment m.

$$x[n] = \sum_{m=1}^{\infty} x[n]w[n-m] \tag{2.9}$$

The inverse operation **iSTFT** then amounts to,

$$x[n] = \frac{1}{2\pi} \sum_{m=1}^{256} \sum_{\omega=1}^{256} X_{m,\omega} e^{i\omega n} \tag{2.10}$$

The inverse mapping of h, $h^{-1} : \mathbb{C}^{256\times256} \mapsto \mathcal{X}$, as found in figure 2.4, gives a sequence of numbers $h^{-1}(X)$ of length 64000. For position n in signal $h^{-1}(X)$,

$$h^{-1}(X)[n] = \frac{1}{2\pi} \sum_{m=1}^{256} \sum_{\omega=1}^{256} X_{m,\omega} e^{i\omega n} \tag{2.11}$$

$X_{m,\omega}$ refers to the value of X at row ω and column m, i.e. at frequency ω and time step m.

Argument and Magnitude The calculation of short-time Fourier transform gives complex values. In order to further process the short-time Fourier transform for phase and spectral magnitude information, we split the complex values into its argument and magnitude.

Let $z = x + yi$ be a complex number, which can be split into its magnitude r,

$$r = |z| = \sqrt{(x^2 + y^2)} \tag{2.12}$$

and argument φ,

$$\varphi = \arg(x + yi) = \begin{cases} 2\arctan\left(\dfrac{y}{\sqrt{x^2+y^2}+x}\right) & \text{if } x > 0 \text{ or } y \neq 0, \\ \pi & \text{if } x < 0 \text{ and } y = 0, \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0. \end{cases} \tag{2.13}$$

Let $\tau : \mathbb{C}^{256\times256} \to \mathbb{R}^{256\times256} \times [0, 2\pi)^{256\times256}$, as found in figure 2.3, be the complex function that splits a 256×256 array of complex numbers into an 256×256 array of magnitudes and arguments.

Accordingly let $\tau^{-1} : \mathbb{R}^{256\times256} \times [0, 2\pi)^{256\times256} \mapsto \mathbb{C}^{256\times256}$, as found in figure 2.4, be the inverse operation. Let R be a 256×256 array of real values. Let P be a 256×256 array of real values in $[0, 2\pi)$.

Then this operation creates a complex value from $R_{m,\omega}$ and $P_{m,\omega}$ as its magnitude and argument, i.e. $R_{m,\omega}e^{iP_{m,\omega}}$ for row ω and column m.

2.2.4. Spectral Magnitude Information

A spectrogram is a visual representation of the frequency spectrum of a signal as it varies with time. It is obtained from the magnitudes of the short-time fourier transformation. Given the short-time fourier transform magnitudes $R \in \mathbb{R}^{256\times256}$, calculated by τ, the spectrogram representation of the signal can be obtained as follows,

$$\text{spectrogram}\{x\}(m,\omega) \equiv R^2_{m,\omega} \tag{2.14}$$

m and ω represent time step and frequency bin, and

$$R_{m,\omega} = |\mathbf{STFT}\{x[n]\}(m,\omega)| \tag{2.15}$$

as obtained in the previous section 2.2.3 by τ.

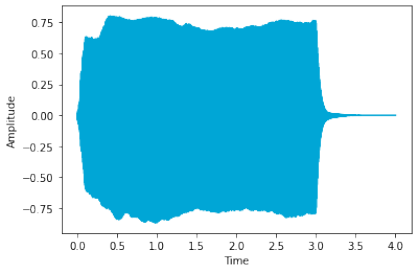

Figure 2.6: Plot of the waveform of a signal

Figure 2.6 shows the waveplot of signal x. The spectrogram of the same signal x is seen in 2.7. However as seen in the comparison in figure 2.7, the $\log$ magnitudes of frequencies gives a better visual representation for frequencies than regular frequency magnitudes. With log magnitudes, more subtle spectral information is visualized.

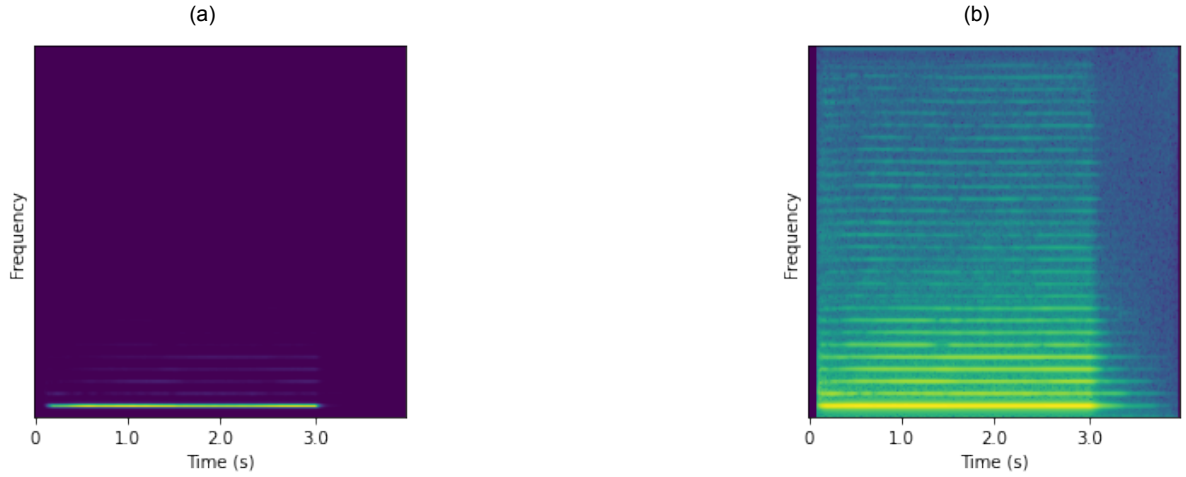

Figure 2.7: A side by side comparison of (a) a regular spectrogram and (b) a log magnitude spectrogram. Visually, a log magnitude spectrogram (b) yields more information on what frequencies are audible at a time.

The composition $\tau \circ h : \mathcal{X} \mapsto \mathbb{R}^{256\times256} \times [0, 2\pi)^{256\times256}$ decomposes the complex values in the result of $h(x)$ for $x \in \mathcal{X}$ into magnitudes R and arguments P. The log magnitude spectrogram of a signal x is

obtained from R by elementwise $\log$ square of R. This gives us a definition for q, as found in figure 2.3. Namely, given a time step m and an frequency bin ω,

$$q(R)_{m,\omega} = \log R^2_{m,\omega} \tag{2.16}$$

Consequently the inverse q^{-1}, as found in 2.4, amounts to

$$q^{-1}(S)_{m,\omega} = e^{\sqrt{S_{m,\omega}}} \tag{2.17}$$

for a log magnitude spectrogram in $S \in \mathbb{R}^{256\times256}$

2.2.5. Phase Information

Let the phase ϕ of a signal x at position n be denoted by $\phi[n] = \arg\{x[n]\}$. Phase information is not easily represented in an image. In fact, the result is often noisy similar to that in figure 2.8 which is the phase representation of the signal in figure 2.6 and spectrograms in figure 2.7.

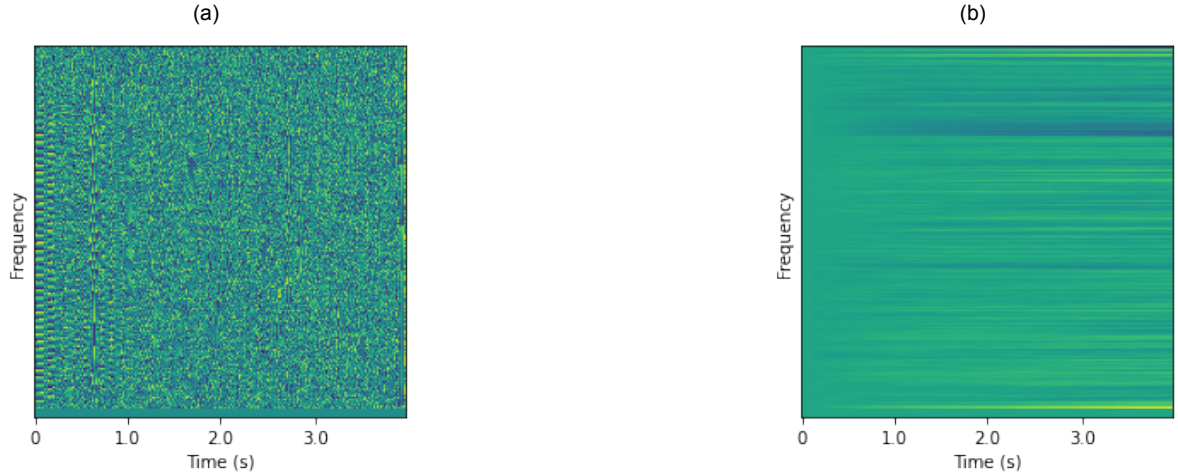

Figure 2.8: Phase of signal $x[n]$ with (a) wrapped between $[0, 2\pi)$ and (b) unwrapped

Instantaneous Frequency For unwrapped ϕ, that is not wrapped in $[0, 2\pi)$ as seen in figure 2.8, the instantaneaus frequency f is defined as the temporal rate of the phase, in other words, the finite difference of ϕ at position n with respect to n with step size $h = 1$,

$$f[n] = \phi[n+1] - \phi[n] \tag{2.18}$$

Unwrapping Phase By unwrapping ϕ discontinuities can be removed by adding 2π to $\phi[n]$ whenever $\phi[n+1] - \phi[n] \leq -\pi$, and subtracting 2π from $\phi[n]$ whenever $\phi[n+1] - \phi[n] > \pi$.

$$u(\phi)[n] = \begin{cases} \phi[n] & \text{if } -\pi < \phi[n+1] - \phi[n] \geq \pi, \\ \phi[n] + 2\pi & \text{if } \phi[n+1] - \phi[n] \leq -\pi, \\ \phi[n] - 2\pi & \text{if } \pi < \phi[n+1] - \phi[n] \end{cases} \tag{2.19}$$

Due to the periodic nature of phase, the instantaneous frequency f gives a more structured display of phase information. Given that the phase in figure 2.7 gives a noisy display, we find that the instantaneous frequency from figure 2.9 is more suitable for pattern recognition. [6]

From τ in section 2.2.3 we obtain the phase information $P \in [0, 2\pi)^{256\times256}$. P has phase information $\phi_\omega[m]$ for every frequency bin ω (row), and time step m (column).

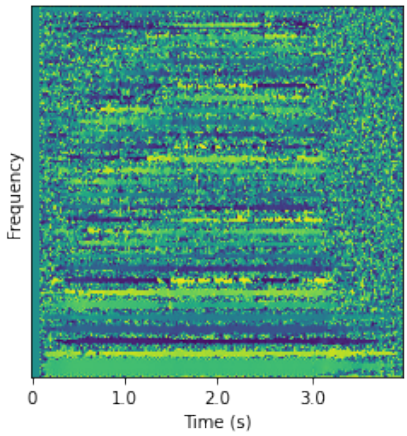

Figure 2.9: Instantaneous frequency of signal x

The function mapping $p : [0, 2\pi)^{256\times256} \mapsto \mathbb{R}^{256\times256}$, as found in 2.3, amounts to the finite difference of unwrapped ϕ_ω. This gives us a 256×256 array of instantaneous frequencies $p(P)$ given a 256×256 array of phases P. For time step m, and frequency bin ω,

$$p_{m,\omega}(P) = u(\phi_\omega)[m+1] - u(\phi_\omega)[m] \tag{2.20}$$

Inversely, given an array of instantaneous frequencies $\tilde{P} \in \mathbb{R}^{256\times256}$, the phase at frequency bin ω and time m amounts to the cumulative sum of $f_\omega[i]$ (or $\tilde{P}_{i,\omega}$), where $0 \leq i \leq m$, modulo 2π. So, $p^{-1} : \mathbb{R}^{256\times256} \mapsto [0, 2\pi)^{256\times256}$, with

$$p^{-1}_{m,\omega}(\tilde{P}) = \sum_{i=0}^{m} \tilde{P}_{i,\omega} \mod 2\pi \tag{2.21}$$

2.2.6. Frequency Scaling

Figure 2.7 shows that the most dominant frequencies lie in the bottom part of the frequency axis, that is ranging from 0 Hz to Nyquist frequency $F_N = \frac{F_s}{2}$ Hz, where F_s is the sample rate. The upper part of the spectrogram contains little information.

Therefore we use an alternative frequency scale of mel spectrograms (Mel) [36] along the frequency axis. Given the linear scale frequency values of f_{Hz}, the mel scale values of mel can be calculated to by [32],

$$mel(f_{\text{Hz}}) = 1127 \times \ln(1 + \frac{f_{\text{Hz}}}{700}) \tag{2.22}$$

Figure 2.10 shows the effect of frequency scaling on the visual appearance of spectral components depicted by spectrogram and instantaneous frequency images.

The mel scale corresponds to the process of expanding the lower frequency range features (about 0–2 kHz), and at the same time, contracting the higher frequency range features (about 2–8 kHz). Therefore, the application of mel scale effectively provides the network more and less-detailed information about the lower or upper range of the frequency spectrum, respectively.

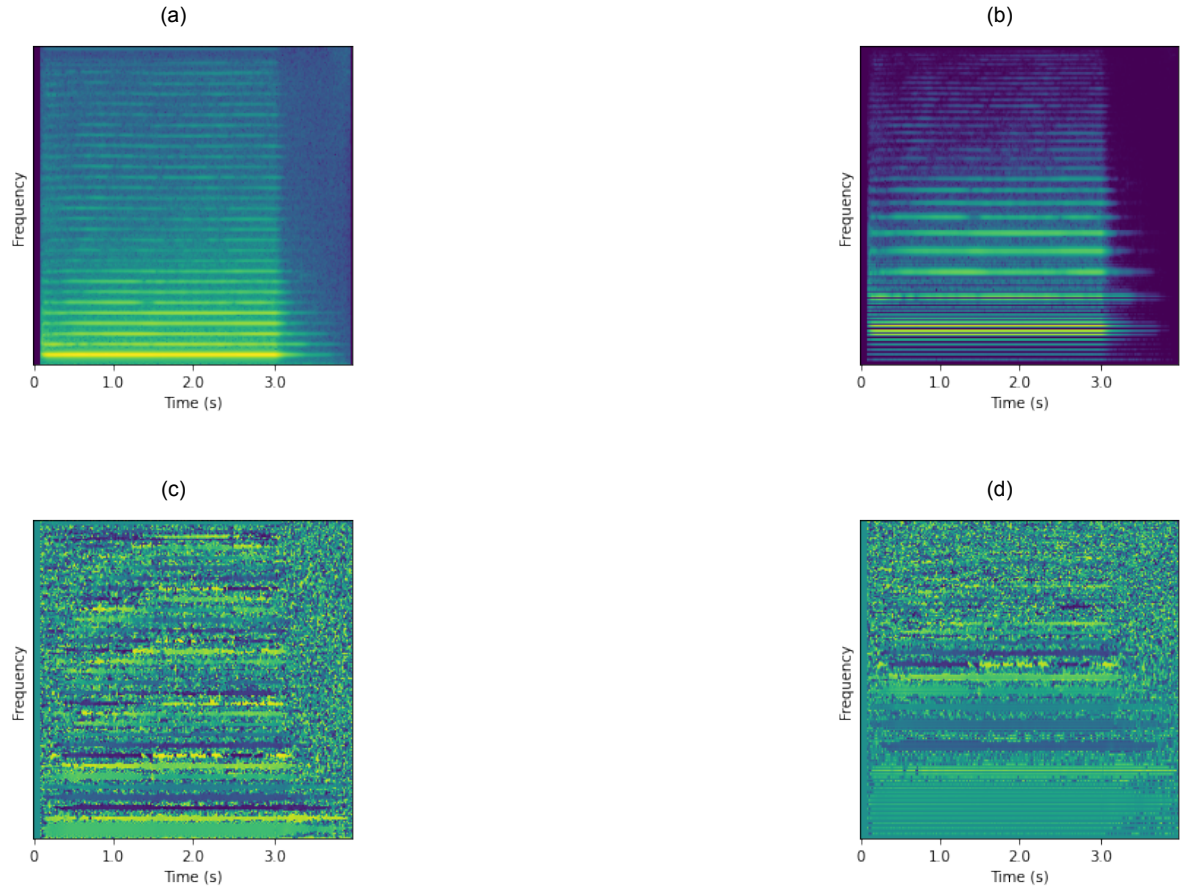

Figure 2.10: Comparison of spectrograms an instantaneous frequencies at Hz scale and mel scale. (a) shows Hz scale log magnitude spectrogram. (b) shows mel scale log magnitude spectrograms; (c) shows Hz scale instantaneous frequency. (d) shows mel scale instantaneous frequency.

These images are calculated by applying the frequency scaling on the frequency bins, i.e. the rows in the spectrogram images. Then given we have a frequency axis of 256 steps, the row f_{bin} correspoding to frequency in Hz, f_{Hz}, is

$$f_{bin}(f_{\text{Hz}}) = \text{floor}(256 \times \frac{f_{\text{Hz}}}{F_N})$$

And the inverse,

$$f_{\text{Hz}}(f_{bin}) = F_N \times \frac{f_{bin}}{256}$$

Since F_N is the maximimum frequency in the Hz scale spectrograms, and the maximum bin is 256, the maximum mel value is equal to,

$$mel_{max} = 1127 \times \ln(1 + \frac{F_N}{700})$$

This gives us the row in the mel spectrogram mel_{bin} from the mel value,

$$mel_{bin}(mel) = 256 \times \frac{mel}{mel_{max}}$$

And then finally the row in the mel scale spectrogram mel_{bin} corresponding to the row in the Hz scale spectrogram f_{bin},

$$mel_{bin}(f_{bin}) = 256 \times \frac{1127 \times \ln(1 + \frac{F_N \times \frac{f_{bin}}{256}}{700})}{mel_{max}}$$

Then for $a = \frac{256 \times 1127}{mel_{max}}$ and $b = \frac{F_N}{179200}$

$$mel_{bin}(f_{bin}) = a \times \ln(1 + bf_{bin})$$

and the inverse scaling,

$$f_{bin}(mel_{bin}) = \frac{e^{\frac{mel_{bin}}{a}} - 1}{b}$$

With a samplerate of 16 kHz, the Nyquist frequency $f_N = \frac{16000}{2} = 8$ kHz,the maximum mel value is $mel_{max} = 1127 \times ln(1 + \frac{8000}{700}) \approx 2840.03$. Then given $a \approx 101.59$ and $b \approx 0.045$, the scaling of the log spectrogram bin scaling to log mel spectrogram scaling amounts to

$$mel_{bin}(f_{bin}) \approx 101.59 \times \ln(1 + 0.045 f_{bin})$$

and the inverse,

$$f_{bin}(mel_{bin}) \approx \frac{e^{\frac{mel_{bin}}{101.59}} - 1}{0.045}$$

Then the function mapping $m : \mathbb{R}^{256 \times 256} \mapsto \mathbb{R}^{256 \times 256}$, as found in 2.3, that scales the log spectrogram in to a log mel spectrograms amounts to the shifting of row f_{bin} in the log spectrogram to the row $mel_{bin}(f_{bin})$ in the log mel spectrogram.

The inverse mapping m^{-1}, as found in 2.4, then amounts to the shifting of row mel_{bin} in the log mel spectrogram to the row $f_{bin}(mel_{bin})$ in the log spectrogram.

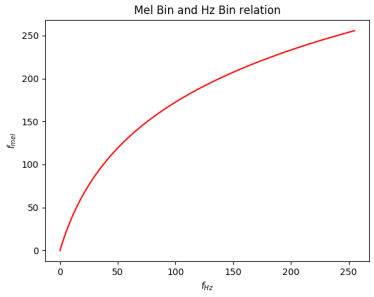

Figure 2.11: Relation between mel scale spectrogram rows and Hz scale spectrogram rows for sample rate 16 kHz.

Summary

In this chapter we covered topics related to sound and its characteristics:

- Sound is physically defined as a propagating wave. In context of musical notes, sound can be characterized by a pitch, an amplitude, and a timbre. (Section 2.1.1)
- Sound can be represented by a discrete signal x. The value $x[n]$ of x at time step n is sampled at a position in the corresponding sound wave with a specified samplerate. In this study, signals are restricted to lenght 64000 represenging 4 seconds of sound sampled at 16 kHz. The space of such signals is refered to as $\mathcal{X}$. (Section 2.1.2)
- We can represent sound in the frequency domain by the sinusoidal frequencies that make up the sound. (Section 2.1.3)

We covered the processes that are used to transform signals into desired representations:

- Φ (figure 2.3) transforms signals from $\mathcal{X}$ into 2 stacked 256×256 image-like frequency time representation of signals, containin spectral magnitude and phase information. (Section 2.2.1)
- Φ^{-1} (figure 2.4) represents the inverse transformation. (Section 2.2.2)

At last we covered the components that make up processes Φ and Φ^{-1}:

- Local sinusoidal frequencies and phase information can be obtained by calculating the *short-time fourier transform*, which gives a matrix of complex numbers with rows representing frequency bins and columns representing time segments. Denoted by h and h^{-1}. (Section 2.2.3)
- The spectral magnitude information can represented by a log spectrogram. This can be obtained from the magnitude of the complex numbers in the *short-time fourier transform* matrix. Denoted by q and q^{-1}. (Section 2.2.4)
- Interpreting phase information directly from the *short-time fourier transform* matrix yields noisy result. Instead, we represent phase information with the instantaneous frequency, which equals to the temporal rate of unwrapped phase. Denoted by p and p^{-1}. (Section 2.2.5)
- The *mel* scaling of a frequency axis for the instantaneous frequency and log spectrogram gives a frequency scale that is more coherent to human hearing. Denoted by m and m^{-1}. (Section 2.2.6)

3

Generative Model with a Generative Adversarial Network

In order to obtain a generative model for a collection of audio, we need a parametric function G_θ that is able to generate audio and whos parameters θ can be adapted to model the collection of audio. Or in other words, make G_θ produce audio similar to that of the collection it is derived from.

Overview

In this chapter we will discuss how we can use a generative adversarial network to obtain a generative model for a collection of audio. We discuss the inference of a generator that results in an audio signal. Also, we cover the training procedure required to obtain a generator that models a collection of audio. At last, we elaborate on the theoretical result of [9] by highlighting the convergence in function space and parameter space.

3.1. Generative Adversarial Network

A generative adversarial network, or GAN, is a method used to obtain a generative model. The network resembles a competition between a generator and a discriminator. [1] The generator is producing fake material, while the discriminator is distinguishing the fake material from the real material. This way it obtains a generative model for a data distribution. [9]

Generating material from scratch has inadequate criteria to be improved on. E.g. when trying to generate images of dogs, it is hard to find a performance evaluation for the generator to improved the parameters on, other than classifying the produced material as 'a dog' or 'not a dog' ('real', or 'fake'). This is what a generative adversarial networks aims to do.

We have seen from the previous chapter that, with appropriate transformations Φ and Φ^{-1}, we can obtain an $256 \times 256 \times 2$ image representation for fixed length audio signals. Representing audio as images allows us to more mature image-based GAN technology. [29]

3.2. Inference of a Generator

The inference of a generator is similar to playing a synthesizer in that an input is given to the generator from which the generator creates sound. A synthesizer's input are, other than the note played, controls for the synthesizer components such as filters and oscillators. When inferencing our generator the "control" is a multi-dimensional value z, from a *latent space* $\mathcal{Z}$.

[1] The neural network architectures of the generator and discriminator will be covered in chapter 4

A *latent space* refers to an abstract multi-dimensional space containing feature values that we cannot interpret directly. However the latent space encodes a meaningful representation of the observed outputs, e.g. generated audio.

In order to demonstrate the analogy between a synthesizer and a generator in GAN, we can view the synthesizer's controls as a hand-designed input space for the produced audio while the generator's input space represents an abstract input space, the latent space $\mathcal{Z}$.

In figure 3.1 the schematic of inferencing the generator network is shown. Instead of directly generating audio, the generator produces $256 \times 256 \times 2$ images which can be interpreted by the image-to-signal transformation ϕ^{-1}. Let $\mathcal{Z}$ and $\mathcal{S}$ be an *input* space and an *output* space for a generator G_θ parameterized by θ.

Recall $\mathcal{X}$ is the space of signals with length $N = 64000$ and $\mathcal{S}$ is the space of 2 stacked 256×256 images. Then given input $z \in \mathcal{Z}$, the output of parametric function G_θ on z is in $\mathcal{S}$, $G_\theta(z) \in \mathcal{S}$. Such, that with the image-to-signal transformation Φ^{-1} a signal $\Phi^{-1}(G_\theta(z)) \in \mathcal{X}$ can be obtained from $G_\theta(z) \in \mathcal{S}$.

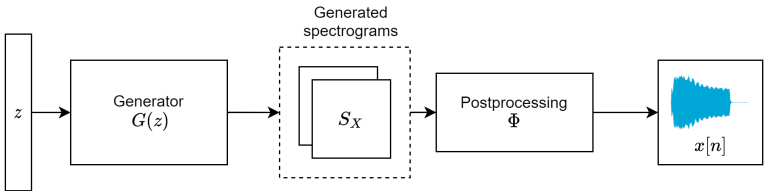

Figure 3.1: Generator inference: Given a $z \in \mathcal{Z}$ and a generator network G_θ, than inference of G_θ, $G_\theta(z)$ gives generated image-like representations S_x. Then image-to-signal transformation Φ^{-1} transforms the image-like representation into an audio signal $x = \Phi^{-1}(G_\theta(z))$.

3.3. Adversarial Training

To obtain parameters θ for $G_\theta(z)$ such that the produced signal $\Phi^{-1}(G_\theta(z))$ has characteristics similar to that of a target collection of audio $\mathfrak{D}$ we use an adversarial network.

3.3.1. Classifying Real and Fake

In order to obtain a trained generator. We introduce another parametric function D_ϕ that functions as a discriminator and is parametrized by ϕ. The discriminator is simply a classifier for 2 classes, *real* and *fake*, which can be trained according to a loss function for binary classification called cross entropy [9].

Let y be the true label and $\hat{y}$ be the predicted label, than the cross entropy loss function for binary classification $\mathcal{L}$ is:

$$\mathcal{L}(\hat{y}, y) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}))$$

This amounts to $-\log(\hat{y})$ if $y = 1$ and $-\log(1 - \hat{y})$ if $y = 0$. Suppose $\hat{y}$ is bounded between 0 and 1. If $y = 1$, the minimum of $\mathcal{L}(\hat{y}, 1) = -\log(\hat{y})$ is 0 at $\hat{y} = 1$. On the other hand, if $y = 0$ the mimimum of $\mathcal{L}(\hat{y}, 0) = -\log(1 - \hat{y}))$ is 0 at $\hat{y} = 0$. Figure 3.2 shows the cross entropy loss function $\mathcal{L}$ for both cases.

3.3.2. Turn-wise Training

The two functions, G_θ, D_ϕ are trained turn-wise. The function parameters the discriminator are trained on real samples labeled as *real* and on generated fake samples labeled as *fake*. Then, the discriminator parameters are frozen, meaning that they will not be updated in the next step, as we will now train the generator. The parameters of the generator are trained on points from the latent space labeled as *real*, where the loss function for the training is the prediction of the discriminator on the generated result. In other words, the generator's material is labeled as real to train the generator in fooling the

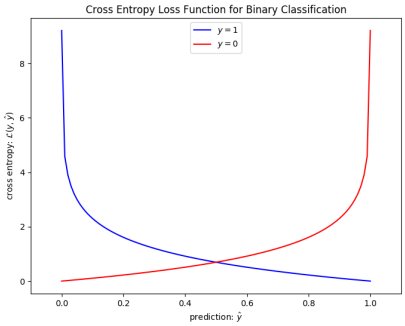

Figure 3.2: The figure shows the cross entropy loss function for binary classification $\mathcal{L}(\hat{y}, y)$, where y is the true label with two classes $\{0, 1\}$ and $\hat{y}$ is the predicted label bounded between 0 and 1. A minimum for $\mathcal{L}$ is achieved at $\hat{y} = 0$ if $y = 0$ and $\hat{y} = 1$ if $y = 1$.

discriminator, however the parameters of the discriminator are not influenced by this fooling.

In figure 3.3 a schematic for the adversarial training of the generator G_θ and discriminator D_ϕ, parameterized by θ and ϕ respectively, is visualized. Signals from our database $\mathfrak{D}$ are subject to the signal-to-image transformation Φ such that for signal $y \in \mathcal{X}$, $\Phi(y)$ is conform to the format of $s_y \in \mathcal{S}$, e.g. $\Phi(y) \in \mathcal{S}$. Φ turns signals into image-like representation, whereas the image-to-signal transformation Φ^{-1} turns these images into signals.

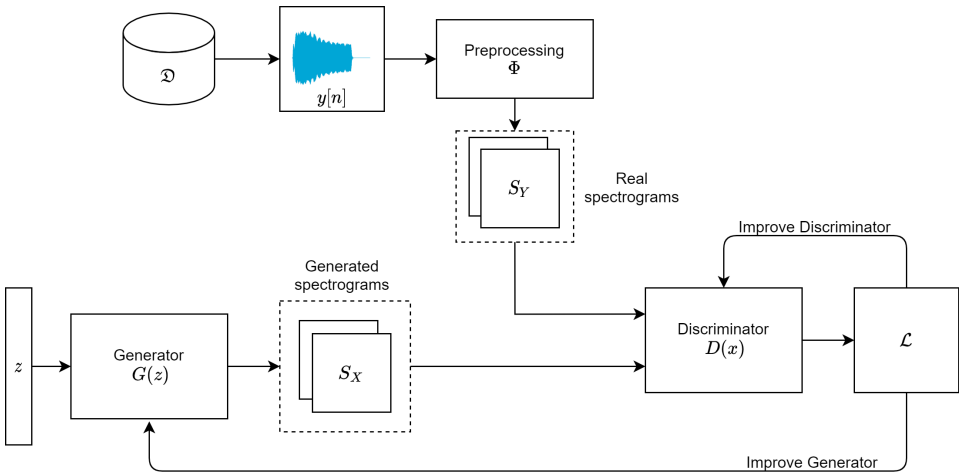

Figure 3.3: Adversarial training of generator G_θ: image-like signal representations S_Y, from real sound samples, are compared to synthetically generated image-like signal representations S_X by a discriminator D_ϕ. S_Y and S_X are labeled with $real$ and $fake$ classes. Then the combination of the two provides a labeled data set, against which D_ϕ is subject to supervised training. Given a $s_x \in \mathcal{S}$, resulting $D_\phi(s_x)$ is the classification of $real$ and $fake$ classes by trained D_ϕ. In its turn the performance of discriminator D_ϕ provides a training criterion for generator G_θ, with the target of increasing the resemblance between generated image-like signal representations S_X and real image-like signal representations S_Y.

Training Iteration One iteration of adversarial training of the discriminator and generator is produced by the following steps:

- First, m data points (audio signals) are sampled from our data set $\mathfrak{D}$,

$$\{x^{(i)} | 1 \leq i \leq m, x^{(i)} \in \mathfrak{D}\} \tag{3.1}$$

  signal-to-image transformation is applied to obtain m image-like signal representations from the audio signals

$$\{\Phi(x^{(i)}) | 1 \leq i \leq m, x^{(i)} \in \mathfrak{D}\} \tag{3.2}$$

  , such that $\Phi(x^{(i)}) \in \mathcal{S}\ \forall i \leq m$.

- m points, $\{z^{(i)}\}_{i=1}^{m}$, are sampled from a multivariate normal distribution $\mathcal{N}_{4096}(0, 1)$ ($z^{(i)}$ is a *standard normal random vector*), such that inferencing network G_θ on $z^{(i)}$, $\{G_\theta(z^{(i)})\}_{i=1}^{m}$, $G_\theta(z) \in \mathcal{S}$ is diversely distributed between and within batches.

- Now, the following loss functions are calculated and optimized w.r.t. the parameters accordingly,
  - $\min \mathcal{L}(D_\phi(\{\Phi(x^{(i)})\}_{i=1}^{m}), \mathbf{1})$ w.r.t parameters ϕ
  - $\min \mathcal{L}(D\phi(\{G_\theta(z^{(i)})\}_{i=1}^{m}), \mathbf{0})$ w.r.t parameters ϕ
  - $\min \mathcal{L}(D\phi(\{G_\theta(z^{(i)})\}_{i=1}^{m}), \mathbf{1})$ w.r.t parameters θ

3.3.3. Gradient Descent

In order to optimize functions G_θ and D_ϕ w.r.t. their parameters θ and ϕ descending or ascending along gradients calculated w.r.t. to the parameters introduces an algorithm for finding local minima, or maximima, given the function is differentiable. [41]

The algorithm is based on the notion that if a multi-variable function $f(x)$ is defined and differentiable in a neighborhood of a point a, then $f(x)$ decreases the fastest from a in the opposite direction of the gradient at a. Accordingly, the algorithm amounts to taking repeated steps in the direction of the negative gradient of f at a_n, $-\nabla f(a_n)$, updating the position of an each step with $-\gamma \nabla f(a_n)$, given a $\gamma \in \mathbb{R}$.

$$a_{n+1} = a_n - \gamma \nabla f(a_n)$$

Conversely, stepping in the direction of the gradient will lead to a local maximum of that function:

$$a_{n+1} = a_n + \gamma \nabla f(a_n)$$

3.4. GAN as Minimax Optimization

In order to illustrate how the adversarial training process theoretically [9] comes to an optimimum, we identify GAN as a minimax optimization. That is, a function is minimized according to one parameter and maximized according to an other parameter. Or in other words, minimizing the maximum of a function.

3.4.1. Function Space Minimax Optimization

Suppose we are able to optimize w.r.t. D and G directly in function space. [2] Let y be a from the data set. We wish $D(y)$ to be indicating real, or 1. Accordingly, an optimal discriminator would be obtained by,

[2] Upon till now, we have interpreted the discriminator and generator as parametric functions D_ϕ and G_θ. Accordingly optimizations have been done w.r.t. to the parameters. This does not withstand in the theorethical foundation of GAN [9].

$$\arg\min_D \mathcal{L}(D(y), 1) \tag{3.3}$$

Also output generator by the generator G should be indicated by the discriminator as fake, or $D(G(z)) = 0$. This introduces a second criteria for an optimal discriminator,

$$\arg\min_D \mathcal{L}(D(G(z)), 0) \tag{3.4}$$

Let D^* be an optimal discriminator, then following from equation 3.3 and 3.4,

$$D^* = \arg\min_D \mathcal{L}(D(G(z)), 0) + \mathcal{L}(D(y)), 1) \tag{3.5}$$

An optimal generator would be able to fool a discriminator into predicting the output of the generator, $G(z)$, as real, or $D(G(z)) = 1$. Optimal generator G^* would be obtained by minimizing the following w.r.t function G,

$$G^* = \arg\min_G \mathcal{L}(D(G(z)), 1) \tag{3.6}$$

Rewriting the minimization in equation 3.6 to a maximization gives us,

$$G^* = \arg\max_G \mathcal{L}(D(G(z)), 0) \tag{3.7}$$

At last, the term $\mathcal{L}(D(G(z)), 0)$ in equation 3.5 and 3.7 can be joined to obtain the minimax optimization for both optimal discriminator D^* and G^* simultaneously:

$$D^*, G^* = \arg\min_D \max_G \mathcal{L}(D(G(z)), 0) + \mathcal{L}(D(y)), 1) \tag{3.8}$$

The theorethical result in [9] shows that the improving of the minimax optimizatation in GAN resembles minimizing the Jensen-Shannon divergence [20] between the data and the model distribution. [3] Accordingly, the minimax converges to its equilibrium if both functions can be updated directly in function space [9].

3.4.2. Convergence in Parameter Space

Upon till previous section, we have interpreted the optimization within GAN w.r.t. parameters θ for G_θ and ϕ for D_ϕ. However, the theory described in previous section does not hold for optimizations in parameter space rather than function space.

In [9] the theoretical result rely on G and D being updated in the function space directly. In practice, G and D are approximated by neural networks with parameters θ and ϕ respectively. Accordingly, instead of optimizing the functions G and D directly, the minimax in equation 3.9 is optimized w.r.t. to the parameters θ and ϕ for generator G_θ and discriminator D_ϕ. This should give optimal parameters θ^* for G_θ and ϕ^* for D_ϕ, where ϕ^*, θ^* are obtained as follows,

$$\phi^*, \theta^* = \arg\min_\phi \max_\theta \mathcal{L}(D_\phi(G_\theta(z)), 0) + \mathcal{L}(D_\phi(y)), 1) \tag{3.9}$$

Using neural network to define the generator and discriminator and optimize in parameter space is cutting corners. Currently, there is neither a theoretical argument that GAN should converge when the updates are made to parameters of deep neural networks, nor a theoretical argument that GAN should not converge [8]. However, the excellent performance of multilayer perceptrons in practice suggests that they are a reasonable model to use despite their lack of theoretical guarantees [9].

[3] The Jensen-Shannon divergence is a rate for similarity between two data distributions. [20]

Summary

In this chapter we covered topics related to Generative Adverserial Networks

- Generative Adverserial Networks introduce a method to obtain a generative model by turnwise training of a generator, producing *fake* data, and a discriminator, classifying *real* data from a data set and *fake* data from a generator. (Section 3.1)
- A generator generates data by inferencing (figure 3.1) a generator function G_θ with a point from multi-dimensional space $z \in \mathcal{Z}$ to ”control” the generators behaviour. (Section 3.2)
- Generator G_θ are is trained adversarially (figure 3.3) with a discriminator D_ϕ to obtain a generative model for a data set by improving the parameters θ and ϕ according to a specified loss function $\mathcal{L}$. (Section 3.3)

Also, we covered theorethical results from [9] and the limitations of the theory when GAN is put to practice

- Optimizing GAN can be seen as a minimax optimization. (Section 3.4)
- When a discriminator and generator are improved directly in function space, [8] shows that improving the minimax optimization amounts to increasing similarity between the data set and the generator produced data. (Section 3.4.1)
- However, in practice these functions G and D are approximated by neural networks with parameters θ for G_θ and ϕ for D_ϕ. Currently there is no proof for convergence of the optimization in parameter space, but practical performance suggests reasonable usage of neural networks as function estimators. [9] (Section 3.4.2)

4

Neural Network as a Synthesizer

With a generative adversarial network we obtain a trained generator G_θ that is able to produce data that is similar to data in the training data set. Similar to that we are able produce sound with a synthesizer with specified controls, we are able to inference the generator G_θ with abstract controls given by a point $z \in \mathcal{Z}$.

However as of yet we have not given a structural architecture for how a generator G_θ is able to parametrically map points from a latent space $\mathcal{Z}$ to an image-like log mel spectrogram of shape $256 \times 256 \times 2$, nor how the discriminator D_ϕ parametrically maps the image-like signal representations to a class indicator of *real* or *fake*.

In fact, G_θ supposedly maps $\mathcal{Z}$ to $\mathcal{S}$,

$$G_\theta : \mathcal{Z} \mapsto \mathcal{S} \tag{4.1}$$

And, D_ϕ maps $\mathcal{S}$ to $(0, 1)$,

$$D_\phi : \mathcal{S} \mapsto (0, 1) \tag{4.2}$$

Overview

In this chapter we discuss the architectural choices for a middle c electric piano log mel spectrogram producing generator and a discriminator indicating real and fake spectrograms. We will briefly discuss the fundamental building block of neural networks, the neuron. After which we will discuss how a network of layers of neurons can be trained to be used as a function approximator [19]. At last the use of convolution neural networks will be demonstrated to attain to the use of the image representation for audio signals from chapter 2 .

4.1. Feedforward Neural Network as a Parametric Function

Feedforward neural networks, networks that form a *acyclic directed graph*[1], can be interpreted as function approximaters. They can be used to map a domain to a specific subspace in the co-domain by adapting the network parameters according to a specified loss function applied on the networks output and desired output [8]. In this section we will discuss how the parameters θ of a feedforward neural network F influence the function behaviour and how they are adapted to conform to the desired function approximation.

[1] The network has a directed flow from the input to the output with no cycles within the network.

4.1.1. Neuron

In artificial neural networks, the main building blocks, neurons can be defined as a mapping from $\mathbb{R}^n$ to $\mathbb{R}$, i.e. $f : \mathbb{R}^n \mapsto \mathbb{R}$. The neuron f takes a multivariable input x and gives a single output. The passing of $x \in \mathbb{R}^n$ through neuron f is the result of a composition of an **activation function** and a **weighted sum** with **bias**. It is exactly the *weights* and *biases* of all the neurons in the neural network that collectively constitute the network function parameters.

Weighted Sum Let $w \in \mathbb{R}^n$ be the weights for neuron f, than we call the calculation of the weighted sum in f the parametric function $\hat{f}$, such that:

$$\hat{f}(x; w) = \sum_{i=1}^{n} w_i x_i \tag{4.3}$$

Bias The bias of a neuron is the addition of a constant term to the weighted sum. This initialy gives us the parametric function $\tilde{f}$ with parameters $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$,

$$\tilde{f}(x; w, b) = \sum_{i=1}^{n} w_i x_i + b \tag{4.4}$$

Activation Function The activation function decides whether a neuron should be ”activated” or not by the value of the calculated weighted sum and addition of bias. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

A neuron without an activation function is essentially just a linear regression model. As the neurons then remains to be a combination of summations and constant multiplications. The identity activation function, for example, does not satisfy this property. When multiple layers use the identity activation function, the entire network is equivalent to a single-layer model.

When the activation function is non-linear, then a two-layer neural network can be proven to be a universal function approximator.[11] This is known as the *Universal Approximation Theorem* of neural networks [2]

In this study we consider the use of the following activation functions:

- **Sigmoid:** $sigmoid(x) = \frac{1}{1+e^{-}x}$
- **Hyperbolic Tangent**: $tanh(x) = \frac{2}{1+2e^{-2x}-1}$
- **Rectified Linear Unit**:$ReLu(x) = \max(0, x)$
- **Leaky Rectified Linear Unit**: $LeakyReLu(x; \alpha) = \max(\alpha x, x)$

[2] Under this assumption the use of neural networks is an adequate choice for modellinig the *generator* function and *discriminator* function in GAN covered in section 3.4.1.

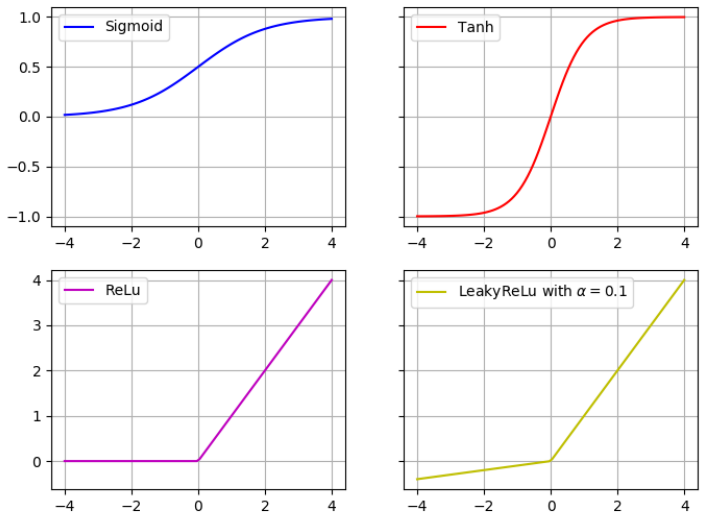

Figure 4.1: Four suitable activatition functions: Sigmoid (top left), Tanh (top right), ReLu (bottom left), and Leaky ReLu (bottom right).

The composition of a valid activation function σ and $\tilde{f}$ from equation 4.4 gives us the function definition of a neuron parametrized by weights and biases.

$$f(x; w, b) = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{4.5}$$

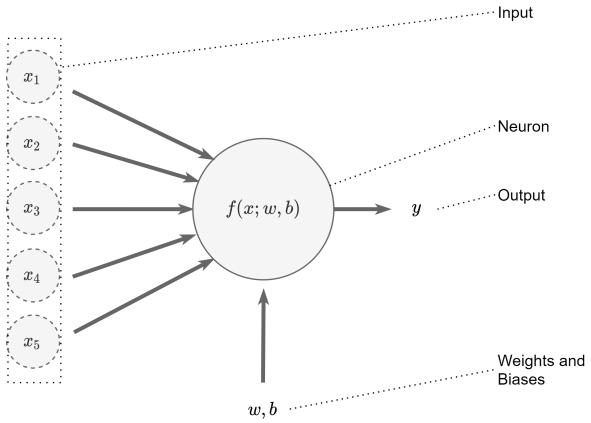

Figure 4.2: A schematic of neuron f. The neuron has input x_1, x_2, x_3, x_4, x_5 and output y. Parameters for the neuron are weights w and bias b.

If we redefine the neuron function to take inputs $x_0, x_1, \ldots, x_n$ by adding x_0 and setting it to 1, we get a function for a neuron as follows,

$$f(x; w) = \sigma\left(\sum_{i=0}^{n} w_i x_i\right) \tag{4.6}$$

Where $w_0 = b$. By setting this identity we are able to replace the bias, b with the 0th entry of the weights, w_0, since it is multiplied by $x_0 = 1$, i.e. $x_0 w_0 = b$.

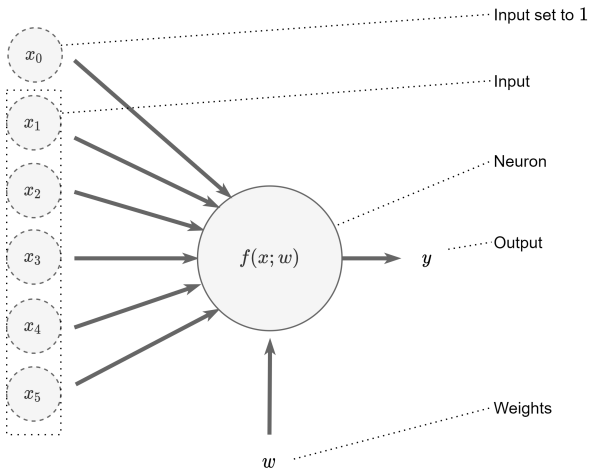

Figure 4.3: A schematic of neuron f without bias. The neuron has input $x_0 = 1, x_1, x_2, x_3, x_4, x_5$ and output y. Parameters for the neuron are weights w. The bias b is replaced by adding input $x_0 = 1$ and w_0.

4.1.2. Network

A feedforward neural network is a network of neurons which can be organized into multiple layers. The layer that receives external data is the input layer. The layer that produces the result is the output layer. In between them are zero or more hidden layers.

Between two neighbouring layers of neurons, multiple connection patterns are possible. They can be *fully connected*, with every neuron in one layer connecting to every neuron in the other. They can be pooling, where a group of neurons in one layer connect to a single neuron in the next layer, thereby reducing the number of neurons in that layer.

Networks where the connections and neurons form a *directed acyclic graph* are known as *feedforward networks*. That is, the network consists of vertices (neurons) and directed edges (connections) such that following the directions of the connections will never form a closed loop.

Layers of Neurons A layer of neurons can be written as the multivariate function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$, where m is the amount of neurons in the layer and n the amount of neurons in the layer before it. [3]

Then for $0 < j \leq m$, f_j is the function of the jth neuron of the layer. Then given the weights and biases for neuron j with corresponding w and b, the function definition for neuron j amounts to,

$$f_j(x; w) = \sigma\left(\sum_{i=0}^{n} w_i x_i\right) \tag{4.7}$$

Where w_i represents the weight for input i at neuron j. Note that we use a similar encapsulation of the bias in the weights in equation 4.7 as in equation 4.6.

Now, we have a value for weights for each combination of $0 \geq j \geq m$ and $0 \geq i \geq n$, inciting us to write this as a $m \times n + 1$ matrix W. Where W_{ji} is the weight that the output of neuron i (or bias if $i = 0$) contributes to the weighted sum of neuron j. If there is no connection between i and j, then W_{ij} equals zero.

$$f_j(x; W) = \sigma\left(\sum_{i=0}^{n} W_{ji} x_i\right) \tag{4.8}$$

[3] or n is the amount of inputs if f is an input layer.

The multivariate output of layer-wise function f parametrized by W amounts to all the individual neuron outputs of f_j. For $x \in \mathbb{R}^n$,

$$f(x;W) = \Big(f_1(x;W), f_2(x;W), \dots, f_m(x;W)\Big) \tag{4.9}$$

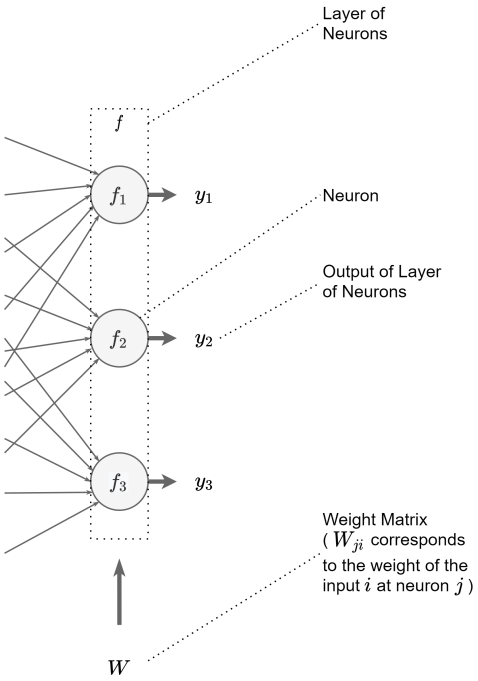

Figure 4.4: A schematic of a layer of neurons f with 3 neurons: f_1, f_2, f_3. Each neuron has 5 inputs from the previous layer.

Feedforward Network A feedforward network can be considered as a stack of layers of neurons. If we would define the network function F that has an input in $\mathbb{R}^N$ and an output in $\mathbb{R}^M$, than $F : \mathbb{R}^N \mapsto \mathbb{R}^M$ amounts to the composition of layer-wise functions f^k, where k is the layer. We write θ as the *network parameters*, where θ^k corresponds to the neuron weights W at layer k as in equation 4.9. Then,

$$F_\theta = f^k_{\theta^k} \circ f^{k-1}_{\theta^{k-1}} \circ \dots \circ f^1_{\theta^1} \tag{4.10}$$

or,

$$F_\theta(x) = f^k_{\theta^k}(f^{k-1}_{\theta^{k-1}}(\dots)) \tag{4.11}$$

We write the paramaters θ of a network F in subscript, so F_θ instead of $F(\bullet;\theta)$, as we will refer to network functions without input often.

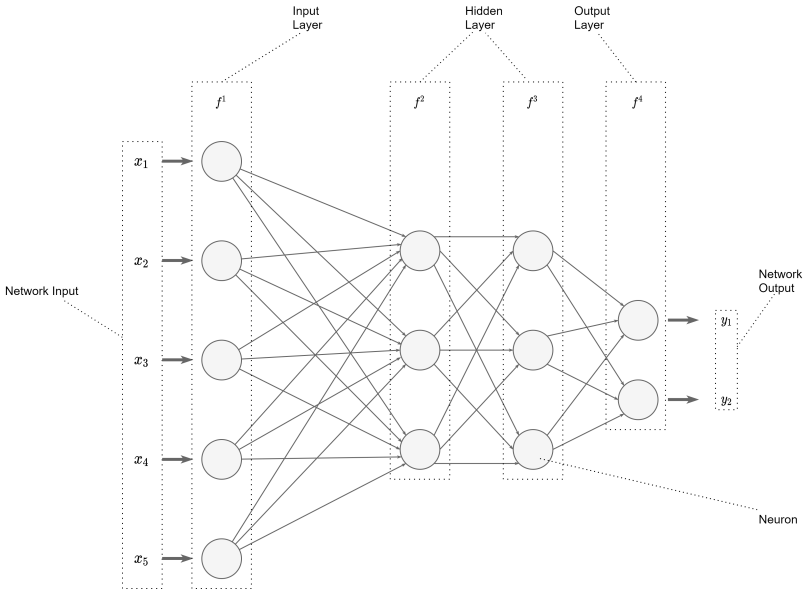

Figure 4.5: A schematic of a feedforward neural network. The network has 5 inputs, 2 hidden layers of three neurons and an output layer with two neurons. This is a fully connected network as every neuron is connected to every neuron in the previous layer.

4.1.3. Parameter Learning

Learning is the adaptation of the network to better handle a the function approximation by considering sample observations and desired outputs. Learning involves the adjusting of network weights of to improve the accuracy of the result. This is done by minimizing the observed errors between the network result and the desired output. Learning is complete when examining additional observations does not contibrute to reducction of the error rate. Even after learning, the error rate typically does not reach 0. A network serves as a *function approximator* after all. If after learning, the error rate is too high, the network typically must be redesigned or adjusted to be improved.

A network F has a multivariable parameter θ, which represents all the weights and biases for each neuron at each layer.

Given a x, the learning of network F amounts to minimizing the value of some function that takes the result of $F_\theta(x)$ and the desired outcome y.

Backpropagation is a method used to adjust the connection weights to compensate for each error found during learning. The error amount is effectively divided among the connections. Technically, backpropagation calculates the gradient (the derivative) of the cost function from a network output and a label with respect to the weights. [30] This allows the use of gradient descent algorithm briefly described in 3.3 to be used with neural networks.

4.2. Image-like Signal Representation Convolutional Neural Network

Fully connected feedforward neural networks, that is layers of neurons that are all connected to each other, can be used to learn features and classifications. However, the architecture is generally impractical for larger inputs such as images[4]. Convolutional neural networks aim to solve this problem by making effective use of hierarchical information of the position of elements inside an image.

4.2.1. Image Convolution

In image processing, convolution is a general purpose filter effect for images. It is used to modify the spatial frequency characteristics of an image. Convolution is performed by applying simple matrix op-

[4] Or, image-like signal representations

erations segmentwise on images. These filters can be used to extract information, or feature maps, from images.

In CNNs the filters are the neurons in the networks and the numerical values in the filters are the weights. The CNN learns important weights for the kernel convolution as result from a machine learning objective. This way it learns to extract important features from images specifically aimed at solving the corresponding machine learning task. In figure 4.6 the information extraction from images with convolutional filters is shown serving a specific classification task.

4.2.2. Network Architecture

Training and designing a convolutional neural network from scratch involves a lot of decision making. This is regarding both the architecture and learning hyperparameters [5]. Design processes behind CNN architectures are often based on heuristics.

The use of convolutional neural networks with GAN in this study is mainly based on the ideas from DCGAN [29], but adapted to attain to the image shape specified in chapter 2.

CNN architectures are often designed to reduce the dimensionality in order to make a classification. This is conform to the behaviour of discriminator D_ϕ, where the network aims to classify *real* and *fake* images. However, convolutional operations can also be combined with upsampling in order to create a CNN architecture that maps a smaller dimensional space to a larger one. This is conform to the behaviour generator G_θ, where the network generates image-like signal representations from a smaller dimensional space, the latent space.

Conventional architecture of convolutional neural networks, or CNNs, can (often) be dissected into three segments: **input**, **convolution**, and **output** segment.

The **input segment** of a CNN consists predominantly of a fixed size input layer. In some cases reshaping operations, such as flattening, or restructuring images, are applied.

The **convolution segment** is characterized by the use of convolutional layers and up and down sampling operations.

Passing of a convolutional layer amounts to the abstraction of a feature map of the image from convolutional operations. The feature map has a shape of (n, h, w, c). Where n is the number of inputs, h is the height of the feature map, w is the width, and c the amount of channels, or sometimes refered to as *depth*. In contrast to fully connected neural networks, where each neuron receives input from all the neurons in the previous layer, in a convolutional layer, each neuron receives input from a restricted area of the previous layer, which corresponds to the size of the convolution kernel size.

Another important element used in CNNs are pooling, or downsample, layers. These are characterized by their ability to reduce the dimensions of data. In other words, the pooling layers shrink the image by mapping a section of an image, typically 2 by 2 pixels, to a single pixel. e.g. this can be done by taking the average of the pixels (average pooling), or the maximum value (max pooling). Upsampling layers are opposite to the pooling layers in that they increase the dimensions of the data. E.g. an image of 32 x 32 is upsampled to 64 x 64. The interpolation type for the newly generated pixels can be bilinear, or nearest.

The **output segment** of a CNN transforms, or classifies, the feature maps obtained from the convolution segment into a desired output. In the case of a CNN classifier, the feature maps are typically classified with a collection of fully connected layers, reshaping, and activation layers. When the CNN is used to produce images from a smaller dimensional space, the feature maps are used to create material of the desired structure. e.g. features maps are fully connected to a $32 \times 32 \times 3$ output, an RGB image with a width and height of 32.

[5] In machine learning, hyperparameters are parameters whose value is used to control the learning process, such as the amount of hidden layers, the convolution kernel size, or the leaky ReLu activation slope.

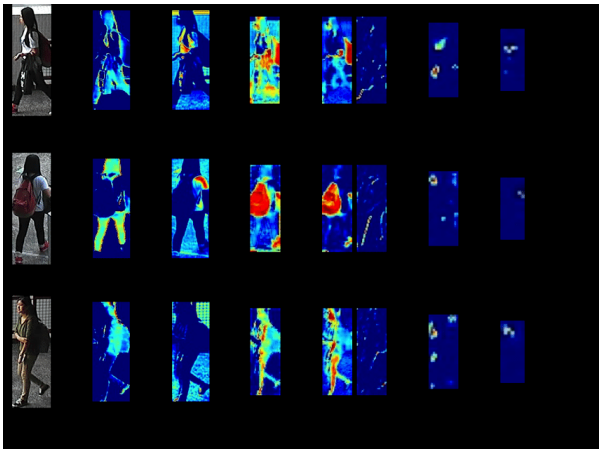

Figure 4.6: Visualization of feature maps learned by a CNN classification network. The classification network is a person re-identification (re-ID) [33], which aims at spotting a person of interest across multiple camera views. Each row represents some typical feature maps from low-level to high-level of a person image. First row: anchor image; second row: positive image; last row: negative image. It demonstrates how the convolution kernels that process images can be used to learn to extract specific feature maps.

Discriminator The architectural choices for the discriminator network D_ϕ are mostly borrowed from DCGAN [29], specified for $256 \times 256 \times 2$ images, a stack of 2 greyscale images of width and height 256 pixel.

$$D_\phi : \mathcal{S} \mapsto (0, 1) \tag{4.12}$$

D_ϕ maps the image-like signal representation space $\mathcal{S}$ to a value in $(0, 1)$. Indicating the predicted class of images from generator G_θ, with **1** and dataset $\mathfrak{D}$, with **0**.

Figure 4.7 shows a schematic for discriminator network D_ϕ. The network is build from convolutional blocks, which contain the following elements:

- Convolution layer with a kernel size of 3 by 3.
- Batch normalization layer [12] with momentum of 0.7 to stabilize the learning by normalizing the input to each unit to have zero mean and unit variance. This is omitted in the first block, similar to [29].
- LeakyReLu [38] activation with the slope of the leak at 0.2. [29]
- Dropout layer [35] with dropout rate 0.2 to combat overfitting.
- Average pooling layer to downsample the feature maps.

The network consists of 6 of such blocks with doubling its amount of filters each block and halfing its dimensions. Furthermore, the 6th convolution block outputs a shape of $4 \times 4 \times 256$, which is flattened, fed into a fully connected layer of width 128 with an LeakyReLu activation (also leak at 0.2), and then fed into a single sigmoid output, bounding the output between 0 and 1.

At last, the gaussian noise is added to the input to combat overfitting. [39]

Generator Similar to the discriminator, the architecture for the generator is also borrowed from DCGAN [29], but adapted to take input of a 4096 sized array and output $256 \times 256 \times 2$ images. Generator G_θ takes input z from input space $\mathcal{Z}$ and outputs a s_x in image-like signal representation space $\mathcal{S}$. This results in the function definition for G_θ,

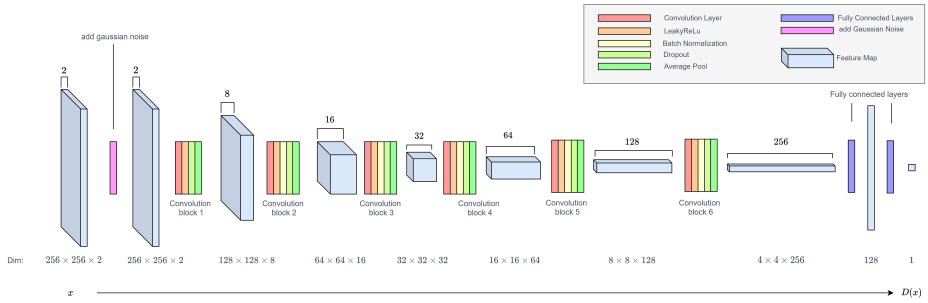

Figure 4.7: Discriminator network D_ϕ: D_ϕ is a convolutional neural network that takes as input data points of size $256 \times 256 \times 2$ and outputs a class from $\{real, fake\}$. Each layer in D_ϕ consists of a convolution layer, a downsampling operation (average pooling), dropout [35], batch normalization (except for block 1), and a leakyReLu activation. Two fully connected layers are used to classify the feature maps and are fed into a single sigmoid output. Gaussian noise is added to prevent overfitting at before convolution block 1.

$$G_\theta : \mathcal{Z} \mapsto \mathcal{S} \tag{4.13}$$

Figure 4.8 shows a schematic for generator network G_θ. Similar to the discriminator network, the generator is build from convolutional blocks, which contain the following elements:

- Convolution layer with a kernel size of 3 by 3.
- Batch normalization layer [12] with momentum of 0.7 to stabilize the learning by normalizing the input to each unit to have zero mean and unit variance. This is omitted in the last block, similar to [29].
- ReLu [38] activation. [29]
- Up sampling layer that doubles the dimensions of the feature maps each block.

At last, the network transforms the last convolution block to the output of $256 \times 256 \times 2$ after which tanh activation is applied.

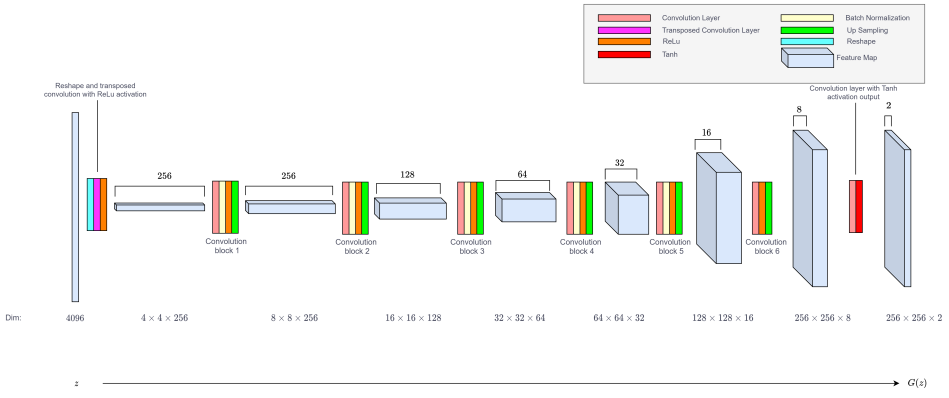

Figure 4.8: Spectrogram generator network G_θ: G_θ is a convolutional neural network that takes as input from $\mathcal{Z}$ and outputs an image-like data point with dimensions $256 \times 256 \times 2$. Each layer in G_θ consists of a convolution layer and an upsampling operation.

Size of the Latent Space Specification of input space $\mathcal{Z}$ can vary. In order to effectively transform the flat-array input of shape 4096 to a structure of shape $4 \times 4 \times 256$, the shape of the first convollutional block we us a transposed convolution layer. Transposed convolutions are generally used when there us a need for a transformation going in the opposite direction of a normal convolution [4]. Since the first layer of our network maps to $4 \times 4 \times 256$, as seen in figure 4.8, we consider dimensions for $\mathcal{Z}$

of similar magnitude, under the assumption this is helpfull. A lower dimension of $\mathcal{Z}$, requires a more compressed representation, a higher dimension gives more freedom to the network. Specification of the latent dimension has influence on the training progression, and could be subject to improvement and is often chosen on heuristics.

Optimization Algorithm The networks G and D are trained according to training schematic described in section 3.3. `ADAM` [17], or *Adaptive Moment Estimation*, is a method for stochastic gradient descent that we use to find optimal parameters ϕ and θ for network D and G, respectively. [29]

Summary

In this chapter we covered topics related to feedforward neural networks

- A feedforwared neural network F is a composition of layers of neurons. The weights in the weighted sum and biases in each neuron at each layer constitute the networks parameters. (Section 4.1)
- Neurons are the building blocks of neural networks. A neuron is a function f_i (ith neuron of a layer f) mapping a multivariable input to a single output. The output is result from a weighted sum with bias, after which a non-linear activation function is applied. (Section 4.1.1)
- With non-linearity in the activation function, a neural network can be proven to be a universal function approximator. [11] (Section 4.1.1)

Following this we described the network architectures for our generator G and discriminator D in use:

- In image processing, convolution is a general purpose filter effect for images that can be used to modify spatial frequency characteristics of an image. (Section 4.2.1)
- Convolutional neural network use image kernel convolution where the values in the filters are considered as the weights of the network. Following this the network learns to extract information from images by applying the filters. This way, CNNs make use of the hierarchical information of position of elements inside an image. (Section 4.2.2)
- Discriminator D is a convolution neural network that has an input of shape $256 \times 256 \times 2$ coherent to the image-like representation of audio signals from $\mathcal{X}$ from chapter 2. The output of D is a value between $(0, 1)$ indicating 1 with the image predicted as *real* and 0 as *fake*. (Section 4.2.2)
- Generator G is a convolutional neural network that takes input of size 4096 representing the dimension of latent space $\mathcal{Z}$. The output of G is of shape $256 \times 256 \times 2$, similarily as the image-like representation in chapter 2. (Section 4.2.2)

5

Qualitative Performance

Before any training of the generator network the output is considerably noisy, as seen in figure 5.1. Little structural patterns are visible. [1]

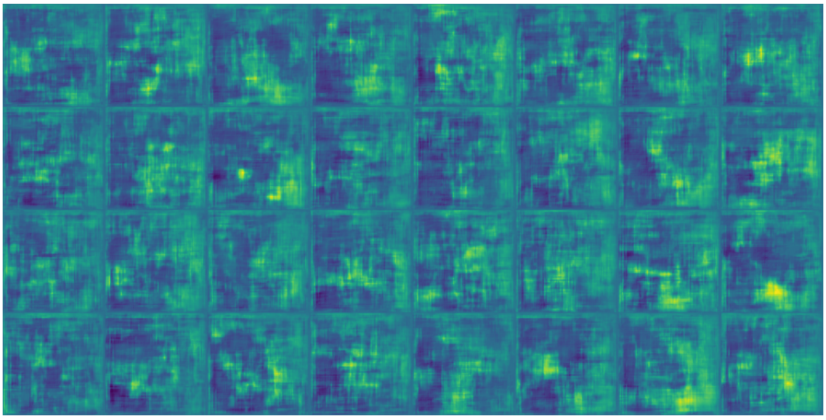

Figure 5.1: 32 (8 by 2) generated images from inferencing generator G_θ on 32 different latent points **before** any training. The vertical axis wihtin each log mel spectrogram is the frequency axis and the horizontal the time axis.

To test the adversarial training of GAN from figure 3.3 of our generator network, we train a generator and discriminator adversarially for 3000 iteratations on a collection of sounds containing electric pianos playing the note middle c. That is, 3000 batches of image-like signal representations are generated, or extracted from the dataset with the signal-to-image transformation from 2.3, and classified as real or fake by the discriminator.

Overview

In this chapter we discuss the the audio collection that is used to train generator G_θ as described in section 3. Furthermore, we inspect the resulting log mel spectrograms from the methods qualitatively and compare them with the log mel spectrograms from the audio collection. At last, we cover the results of latent space interpolation with a trained generator G_θ similarily as the latent space intepolation demonstrated in [9]

5.1. Training Database

The result of a trained GAN, as in many machine learning problems, is highly dependent on the input data. The methods are implemented using a subsection of the NSynth data set [5]. The NSynth data

[1] There are some incidental and negligible effects of the convolutional neural networks components, however, that refrain it from being truely random.

set contains 305,979 recordings of musical notes. It is created from 1006 acoustic and digital instruments originating from commercial sample libraries. Each recording is 4 seconds long, has a sample rate of 16 kHz, and has a unique combination of pitch, velocity, and timbre.

To minimize the scope of the methods in this study, a subset of the dataset is taken where the pitch is limited to midi note 60, or middle c, and the instrument type is limited to electronic pianos [2].

Despite the variety within different types of electronic pianos (rhodes, wurlitzers, digitally sampled pianos), their main characteristics are relatively similar. The electronic piano has structured harmonic and time information, electronic pianos decay similarily and their harmonic distributions only differ slightly, which makes it easier to recognize patterns and promises therefore better a functioning method. Successes in training GAN on electronic pianos, validates the investigation of training GAN on more complex audio.

The restricted audio collection we use contains 658 electronic pianos playing a middle c. It should be noted that using a more varied data set, with different notes and different instruments, could yield a similar result with sufficient conditioning on the different pitch and timbre classes during the adversarial training process as refered to in [24] [3].

We refer to the data set in use as $\mathfrak{D}$, where audio signals in $\mathfrak{D}$ are refered to with y (in contrast to generated signals x) and a batch of audio signals from $\mathfrak{D}$ with Y. Consequently, the data set is a subspace in the signal space $\mathcal{X}$, i.e. $\mathfrak{D} \subset \mathcal{X}$ s.t. for $y \in \mathfrak{D}$, the image-like signal representation $s_y = \Phi(y) \in \mathcal{S}$. A batch of image-like signal representations of $Y \subset \mathfrak{D}$ is refered to as S_Y.

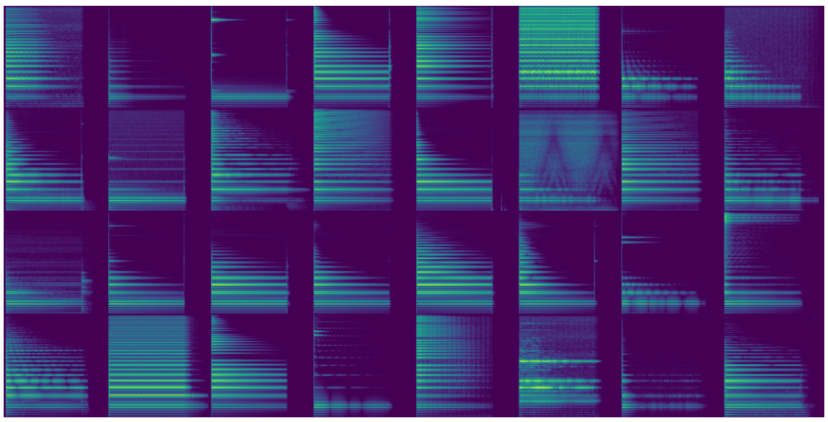

Figure 5.2: Log mel spectrograms of 32 electronic piano samples from the training data set. The vertical axis wihtin each log mel spectrogram is the frequency axis and the horizontal the time axis.

5.2. Generated Log Mel Spectrograms

In this section we will investigate the training process of the generator network by looking at the generated log mel spectrograms. While our generator network produces an image-like signal representation consisting of a stack of 2 256×256 images, one representing log mel spectrograms and the other respresenting mel instantaneous frequencies as derived from chapter 2, we limit visual inspection to the log mel spectrograms. After all, the spectral information is most prominent audible perception in the reconstructed audio signal.

[2] The selection from NSynth yields the following tags: note *c*, instrument family, *keyboard*, instrument source *electronic*. The resulting data set is a combination of different electronic piano sounds, including rhodes, wurlitzers and digitally sampled pianos.

[3] More on variety of audio and conditional GAN in chapter 7

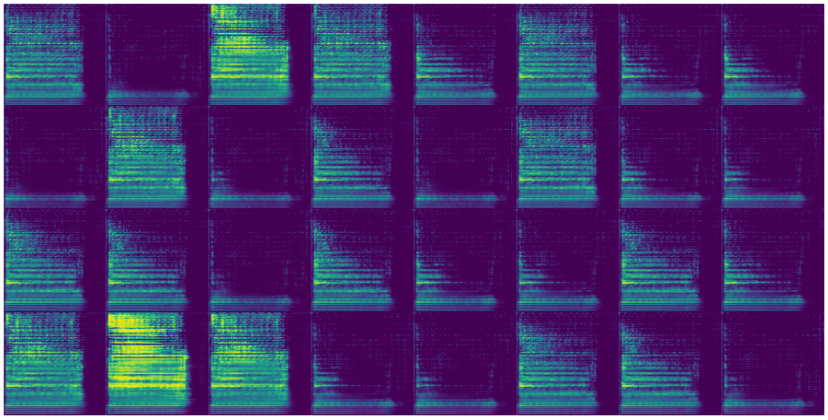

Figure 5.3: 32 generated images from inferencing generator G_θ on 32 latent points **after** training of GAN. The vertical axis wihtin each log mel spectrogram is the frequency axis and the horizontal the time axis.

In figure 5.3 we see log mel spectrograms from a trained generator. We see clear structural similarities to log mel spectrograms originating from the training data set displayed in 5.2. The generated log mel spectrograms appear to have similar fundamental frequency and indicate a note stopping at three quarters of the time, i.e. 3 seconds. Furthermore, we see diversity in harmonic distribution and decay similar to what is seen in 5.2.

The progression made in modelling the log mel spectrograms of electronic piano notes from $\mathfrak{D}$ by generator G_θ, as seen from the before (5.1) and after training (5.3) results, is huge. However, the generators result often contain artifacts such as noisy structures in high frequencies, blurry areas in the middle to high frequencies and wobbly partial amplitudes. The artifacts are demonstrated in figure 5.4.

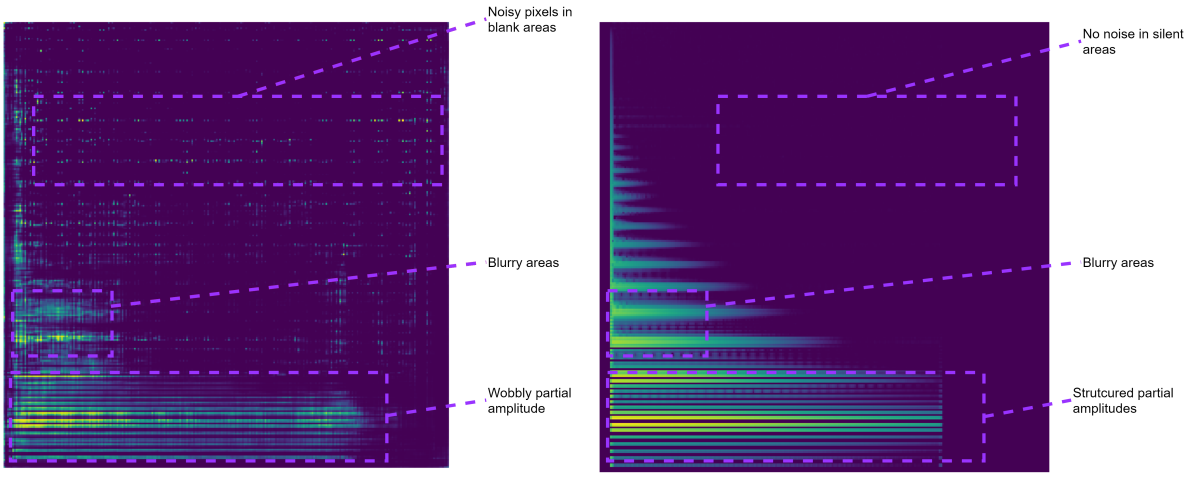

Figure 5.4: This figure shows a result from the generator (left) compared to a similar training data set sample (right). General form and structure of a log mel spectrogram is learned by the generator, however the generator result has artifacts such as noisy structures in high frequencies, blurry areas in the middle to high frequencies and wobbly partial amplitudes.

5.3. Latent Space Exploration

The position of a point in the latent space $\mathcal{Z}$ determines the output of generator G_θ. By backtracking latent points from instances of the generator output, qualitative insight in the generators range can be achieved.

Figure 5.5 shows three generated log mel spectrograms $G_\theta(z_1), G_\theta(z_2), G_\theta(z_3)$, with $z_1, z_2, z_3 \in \mathcal{Z}$. Clearly, the log mel spectrograms show similarity in fundamental frequency, but variation in timbre. That is different distributions of frequencies. Figure 5.5 (c) has more high frequencies than figure 5.5

(b). Also the note in 5.5 (b) appears to fade away faster than 5.5 (c) and 5.5 (a). This suggests the generator projects at least some variety from the latent space on the image-like signal representation $\mathcal{S}$.

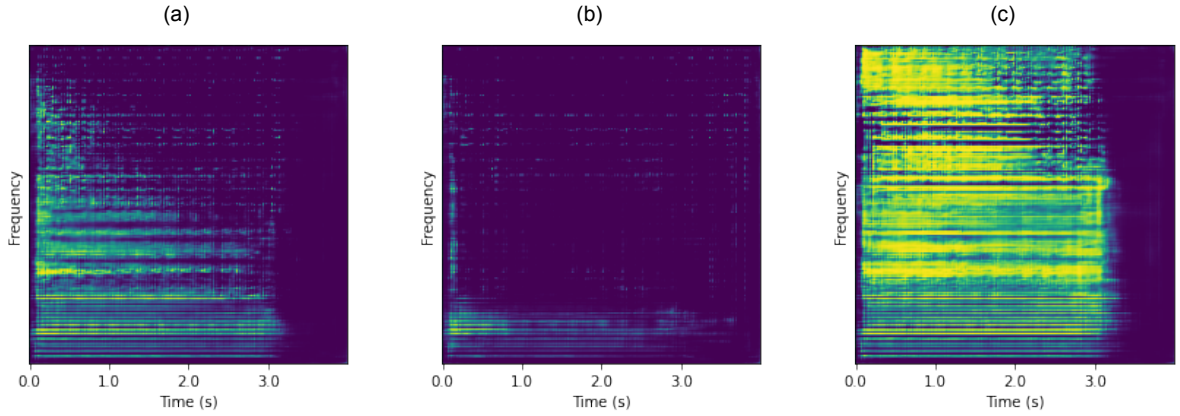

Figure 5.5: Three generated spectrogram show different timbres, but similar pitch; (a) $G_\theta(z_1)$; (b) $G_\theta(z_2)$; (c) $G_\theta(z_3)$

Traditional Audio Interpolation

In traditional audio interpolation, the process is essentially overlapping waveforms, or in dynamic context a cross-fade. In log mel spectrogram this operation yields the overlap of two spectrograms. In figure 5.6 we see two images of log mel spectrograms spectrograms of our dataset overlapping, which is the result of overlapping the log mel spectrograms.

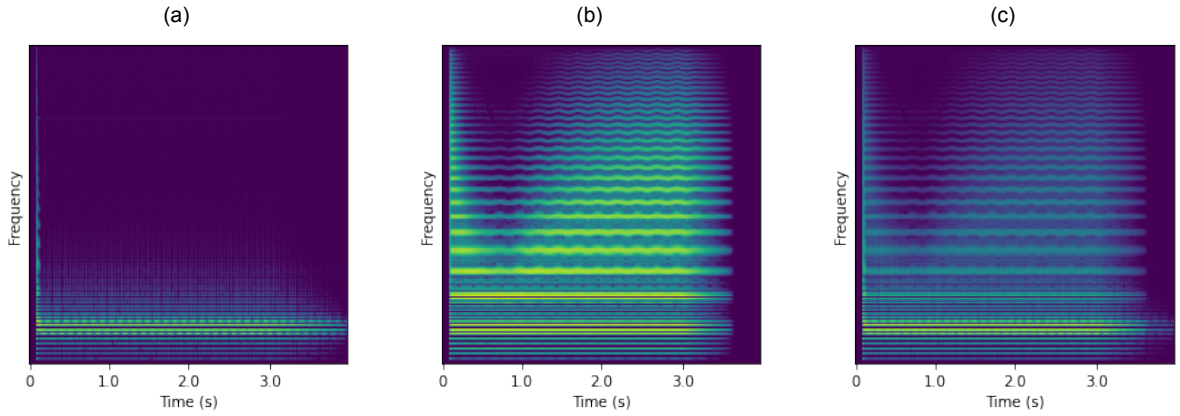

Figure 5.6: Two log mel spectrograms and their sum. An example of traditional audio interpolation. The log mel spectrogram in (c) is obtained from the sum of log mel spectrogram (a) and (b).

Latent Space Interpolation

As seen in [9, 29], latent space interpolation introduces a different approach to interpolating between two points. GAN introduces a novel method of interpolating between two points in the codomain of the generator. As seen in figure 5.7, [29] shows a visual turn of faces in images by latent interpolation. The latent space interpolation is a key element in the motivation to use GANs for audio synthesis.

Figure 5.7: Latent space interpolations along two axes between random samples transforms facial poses in a face image generating GAN. [29]

Given the latent points z_1, z_2, z_3 from the generated log mel spectrograms in figure 5.5, a 2 dimensional grid can be constructed from the three points. Inferencing the network G on this grid gives us a projection of the latent space grid in $\mathcal{Z}$ on the codomain $\mathcal{S}$ as seen in figure 5.8.

It shows the ability of the generator to morph between log mel spectrograms. Specifically in this grid the morphing is found in the quantity of high frequency information, differentiating its timbre.

A morph along these axes visually yields similarity to the a combination of a *low pass filter* filtering out high frequencies, and shortening *decay time* of the notes.

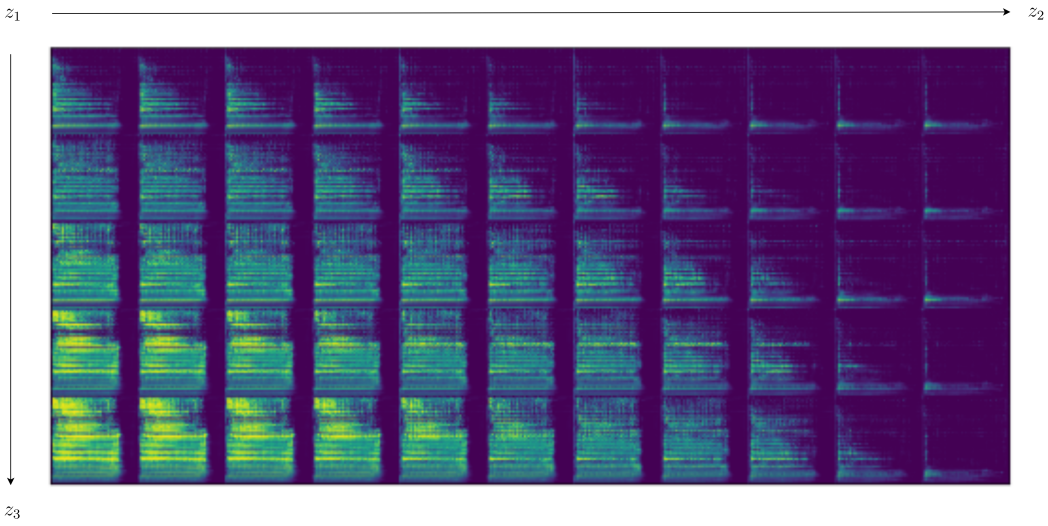

Figure 5.8: Latent space visualization of a 2D grid produced from three points $z_1, z_2, z_3 \in \mathcal{Z}$. The points 2D grid in the latent space are fed as input to the generator G_θ. Every log mel spectrogram in the figure corresponds to an output of $G_\theta(z)$ where z is positioned on the latent space grid similar to how the log mel spectrogram is positioned in the figure.

Summary

In this chapter we covered the specifications for the collection of audio:

- The collection of audio is a restricted version of the NSynth [5] dataset. NSynth contains musical notes playing for 3 seconds and 1 second of decay. The recordings are sampled at 16 kHz. The musical notes in this study are limited to electronic pianos playing middle c. (Section)

Also, we investigated the generated log mel spectrograms generated by the generator G_θ after a training procedure:

- Seen by the progression before and after the training of generator in figure 5.1 and 5.3, the generator has learned to produce structural similarities to the log mel spectrograms in the training data set. This is mostly visible in the presence of fundamental frequency coherent to middle c. The generator does produce artifacts such as noisy and blurry areas, as seen in figure 5.4. (Section 5.2)

At last, we inspected the effects of latent space interpolation as seen in [9] on the log mel spectrograms.

- The effect of latent space interpolation is clearly visible. In figure 5.8 interpolation along axes yields audibly and visually similarity to the a combination of a *low pass filter* filtering out high frequencies, and shortening *decay time* of the notes. (Section 5.3)

6

Quantitative Perfomance

As we have seen in the previous chapter, we are able to visually evaluate the generators performance. A recurring criticism of GANs however is the lack of robust and consistent evaluation methods. There is no single valuable way to determine the performance of GAN other than observing the generator's output and qualitatively assign a value with respect to the specific task.

Ironically, it is a problem that is present in any assesment of a generative model and one of the problems GAN solves. No requirement or objective for the generators result needs to be specified in GAN to obtain high quality example similar to the data set. This solves the lack of supervised quantifications of certain domains. It is difficult to compactly quantify audio, to evaluate generated instances on similarity. Audio signals can have similar characters, but sound completely different. We can measure characteristics of audio, but the audio is not defined by them. GAN's criteria is that the audio yields similarity with the collection of audio it is trained on. No engineered audio characteristic has to be specified in order to train GAN.

However, as the collection of audio $\mathfrak{D}$ is restricted in its variety there are audio characteristics that are commonly present within every signal. For example, we have restricted the collection of audio to only contain pitched notes of middle c. Accordingly, the generator should be producing audio signals with a pitch of middle c.

This idea can be extended further. Viewing the audio signal characteristics as numeric properties describing the signal, rough comparisons between collections of signals can be made.

Overview

In this chapter we measure and compare audio qualities of generated signals and that of the audio collection $\mathfrak{D}$. Signals generated by G_θ and signals from $\mathfrak{D}$ are compared w.r.t. to a set of audio characteristics derived from low level signal descriptors.

Furthermore, we quantify the performance of G_θ throughout adversarial training. A performance metric based on the audio characteristics between two sets of audio signals is calculated through out the training procedures to quantify the effect of training on the generated signals in comparison to the singals in $\mathfrak{D}$.

6.1. Audio Signal Characteristics

In order to evaluate audio, a quantitative method for audio description is required. Sound can be subjectively assigned a list of qualities, such as loud or soft, and bright or dark. Acquiring meaningfull statements about these high level decriptors however, would require a lot of human resources in assessing these qualities. To evaluate audio efficiently, quantitative comparisons of *low level descriptors* and accompanying *global functionals* are used instead.

6.1.1. Low Level Descriptors

Acoustic *low level descriptors* (LLDs) of signals are often used in speech analysis [31]. They provide directly measured descriptions of the signals. Signal LLDs are calculated frame by frame, resulting in a time dependent contour that describes the signals property throughout time.

For signal $x \in \mathcal{X}$ we can define a LLD d by,

$$d : \mathcal{X} \to \mathbb{R}^m \tag{6.1}$$

Where integer m with $1 \leq m \leq N$ is the amount of frames used to analyze signal x and $N = 64000$ is the length of x.

For example, figure 6.1 shows the pitch contour of two audio signals.

Figure 6.1: Pitch contour of two audio signals.

6.1.2. Global Functionals

The low level descriptors, such as pitch contours as seen in 6.1, provide time dependent details on the audio characteristics. These sequences are still difficult to compare directly, e.g. a shift along the time axis of two similar contours, could be interpreted as a large difference. In order to extract valuable quantitative information from the audio signals, we need a comparison that is invariant to these timing differences. A simple way to circumvent this, is to extract these numeric values from the contours with *global functionals*.

Insight in the characteristics of audio signals can be obtained by applying *global functionals* on measured *low level descriptors*. This allows us to obtain information on batches of generator produced signals and comparing them to batches of signals from the audio collection $\mathfrak{D}$. While neglecting the time dependency of the LLD contours, applying global functionals on the calculated LLDs of a signal provides a robust brute force method for audio signal analysis.

Let $N \in \mathbb{N}$. Then, we define $\mathcal{R}^N$ as a space that is the union of $\mathbb{R}, \mathbb{R}^2, \ldots, \mathbb{R}^N$, or $\cup_{1 \leq i \leq N} \mathbb{R}^i$. This can represent the space of real number time series of arbitrary length from 1 to N. Or in other words, the low level descriptor contours independent of the size of the time steps used for the calculations.

We define a *global functional* f that can be applied to any time series of length smaller than N and maps to a single real number in $\mathbb{R}$.

$$f : \mathcal{R}^N \mapsto \mathbb{R} \tag{6.2}$$

This allows the composition of a low level descriptor d and a global functional f, independently of the framing used in calculation of d.

$$f \circ d : \mathcal{X} \mapsto \mathbb{R} \tag{6.3}$$

An **example** of a global functional f, could be the the **mean** of entries in $\mathbf{a} \in \mathbb{R}^m$, for $m \leq N$.

$$f(\mathbf{a}) = \frac{1}{m}\sum_{i=1}^{m}\mathbf{a}_i \tag{6.4}$$

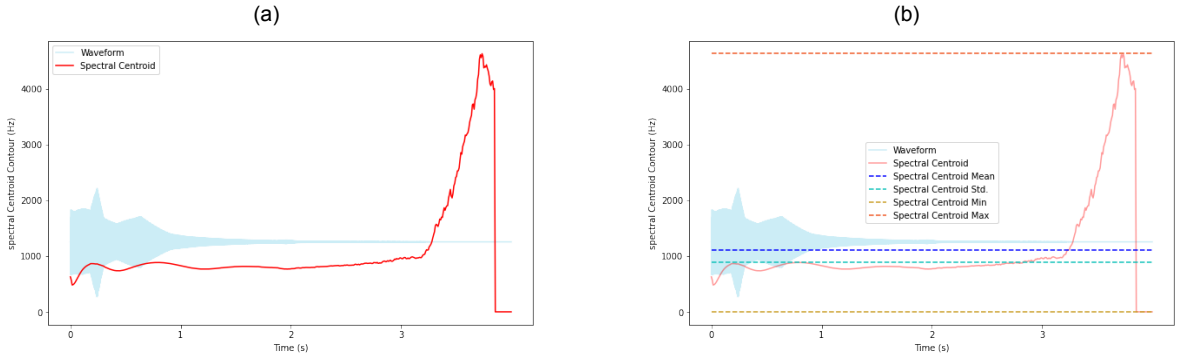

Figure 6.2: Spectral centroid contour of an audio signal and the *mean*, *std.*, *min*, *max* of the contour. This demonstrates the process of calculating the numeric audio characteristics. From a signal $x \in mathcalX$ a low level descriptor d is calculated which gives us the contour (red). Than, the global functionals f_1, f_2, f_3, f_4 representing *mean*, *std.*, *min*, *max* are calculated on the contour resulting in 4 numeric values for signal x.

Fixing Inaccuracies in Pitch Contour Statistics Since pitch information is an important characteristic in answering to our research, we must ensure that pitch measurements correctly reflect the reality. Pitch information in the audio collection $\mathfrak{D}$ only consists of pitched notes of middle c. There are no pitch deviations in the audio collections, like slides, or glides. So the calculation of pitch can be restricted to accurately represent the pitches that are found in audio signals.

For example figure 6.3(a) shows an inaccuracy in measuring the pitch. The audible pitch is clear, but the interuptions and silences in audio cause the pitch measurement to be lower. To solve this problem, we ignore silent parts and achieve a better representation of the pitch such as in figure 6.3(b).

Figure 6.3: Pitch contour and *mean*; (a) with inaccuracies; (b) without.

6.1.3. Calculation of a Feature

The composition of a LLD d and a global functional f on a signal x, gives us a single numeric value that describes some property for signal x. We refer to such a value as a *feature*. A feature is an individual measurable property of a phenomenon. An example of a feature in analyzing audio signals, could be the maximum of the pitch, which can be calculated by the application of the low level descriptor *pitch* and the global functional *maximum*.

Convergence of Features Throughout the training of GAN the *mean* and *std* of measured *pitch* contours (seen in section 6.1.1) converge to that of the data set $\mathfrak{D}$, as seen in figure 6.4.

Figure 6.4: Batch results (*mean*, *std.*) of *pitch* contour statistics (*mean*, *std.*).

Provided that we limited data set $\mathfrak{D}$ to only contain notes with a constant *pitch* (middle c, midi 60), the convergence suggests that generator G_θ is able to learn to create pitched notes similar to that in a given data set.

Figure 6.5: Batch results (*mean*, *std.*) of *spectral centroid* statistics (*mean*, *std.*).

Figure 6.6: Batch results (*mean*, *std.*) of *rms* statistics (*mean*, *std.*).

Figures 6.5 and 6.6 show similar results w.r.t. *spectral centroid* contour *mean* and *std* and *rms* contour *mean* and *std*, respectively. This convergence however, in comparison to the representation of *pitch* contours G_θ in figure 6.4, is less evident.

That pitch contour mean and std have a higher convergence rate than rms (mean and std) and spectral centroid (mean and std) can be explained by the homogeneous pitch contours in the data set and the relatively hetereogeneous rms and spectral centroid. There are loud and soft notes, wide and narrow

ranged-frequency notes, but only notes with pitch midi 60 (equivalent with 261.63 Hz).

However results with the most extreme outliers in rms and spectral centroid at the start of training of G_θ, are discarded to some extent during training.

6.2. Feature Sets

We have covered the convergence of individual features that yield high importance w.r.t. to the generation of musical audio signals, such as the *pitch* of a signal. However, to characterize the signals along other criteria we need more features to describe the signals in its entirety.

Applying a set of global functionals on a set of low level descriptors of a signal, gives us a large array of values that describes the audio signal we refer to as a *feature set* of the signal. Given a collection of LLDs $D = \{d | d \text{ is a lld }\}$ and global functionals $F = \{f | f \text{ is a global functional}\}$. We define a *feature set* $\mathbf{f}$ of a signal $x \in \mathcal{X}$ as a function,

$$\mathbf{f} : \mathcal{X} \to \mathbb{R}^M \tag{6.5}$$

Where $M = |F| \times |D|$. Then for $x \in \mathcal{X}$, $\mathbf{f}(x)$ amount to the application of a set of global functionals F (from 6.3) on a set of low level descriptors D (from 6.2).

Then, feature $\mathbf{f}_{i \mod |F|+|F|\times j}$ of signal $x \in mathcalX$ is defined as

$$\mathbf{f}_{i \mod |F|+|F|\times j}(x) = f_i(d_j(x)) \tag{6.6}$$

Then $\mathbf{f}(x)$ amounts to the row vector,

$$\mathbf{f}(x) = \begin{bmatrix} f_0(d_0(x)) & f_1(d_0(x)) & \dots & f_{|F|-1}(d_{|D|}(x)) & f_{|F|}(d_{|D|}(x)) \end{bmatrix} \tag{6.7}$$

6.2.1. Batch Feature Set

Extending the feature set calculation to work for a batch of signals gives the following defition for batch feature set $\mathbf{F}$,

$$\mathbf{F} : \mathcal{X}^N \mapsto \mathbb{R}^{N \times M} \tag{6.8}$$

where $\mathbb{R}^{N\times M}$ represents a real valued array of N rows and $M = |F| \times |D|$ columns.

Than for a batch of N signals $X \subset \mathcal{X}$ and a feature set $\mathbf{f}$,

$$\mathbf{F}(X) = \begin{bmatrix} \mathbf{f}(X_1) \\ \mathbf{f}(X_2) \\ \vdots \\ \mathbf{f}(X_{N-1}) \\ \mathbf{f}(X_N) \end{bmatrix} \tag{6.9}$$

X_n is the nth signal of batch X, such that $X_n \in \mathcal{X}$.

$\mathbf{F}(X)$ is a matrix which the rows represent the individual signals in the batch X and the columns represent the feature in the feature set f. A feature k of signal n is the value in $\mathbf{F}(X)$ at row n and column k. Given that F is a collection of global functionals and D is a collection of low level descriptors, the values in $\mathbf{F}(X)$ represent the value of unique global functional $f \in F$ applied on unique low level descriptor $d \in D$ of unique signal $x \in X$. The $N \times M$ matrix $\mathbf{F}(X)$ is written in its entirety by combining equation 6.9 and equation 6.7 as follows,

$$\mathbf{F}(X) = \begin{bmatrix} f_0(d_0(X_1)) & f_1(d_0(X_1)) & \dots & f_{|F|-1}(d_{|D|}(X_1)) & f_{|F|}(d_{|D|}(X_1)) \\ f_0(d_0(X_2)) & f_1(d_0(X_2)) & \dots & f_{|F|-1}(d_{|D|}(X_2)) & f_{|F|}(d_{|D|}(X_2)) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ f_0(d_0(X_{N-1})) & f_1(d_0(X_{N-1})) & \dots & f_{|F|-1}(d_{|D|}(X_{N-1})) & f_{|F|}(d_{|D|}(X_{N-1})) \\ f_0(d_0(X_N)) & f_1(d_0(X_N)) & \dots & f_{|F|-1}(d_{|D|}(X_N)) & f_{|F|}(d_{|D|}(X_N)) \end{bmatrix} \tag{6.10}$$

6.3. Mean Batch Feature Distance

Since these feature sets give a large array of numeric values describing a signals characteristics, it is difficult to interpret the convergence of feature sets of generated signals and signals from the data set directly. Therfore we propose a metric, that calculates the column mean of the feature sets for two batches of signals X and Y and calculates the distance between the means.

Let X and Y be two N sized batches of fake and real audio signals respectively, so $X, Y \subset \mathcal{X}$ and $Y \subset \mathcal{D}$. Then given a batch feature set $\mathbf{F}$, as derived in section 6.2.1, and a distance measure d we define the mean batch feature distance **MBFD** between X and Y as,

$$d(\overline{\mathbf{F}(X)}, \overline{\mathbf{F}(Y)}) \tag{6.11}$$

where $\overline{\mathbf{A}}$ represents the column mean for a $N \times M$ matrix $\mathbf{A}$

$$\overline{\mathbf{A}} = \left[\overline{\mathbf{a}_1}\ \overline{\mathbf{a}_2}\ \ldots\ \overline{\mathbf{a}_M}\right] \tag{6.12}$$

With,

$$\overline{\mathbf{a}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{a}_i \tag{6.13}$$

In other words, the column mean of a batch of feature sets gives a mean per feature. This results in that $\overline{\mathbf{F}(X)}$ and $\overline{\mathbf{F}(Y)}$ are two row vectors in $\mathbb{R}^M$, where each element in the row vector represents the mean of the feature value measured in the batch of signals X or Y respectively.

To calculate the distance between $\overline{\mathbf{F}(X)}$ and $\overline{\mathbf{F}(Y)}$, we use euclidean distance. That is for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$

$$d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}|| = \sqrt{|x_1 - y_1|^2 + |x_1 - y_1|^2 + \cdots + |x_1 - y_1|^n} \tag{6.14}$$

6.3.1. Decline of Mean Batch Feature Distance

Now that we have defined the **MBFD**, we calculate it through out the GAN training procedure for a variety of feature sets.

In order to evaluate the metric w.r.t. to the produced image-like signal representations, we introduce figure 6.7 containing a generated log mel spectrograms from the same latent point through out the training procedure. Given a fixed $z \in \mathcal{Z}$, every 100th iteration $G_\theta(z)$ is calculated and the log mel spectrogram is shown. This gives an indication of at what iteration certain structures described in chapter 5 began to occur in the generated log mel spectrograms. The most promiment information from figure 6.7 is that the most impressive development occur in the first 1000 iterations.

Composition of Feature Sets in Use The variety of feature sets gives perspective on how different combinations of signal characteristics (*features*) constitute similarity between generated signals and signals from the collection. The feature sets we use to calculate the **MBFD** are given in table 6.1.

Table 6.2 shows the used LLDs in set A and set B. Every LLD in table 6.2 represents a time dependent contour similar to the pitch contour in figure 6.1, but for different properties of the audio signals. The description of the property that the LLD gives is also given in table 6.2. The exact definitions and calculation methods of the LLDs are omitted in this study, but can be found in [18]. LLD set A in table 6.2 resembles a minimal representation of audio characteristics, while LLD set B is more exhaustive.

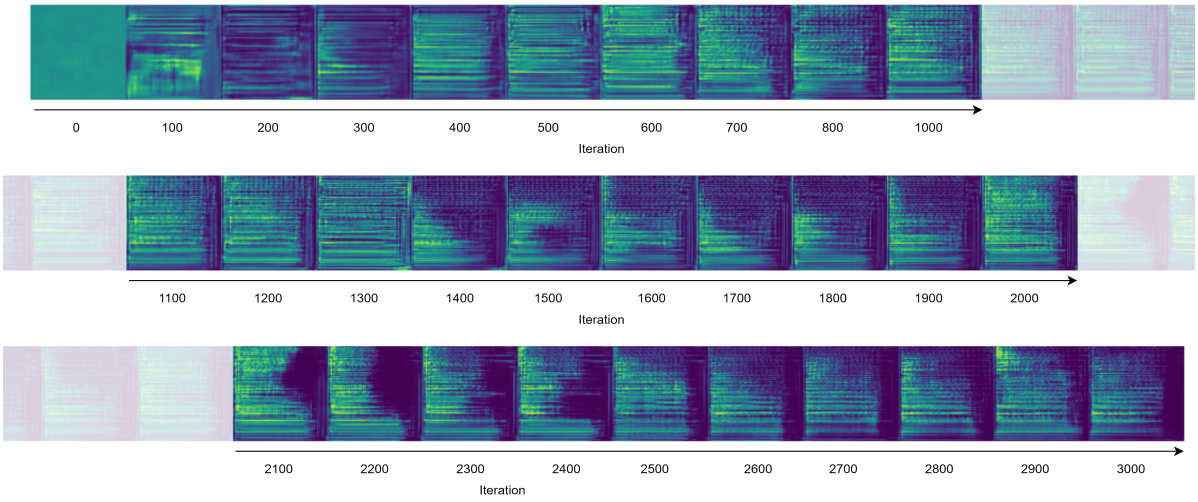

Figure 6.7: Generated log mel spectrograms through training. Given a fixed $z \in \mathcal{Z}$, every 100th iteration $G_\theta(z)$ is calculated and the log mel spectrogram is shown. The most impressive development occurs in the first 1000 iterations (row 1).

Feature Set	LLD Set (Table 6.2)	Global Functional Set (Table 6.3)	Size (*)
$\mathbf{f}^{A,A}$	A	A	69
$\mathbf{f}^{A,B}$	A	B	12
$\mathbf{f}^{A,C}$	A	C	6
$\mathbf{f}^{B,A}$	B	A	515

Table 6.1: Table showing feature sets. (*= sizes are approximately equal to correspoding $|D| \times |F|$. Any deviations can be caused by inability to calculate LLD or functionals for LLD)

Audio Quality	Low Level Descriptor	Description	Set A	Set B
Pitch	Pitch Contour	Curve that tracks the perceived pitch of the sound over time	□	•
Dynamics	Root Mean Squared	Root mean squared of signal amplitude during frame	•	•
Timbre	MFCC (1-14)	Mel Frequency Cepstrum Coefficients		•
	Spectral Centroid	Center of mass of the spectrum	•	•
	Spectral Measures	(Slope, Flux, Entropy, Spread, Skewness, Kurtosis, Rolloff, Flatness)		•
	Harmonics to Noise Ratio	The ratio between periodic and non-periodic components.		•
	Formants	Local maxima in the spectrum.		•
Other	Zero Crossing Rate	The rate at which a signal crosses the zero.		•

Table 6.2: Table showing sets of low level descriptors. All frames used during the LLD calculations are obtained with `n_fft_seconds=0.04`, `hop_length_seconds=0.01` where necessary. The methods are implemented with the *surfboard* audio analysis toolkit. [18].

Table 6.3 shows global functionals that are applied on the LLDs from 6.2. Three sets A, B, C of global functionals are used. These sets vary in exhaustiveness, similarily to the LLD sets in 6.2.

Global Functional	Set A	Set B	Set C
Min, Max	•	•	
Mean, Std.	•	•	•
Skewness, Kurtosis	•		
First Derivative (Mean, Std, Skewness, Kurtosis)	•		
Second Derivative (Mean, Std, Skewness, Kurtosis)	•		
1st, 2nd, 3rd Quartile	•		
Interquartile Ranges (2-1, 3-2, 3-1)	•		
Percentile Range (1%)	•		
Percentile Range (99%)	•		
Interpercentile Range (1-99%)	•		

Table 6.3: Table showing sets of global functionals. Calculated with *surfboard* [18]

Regular Mean Batch Feature Distance Figure 6.8 shows mean batch feature distances **MBFD**s from four feature sets as seen in table 6.1. The feature sets are calculated in batches of signals, denoted with batch feature sets as in section 6.2.1: $\mathbf{F}^{A,A}, \mathbf{F}^{A,B}, \mathbf{F}^{A,C}, \mathbf{F}^{B,A}$. The batch feature sets correspond to the feature sets $\mathbf{f}^{A,A}, \mathbf{f}^{A,B}, \mathbf{f}^{A,C}, \mathbf{F}^{B,A}$ from table 6.1 respectively.

Given a generator G_θ at a training iteration. Let Y be a batch of signals from the sound collection $\mathfrak{D}$. Let X be a batch of signals generated by generator G_θ. Then in figure 6.8 we see the **MBFD** as defined in section 6.3,

$$d(\overline{\mathbf{F}(X)}, \overline{\mathbf{F}(Y)}) \tag{6.15}$$

The horizontal axis in figure 6.8 reperesents the training iteration. The vertical axis represents $d(\overline{\mathbf{F}(X)}, \overline{\mathbf{F}(Y)})$ for a batch X sampled from $\mathfrak{D}$ and a batch Y generated by generator G_θ at training iteration t.

The trajectory of the **MBFD** during training in figure 6.8 suggest some decline in the distance between generated signal characteristics and signal characteristics from the sound collection $\mathfrak{D}$. While the absolute difference between the **MBFD** given different feature sets is large, they all appear to decline through out the training procedure at some point. The **MBFD** with batch feature set $\mathbf{F}^{B,A}$ has a drastically higher value and higher relative decline compared to the other **MBFD**s. This can be explained by the high amount of features that are present in batch feature set $\mathbf{F}^{B,A}$, as seen in table 6.1 (515, compared to $69, 12, 6$).

The most dramatic decline in figure 6.8 occurs in the first 1000 iterations. This corresponds to the inspected visual developments in figure 6.7.

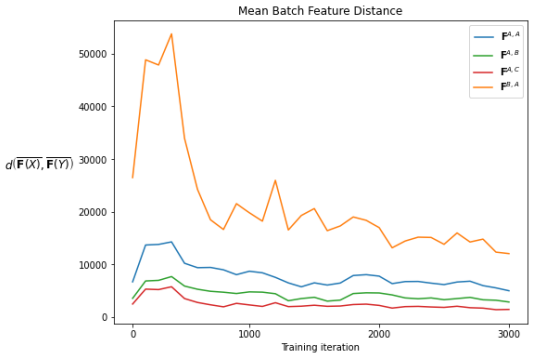

Figure 6.8: Mean batch feature distance during training with different feature sets. The absolute difference between the different **MBFD**s is large. Every **MBFD** appears to decline through out the training procedure at some point.

Scaled Mean Batch Feature Distance While figure 6.8 shows an impressive decline for the **MBFD** with batch feature set $\mathbf{F}^{B,A}$, the decline of the other **MBFD**s are slightly obscured by its scale. This incites us to introduce a scaled version of the **MBFD** to obtain more insight in the relative declines of each of the **MBFD**s.

In order to achieve the scaled version of the **MBFD** the trajectory from figure 6.8 is devided by its maximum value,

$$\frac{d(\overline{\mathbf{F}(X)}, \overline{\mathbf{F}(Y)})}{\max d(\overline{\mathbf{F}(X)}, \overline{\mathbf{F}(Y)})} \tag{6.16}$$

Figure 6.9 shows the scaled **MBFD** from figure 6.8. In this figure a decline in each of the **MBFD**s is clearly visible. Similar to regular **MBFD** the decline of the scaled **MBFD** is most promimently present in the first 1000 iterations. Again, this corresponds to the visual inspection of generated log mel spectrograms through the generator training, as seen in figure 6.7.

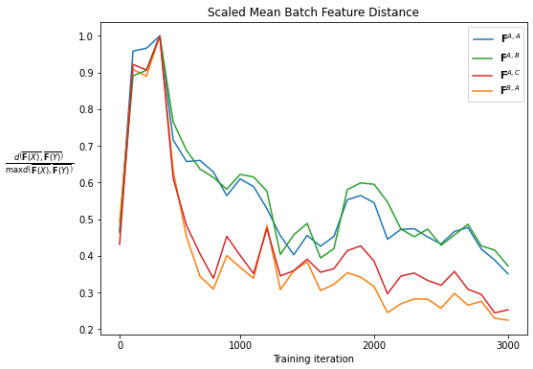

Figure 6.9: Scaled mean batch feature distance during training with different feature sets. A decline in each of the **MBFD**s is clearly visible.

Mean Batch Normalized Feature Distance A different limitation of the regular **MBFD** in figure 6.8 is are the unnormalized features heavily influencing the result. Features with high variance can cause the distances between column means $\overline{\mathbf{F}(X)}$ and $\overline{\mathbf{F}(Y)}$ to be distorted. To mitigate this, we introduce the **MBFD** with *normalized* features,

$$d(\overline{\hat{\mathbf{F}}(X)}, \overline{\hat{\mathbf{F}}(Y)}), \tag{6.17}$$

Here the unit normalziation of $\mathbf{u}$ is depicted with $\hat{\mathbf{u}}$, such that $\hat{\mathbf{u}} = \frac{u}{||u||}$ (*unit normalization*).

At last, as seen in figure 6.10, the mean batch normalized feature distance has a slight decline. While the decline is less clear to the declines in figures 6.8 and 6.9, there is definitely a drop from the **MBFD**s for each batch feature set at iteration 0 to iteration 3000. Again, the most prominent contribution to the drop in value is present in the first 1000 iterations, similar as in the figures 6.8, 6.9 and figure 6.7.

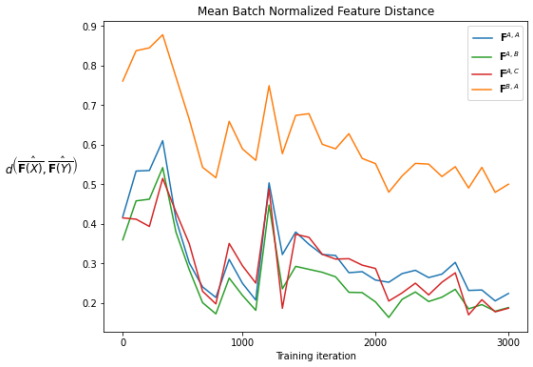

Figure 6.10: Mean batch normalized feature distance during training with different feature sets. A slight decline is seen in for all feature sets. **MBFD** with feature set $\mathbf{F}^{B,A}$ appears to remain in high value.

It remains questionable which of the variants of a **MBFD** yields the most importance w.r.t. evaluating the perfomance of our generator. However, each of the variants had some amount of decline. This is

coherent with the visual inspection of the generator at iteration 3000 in chapter 5 and the progression of generated log mel spectrograms in figure 6.7. We can value each of these metrics w.r.t. some amount of measuring similarity. The declines of the **MBFD** variants in figures 6.8, 6.9, 6.10 suggest that at least some generated audio signal characteristics are approaching the audio collection characteristics.

Summary

In this chapter we covered topics related to extracting numerical audio characteristics through feature engineering:

- Low level descriptor provide characteristic contours for a signal, such as the pitch contour of a signal. (Section 6.1.1)
- Global functionals are used to extract global information of a contour, such as the mean or standard deviation of a contour. (Section 6.1.2)
- The composition of a collection of low level descriptors and global functionals gives us a feature set that can be calculated for a signal. The feature set gives a objective global indication of singal properties. (Section 6.2)

Furthermore, we covered the convergence of features calculated on the generated audio throughout the training of generator to features calculated on the audio in the data set.

- The pitch contour mean and standard deviation of generators converge clearly, as seen in figure 6.4. The mean spectral centroid and rms however converge only slightly, as seen in figures 6.6 and 6.5. (Section 6.1.3)

At last we constructed a metric based on mean of batches of feature sets, called mean batch feature distance **MBFD**.

- While being a rough derivative of initial signal characteristics, the metric declines steadily throughout the training of a generator for a variety of feature sets and variations on the **MBFD** (including unit normalized and scaled variations) (Section 6.3)
- Also the decline in **MBFD** shows coherence with the inspected progression in generated log mel spectrograms as seen in figure 6.7 (Section 6.3)

7

Conclusion

7.1. Summary

In this study we worked towards a proof of principle (PoP) for GAN driven audio generation with the use of an image-like representation for audio, and convolutional neural networks. The methods were implemented for modelling [1] a collection of audio consisting of electric pianos playing 4 second notes of middle c. With this, we obtain a generative model that is able to generate sound similar to the sounds it is trained on.

While the goal in this study is to generate audio, the modelling component takes place in the image domain. Accordingly, in chapter 2 we convert audio signals into an image-like representations of time and frequency. The signal-to-image transformation Φ, as seen in figure 2.3, converts a 4 second long and 16 kHz sampled signal to 2 stacked 256×256 images. The representation displays frequency magnitudes and phase information at time segments in the signal. The frequency axis is scaled to *mel* scale, a better representation of human hearing. On the other hand, the image-to-signal transformation Φ^{-1}, as seen in 2.4, amounts to the inverse: 2 stacked 256×256 images, interpreted as time-frequency information, are converted to a 4 second audio signal, again sampled at 16 kHz.

In order to obtain a generative model for a collection of audio, we need a parametric function G_θ that is able to generate audio with appropriate processing and whose parameters θ can be adapted to model the collection of audio. G_θ needs to produce audio similar to that of the collection it is based on. A generative adversarial network, or GAN, is a method used to obtain such a model. The GAN network resembles a competition between a generator and a discriminator. The generator is producing fake material, while the discriminator is distinguishing the fake material from the real material. This way we obtain a generative model for a collection of audio [9]. In chapter 3 we discuss the design of GAN in context of generating audio. We show how audio generation is the result from the inference of G_θ on a point in latent space $\mathcal{Z}$, as in figure 3.1. Furthermore, we show how the generator's parameters θ are optimized through an adversarial training scheme, as seen in figure 3.3.

Now that we described a method to obtain a generative model from parametric function G_θ for a collection of audio, we need to define G_θ. We need to design G_θ such that its able to produce the image-like representation of the audio signals and be trained by the adversarial network. For this, we design a convolutional neural network. In chapter 4, we discuss the fundamental building block of neural networks, the neuron. After which we will discuss how a network of layers of neurons can be trained to be used as a function approximator [19]. Furthermore, the use of convolution neural networks will be demonstrated to attain to the generation of image-like signal representations defined in chapter 2. At last, we give an architectural overview of the discriminator network D_ϕ, that has an input of shape $256 \times 256 \times 2$, and generator network G_θ, that has an output of shape $256 \times 256 \times 2$, that are used in GAN to model the collection of audio.

[1] as defined in the introduction.

In chapter 5, we inspect the generators output after following a training procedure. The trained generator network G_θ generates similar spectral information to the spectral information of signals from the dataset $\mathcal{D}$, as seen in figure 5.2. The structural similarities in the spectrum, such as the distribution of harmonics and note envelopes, suggest an effective approach to modelling electric piano notes. Also there appears to be a similar amount of diversity in the generated material as in the data set material, seen in figure 5.3. However, visual (and audible) artifacts are present in the generated spectra, such as noisy structures in high frequencies, blurry areas and wobbly harmonic magnitudes, as seen in figure 5.4. At last, we observe the effects of latent space interpolation on audio in figure 5.8. The effects of latent space interpolation between points in latent space $\mathcal{Z}$ and inferencing G_θ, yields the morphing between spectral information analoguous to the morphing of faces in [29].

To quantitatively assess the effect of the generator training process, we measure and compare audio qualities of signals produced by the generator and that of the data set in chapter 6. Signals produced from trained generator network G_θ and signals from the dataset $\mathcal{D}$ show similarities w.r.t. a set of audio low level descriptors, such as *pitch*. These comparisons show a convergence throughout training of the generator to the characteristics of the training data set. Following this, we quantify the performance of G_θ throughout adversarial training in chapter 3. We define a quantitative performance, named *mean batch feature distance* (**MBFD**), for evaluating the difference between audio characteristics of two sets of audio signals. The metric is based on audio characteristics and declines throughout training, as seen in figures 6.8, 6.9, and 6.10.

7.2. Interpretation of Results

The results suggest that the generative adversarial network is able to produce audio signals that yield some limited similarity to the signals of the audio collection it was trained on. The similarities are not only perceived visually by inspecting the image-like representations. They can also be quantified by measuring low level audio signal characterstics of the generator produced signals and signals from the audio collection. From these audio characteristics a declining performance metric can be derived throughout the training iterations of the generator. Viewing the dramatic improvement in visual resemblance and performance metric before and after training, suggests a functioning training architecture and training process for modelling the audio collection.

The most promising result from measuring the audio characteristics is the prominent convergence of the pitches of the generated audio throughout network training. The pitches in the audio collection were all middle c, so the characteristic had a clear aimed value, which the generator was able to achieve. Furthermore, the capability of producing a pitched note is a fundamental element in traditional synthesis, or musical instruments in general.

At last, the generative adversarial network introduces a method to non-linearily interpolate between points in the codomain as a result of latent space interpolation and inferencing the trained generator network on the latent points. This establishes a conceptual difference in morphing between two sounds. Every point in the latents space interpolation produces a modelled sample from the audio collection as interpreted by the generator.

7.2.1. Implications

Given these abilities, using a generative adversarial network trained on a specified audio collection to model it, gives us a controllable synthesizer that is able to produce audio with characteristics from that collection, where the content is from arbitrary source. The capabilities of the synthesizer are not limited by the available synthesis components, such as oscillators and filters, and the ability to tune these components by hand, but rather the pattern recognition capabilities of neural networks and the generative modelling capabilities of GAN.

This introduces a branch of audio synthesis methods where potentially the training of a generator based on a user specified collection yields the form of expression, rather than tuning parameters by hand to obtain a sound as desired. For example, a synthesizer that is able to produce bird sounds can be

obtained by training the generator network with GAN on a collection of bird sounds. Whereas the hand-tuning of synthesizer controls to obtain bird sounds can be a difficult task. Also, the training of a generator on a diverse sound collection, such as that of drum kit sounds, allows the latent space interpolation between different extremes within the collection, such as a kick and a hi hat, such that the resulting interpolated audio is not a sum of the two, but rather an other learned sound that yields similarites to the sound collection.

7.2.2. Limitations

Allthough serving as a proof of principle (PoP), the validity for generalization of this study has some limitations. These limitations are mainly the result of limited resources, such as computational power and time. Following this, we have little insight in how the method performs with more varied data. Also, as we have only investigated a limited amount of neural network architectures, audio representations, and processing methods, we can hardly differentiate the bottlenecks within the system that influence the artifacts that are present in the generated audio. In fact, it is probable that each component in the system contributes to the presence of the artifacts in some way, shape, or form. At last, the evaluation of our method has been simplified to serve the PoP. However, using different evaluation methods, and evaluating on different domains, is necessary to establish valuable decisions in improving the PoP.

Limited Diversity The results in this study have been obtained by training on a restricted sound collection, namely the electric piano notes playing middle c. Results on training the generator with more diverse or complex sound collections could give a better indication on the generalized performance of the method. For example, we have trained the generator network to produce a single pitch, middle c. This serves the PoP, but extending the pitch and instrument producing capabilities by training on a sound collection with multiple pitches and multiple instruments, is necessary for further development of the PoP. The high quality human face generation [15] does suggest that GAN is able to learn with a diverse data set. With appropriate adaptations to our method, the ability to increase the generation diversity looks promising.

Inadequate Insight in Source of Artifacts Another limitation of this study, is that the artifacts that are present in the generated log mel spectrograms and audio are difficult to appoint to a source. In this study we have used audio processing methods to produce image-like audio representations and convolutional neural networks to generate and discriminate these representations. In the design of each of the singal representation, processing and networks causes for the artifacts can be introduced. The log mel spectrograms give a versatile way of representing audio, but it neglects a lot of structures that are naturally present in some sounds, such as the differentiation between harmonic information and noise present in many musical instruments. Log mel spectrograms deliver value when arbitrary spectra of audio need to be learned to produce, but for generating electric pianos representing the audio as a sum of harmonics and filtered noise, for example, could eliminate a lot of artifacts while still maintaining more than adequate representation capabilities.

Furthermore, while the audio to image transformation on its own is a nearly reversible process, conserving the phase information with the spectral information together might be of negative influence in the capabilities of a neural network to model the audio effectively. At the same time the conserving of phase information has not been investigated in the positive effects of the audio generation. Different approaches to recovering the phase information could yield more or less artifacts.

At last, different neural network architectures yield structural differences in the output, potentially removing artifacts. In this study, little experimentation has been done with different network archictures.

These three examples show how many factors influence the generation and modelling of the audio. Extending the research to other network architectures and processing can give more insight into what causes certain artifacts. Consequently, this gives more insight in the modelling capabilities of GAN rather than the processing or network architectures.

Limited Evaluation While attempts have been made to evaluate the PoP, extending the variety of evaluation methods and evaluation criteria yields is of high value. In this study, we have only focused on measuring the similarity between the data set and the genarator signals. This serves our purpose, as we restricted the data set heavily. However, investigating the diversity of the produced signals in comparison to the diversity of the data set could give more insight when using different and more varied collections of audio. On one hand we wish the model to generate audio that yields similarities to the sound collection. On the other hand we want to generate audio with similar diversity to the audio collection.

As we have focussed on using low level signal descriptors, the resulting quantifactions are heavily influenced by the representation capabilities of these signal descriptors. The use of different methods to represent the signals in measuring similarity and diversity is recommended.

At last, similarity and diversity are not the only criteria that can evaluate the PoP. Especially in context of music, qualitative metrics, such as human evaluation, or metrics for creative use can have the upper hand in decisions for future development of the GAN driven audio synthesis.

7.3. Research Suggestions

During the research and development of teh PoP a variety of actions and questions emerged to combat the limitations of the study. The research recommendations can be divided into three categories: research w.r.t. obtaining insight in the performance of the PoP, research w.r.t. quality improvements of the audio processing, research w.r.t. investigating different neural network architectures.

The following actions relate to experimental improvements that aim to better investigate performance of the PoP directly:

- **Diverse audio data sets**: The methods in this study have been solely tested with electric pianos playing a middle c (from NSynth [5]). Extending the tests of the framework to different instruments and sounds can give a better perspective on how the framework performs. NSynth offers a variety of instrument sounds playing different notes. Different instrument classes, such as strings, and different pitches, can be easily chosen to extend the insight in results. However, especially percussive oriented sounds [34], environmental sounds [28], bird sounds [25], or human speech [26], can give insight in how the PoP deals with more complex and varied sound data sets.
- **Amplitude Weighted MBFD**: The performance metric **MBFD** introduced in this study uses information of low level signal descriptors to analyze generated audio and audio from the audio collection. However, these low level signal descriptors give descriptor contours, such as the pitch contour, throughout the entire signal. This can give false information in the case audio is interpretably silent or soft, potentially obscuring valuable information or cluttering with invaluable information. Using the amplitude of signals to weigh the contribution of a descriptor contour to the performance metric, we can obtain a better representation of what characteristics the audible parts of the signal have.
- **GAN Performance Metrics**: Other performance metrics introduced with GAN can be used such as Frechet Inception Distance (FID) and Inception Score (IS) [3]. FID and IS are metrics for evaluating GANs on the perceptual quality and diversity on synthetic distributions. The metrics penalize models whose generated examples are not each classified into a single class, as well as models whose generated examples collectively belong to only a few of the possible classes. However both FID and IS rely on a pretrained image classifier, this is of little value when using it for audio generation. By replacing the pretrained image classifier with an log mel spectrogram classifier convolutional neural network, for example, the FID and IS metrics could be used. [6] implemented these two metrics by using a pitch classifier.

The following actions and questions relate to quality improvement of audio processing:

- **Filtering log mel spectrograms**: A simple way to potentially improve the quality of the audio from the generated log mel spectrograms, is by manually filtering and cleaning unwanted artefacts. Experimentation with AI adapted filters [40] and noise removal [21] could yield interesting results.

- **Phase reconstruction methods**: In this study we have trained the generator to also generate phase information, in order to retrieve audio signals from the generated output. Other methods to generate audio from log mel spectrograms with phase reconstruction, such as other time-frequency modelling methods [22] and deep Griffin-Lim [23], could potentially increase the quality of the audio.

The following actions and questions relate to improvement of architecture of GAN:

- **Conditioning GAN**: The conditioning of audio qualities, such as instrument type or pitch, in GAN [15, 24], allows the generation of audio based on a set of conditions. This gives users ability to control the characteristics of generated audio better. Also [27] promises to introduce qualitative improvements with the use of conditional GANs.
- **Differentiable Synthesis**: Replacing upsampling convolutional network in the generator with flexible differentiable synthesizers as proposed in [7], proposes a different approach to generating audio than generating log mel spectrograms. If the synthesizers are flexible enough, the generator's synthesis components are able to reproduce arbitrary sounds, while potentially producing high quality audio a lot earlier in the training process. Differentiable digital signal processing components, such as oscillators and filters, are provided by [7]
- **GAN Architectures**: Explore different GAN architectures, such as progressive GAN [14], which allows the generation of higher quality images of larger resolutions. This visual improvement directly correlates to the improvement of audio quality, as the log mel spectrograms can have higher frequency resolution and higher time resolution. The progressive GAN architecture is used in GANSynth [6].
- **Different length audio**: At last, in this study we have focussed on audio of fixed length (4 seconds, or 64000 samples), investigating the ability to generate longer, or shorter audio signals could yield different results. Also the ability to work with audio of arbitrary length, such as with [13, 15], allows the easier combining of different datasets to achieve more general performative results.

7.4. Predictions

The system introduced here in its current state is rough around the edges and has little artistic or commercial use. The effort, time, and hardware that are currently requiered to train and use the method introduced in this study are out of proportion with the intended effect of using a generative adversarial network to model an audio collection for musical deployment.

When comparing the state of our system to the first synthesizer build by the RCA in 1952 [2] similarities can be seen in their rigid, unpolished and immovable properties. Where the RCA synthesizer required a room filled with hardware, our system requires days of processing and high electricity bills for a meager variety of sounds rich in unwanted artifacts. Luckily, the development of artificial intelligence, deep learning in particular, has a high and steadfast velocity. The future of the GAN driven audio synthesis benefits from the efforts of these developments. Optimizations and improvements within neural networks and GAN in particular directly influence in the audio modelling capabalities. With time, optimizations might decrease training times and hardware requirements. These optimizations are vital for the ability of user-tailored training of a GAN driven synthesizers on customly defined data sets. User-tailored training unlocks the complete tool the GAN driven audio synthesis aims to offer.

Advancement in GAN driven audio synthesis with respect to directly improving the creative use can quickly incite the embracement of the method among experimental musical and artistic users. Predictably, this will be no different to how the avant garde embraced early synthesizers in the 1960s.

[2] As stated in the introduction : in 1952 Harry Olson and Herbert Belar of the RCA (Radio Corporation of America) created the first synthesizer capable of artificially creating sound.

Bibliography

[1] Jont Allen. Short term spectral analysis, synthesis, and modification by discrete fourier transform. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 25:235 – 238, 07 1977. doi: 10.1109/TASSP.1977.1162950.

[2] J. Chadabe. *Electric Sound: The Past and Promise of Electronic Music*. Prentice Hall, 1997. ISBN 9780133032314. URL https://books.google.nl/books?id=J4nuAAAAMAAJ.

[3] Min Jin Chong and David Forsyth. Effectively unbiased fid and inception score and where to find them, 2020.

[4] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, 2018.

[5] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoder, 2017.

[6] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. GANSynth: Adversarial Neural Audio Synthesis, 2019.

[7] Jesse Engel, Lamtharn (Hanoi) Hantrakul, Chenjie Gu, and Adam Roberts. Ddsp: Differentiable digital signal processing, 2020. URL https://openreview.net/forum?id=B1x1ma4tDr.

[8] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016. ISBN 9780262035613. URL https://books.google.co.in/books?id=Np9SDQAAQBAJ.

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks, 2014.

[10] F.J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978. doi: 10.1109/PROC.1978.10837.

[11] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208.

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[13] Animesh Karnewar and Oliver Wang. Msg-gan: Multi-scale gradients for generative adversarial networks, 2020.

[14] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.

[15] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

[16] Nasser Kehtarnavaz. Chapter 7 - frequency domain processing. In Nasser Kehtarnavaz, editor, *Digital Signal Processing System Design (Second Edition)*, pages 175–196. Academic Press, Burlington, second edition edition, 2008. ISBN 978-0-12-374490-6. doi: https://doi.org/10.1016/B978-0-12-374490-6.00007-6. URL https://www.sciencedirect.com/science/article/pii/B9780123744906000076.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[18] Raphael Lenain, Jack Weston, Abhishek Shivkumar, and Emil Fristed. Surfboard: Audio feature extraction for modern machine learning, 2020.

[19] Shiyu Liang and R. Srikant. Why deep neural networks for function approximation?, 2017.

[20] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991. doi: 10.1109/18.61115.

[21] Angshul Majumdar. Blind denoising autoencoder, 2019.

[22] Andrés Marafioti, Nathanaël Perraudin, Nicki Holighaus, and Piotr Majdak. Adversarial generation of time-frequency features with application in audio synthesis. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proc. of the 36th ICML*, volume 97, pages 4352–4362, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/marafioti19a.html.

[23] Yoshiki Masuyama, Kohei Yatabe, Yuma Koizumi, Yasuhiro Oikawa, and Noboru Harada. Deep griffin-lim iteration, 2019.

[24] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets, 2014.

[25] Veronica Morfi, Yves Bas, Hanna Pamuła, Hervé Glotin, and Dan Stowell. Nips4bplus: a richly annotated birdsong audio dataset, 2018.

[26] A. Nagrani, J. S. Chung, and A. Zisserman. Voxceleb: a large-scale speaker identification dataset. In *INTERSPEECH*, 2017.

[27] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans, 2017.

[28] Karol J. Piczak. ESC: Dataset for Environmental Sound Classification. In *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pages 1015–1018. ACM Press, 2015. ISBN 978-1-4503-3459-4. doi: 10.1145/2733373.2806390. URL http://dl.acm.org/citation.cfm?doid=2733373.2806390.

[29] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2016.

[30] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986. doi: 10.1038/323533a0. URL https://doi.org/10.1038/323533a0.

[31] Björn Schuller. Affective speaker state analysis in the presence of reverberation. *International Journal of Speech Technology*, 14:77–87, 06 2011. doi: 10.1007/s10772-011-9090-8.

[32] Douglas Shaughnessy. *Speech communication : human and machine*. Addison-Wesley Pub. Co, Reading, Mass, 1987. ISBN 978-0-201-16520-3.

[33] Chen Shen, Zhongming Jin, Yiru Zhao, Zhihang Fu, Rongxin Jiang, Yaowu Chen, and Xian-Sheng Hua. Deep siamese network with multi-level similarity perception for person re-identification. pages 1942–1950, 10 2017. doi: 10.1145/3123266.3123452.

[34] Carl Southall, Chih-Wei Wu, Alexander Lerch, and Jason Hockman. Mdb drums – an annotated subset of medleydb for automatic drum transcription. 10 2017.

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[36] S. S. Stevens, J. Volkmann, and E. B. Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, January 1937. doi: 10.1121/1.1915893. URL https://doi.org/10.1121/1.1915893.

[37] Wikipedia Commons. Fourier transform time and frequency domains, 2013. URL https://en.wikipedia.org/wiki/File:Fourier_transform_time_and_frequency_domains_(small).gif.

[38] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network, 2015.

[39] Xue Ying. An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, 1168:022022, 02 2019. doi: 10.1088/1742-6596/1168/2/022022.

[40] Ryandhimas E. Zezario, Jen-Wei Huang, Xugang Lu, Yu Tsao, Hsin-Te Hwang, and Hsin-Min Wang. Deep denoising autoencoder based post filtering for speech enhancement. In *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 373–377, 2018. doi: 10.23919/APSIPA.2018.8659598.

[41] Jiawei Zhang. Gradient descent based optimization algorithms for deep learning models training, 2019.

Appendices

A

Enlarged Log Mel Spectrogram Images

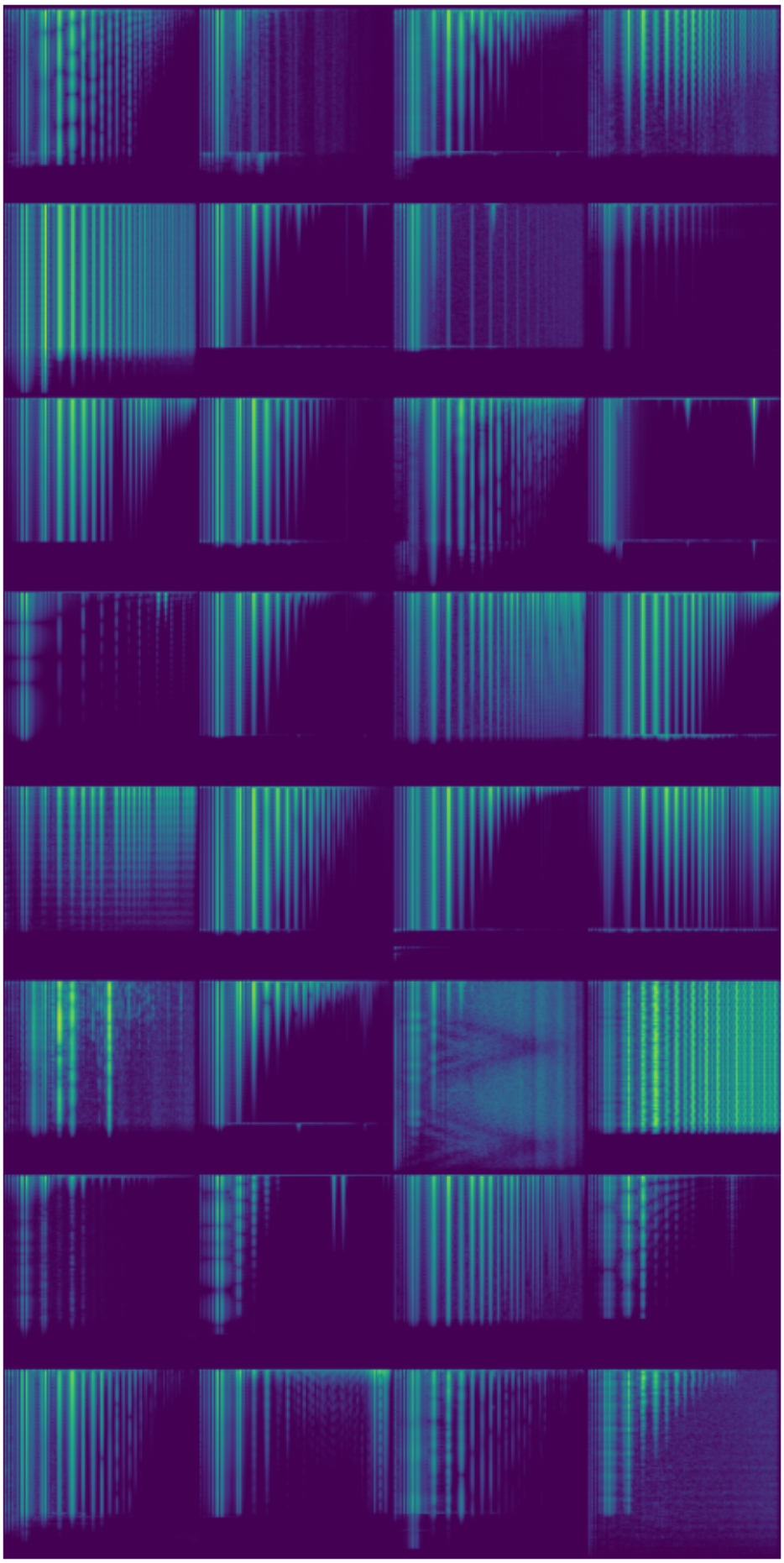

Figure A.1: Log mel spectrograms of 32 electronic piano samples from the training data set. The vertical axis wihtin each log mel spectrogram is the frequency axis and the horizontal the time axis.

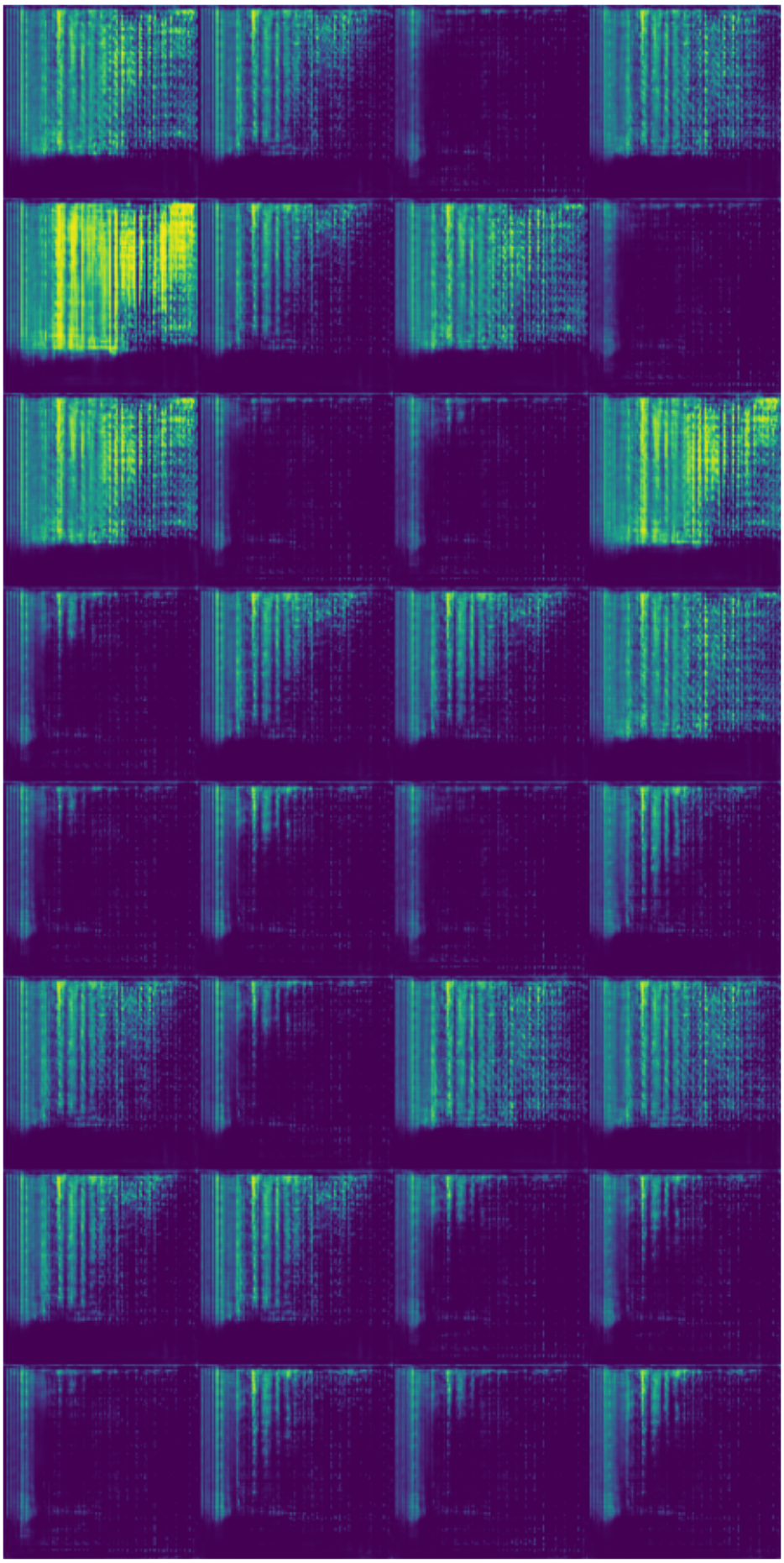

Figure A.2: 32 generated images from inferencing generator G_θ on 32 latent points **after** training of GAN. The vertical axis wihtin each log mel spectrogram is the frequency axis and the horizontal the time axis.