POLITECNICO DI MILANO
Master of Science in Computer Science and Engineering Dipartimento di Elettronica, Informazione e Bioingegneria

Re-synthesis of instrumental sounds with Machine Learning and a Frequency Modulation synthesizer

Tesi di Laurea di:
Philip Claesson
philipcl@kth.se
898664

Supervisor: Fabio Antonacci
Co-supervisor: Henrik Boström

Milano, 3rd of Ocotober 2019
Academic Year 2018-2019

Abstract

Frequency Modulation (FM) based re-synthesis - to find the parameter values which best make a FM-synthesizer produce an output sound as similar as possible to a given target sound - is a challenging problem. The search space of a commercial synthesizer is often non-linear and high dimensional. Moreover, some crucial decisions need to be done such as choosing the number of modulating oscillators or the algorithm by which they modulate each other. In this work we propose to use Machine Learning (ML) to learn a mapping from target sound to the parameter space of an FM-synthesizer. In order to investigate the capabilities of ML to implicitly learn to make the mentioned key desicions in FM, we design and compare two approaches: first a *concurrent* approach where all parameter values are compared at once by one model, and second a *sequential* approach where the prediction is done by a mix of classifiers and regressors. We evaluate the performance of the approaches with respect to ability to reproduce instrumental sound samples from a dataset of 2255 samples from over 700 instrument in three different pitches with respect to four different distance metrics, . The results indicate that both approaches have similar performance at predicting parameters which reconstruct the frequency magnitude spectrum and envelope of a target sound. However the results also point at the sequential model being better at predicting the parameters which reconstruct the temporal evolution of the frequency magnitude spectrums. It is concluded that despite the sequential model outperforming the concurrent, it is likely possible for a model to make key decisions implicitly, without explicitly designed subproblems.

Keywords: machine learning; regression; classification; frequency modulation synthesis; re-synthesis;

Acknowledgements

I would like to thank my supervisor at Politecnico di Milano, Fabio Antonacci, for supervising my thesis. I would also like to thank Bob Sturm at KTH for the helpful guidance regarding signal processing, machine learning and academic writing. I would also like to thank professor Henrik Boström for his extensive feedback, general guidance and for being my supervisor at KTH. I would like to thank David Möllerstedt and Jonas Åberg at Teenage Engineering for suggesting this interesting topic as well as developing some of the software used in this project. A further thank you to everyone else at Teenage Engineering who has been interested in discussing this project, including of course my intern partner in crime, Ben Olayinka. Also, thanks to Moritz Meister for all the good discussions and times in Piazza Leonardo.

Finally, a big thanks to my mother Annika Ahlgren, father Håkan Claesson and brother Lucas Claesson for always supporting me.

Stockholm, October 2019
Philip Claesson

Contents

Chapter 1

Introduction

This thesis explores the task of Machine Learning-guided sound resynthesis using a Frequency Modulation-synthesizer with a large parameter search space. The project is carried out in a partnership with Teenage Engineering in Stockholm. This section introduces the background, problem and methodology of this thesis.

Teenage Engineering (TE) is a company producing synthesizers, speakers and related hard- and software for sound design and music production. TE have previously been in research partnership with scientists at the Metacreation Lab at the School of Interactive Arts and Technology of Simon Fraser University's Faculty of Communication regarding sound re-synthesis through Artificial Intelligence.[22]. For the sake of this thesis, TE has contributed through developing a synthesizer software to be used in the experiments. The synthesizer, referred to as the TE Synthesizer, is explained in detail in section 2.2.

1.1 Background

With the increased performance of general purpose computer hardware, comes an increase in the complexity of software synthesizers. For example, the Native Instruments' FM8 is configured through over 1000 parameters [28], and Teenage Engineering's OP-1 synthesizer can be set to 10^{76} distinct combinations of parameter values [22]. The many parameters often have a non-linear impact on the output sound of the synthesizer, yielding a vast and high dimensional search space.

The vast parameter space is an obstacle to a user attempting to design specific sounds through the synthesiser's parameters. The user's interaction

with the synthesiser's parameter space may roughly be divided into:

- *Search:* the process of searching for the set of parameter values which make the synthesizer produce a desired output sound, e.g. *"I want the synthesizer to sound like a grand piano."*
- *Exploration:* the process of exploring the parameter space to find sets of parameter values which make the synthesizer produce sounds which are not preciously known, e.g. *"I want to hear what the synthesizer can sound like."*

Due to the size and non-linearities of the parameter space the search process can be demanding even for expert sound designers - this process is also referred to as *re-synthesis*. For the same reasons, to determine how much of the variety of the possible output sounds has been explored may be hard or impossible even for the creators of the synthesizers. To facilitate users in search and exploration of the parameter space, it is common to introduce a number of pre-configured combinations of parameter values (presets). A preset may act both as a shortcut in the search for popular output sounds, as well a starting point for exploration.

However, a finite set of presets is not certain to help the user find any desired sounds which can be produced by the synthesiser, and might not introduce the user to all possibilities of the parameter space. Also, producing large set of presets is a non-trivial and time consuming task for the creators of the synthesiser, and even if the preset designer was involved in programming the synthesizer it is hard to discover all the possibilities of the high dimensional parameter space. An automated tool could also be useful for migrating presets between products and between operative system specific software implementations of the synthesizer.

For these reasons, a function which takes a target sound as input and returns a preset which makes the synthesizer produce a similar target sound has been suggested. [28] [22]

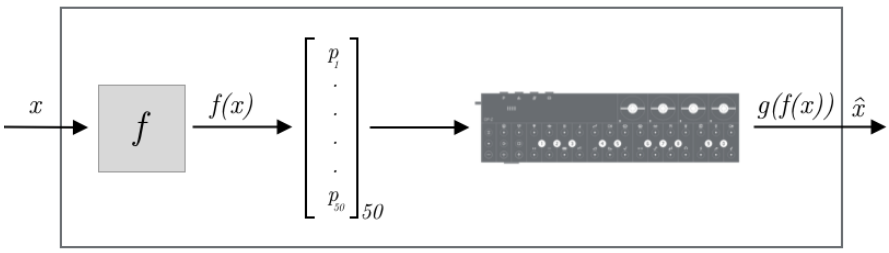

Figure 1.1: A closer conceptual view of the re-synthesis system *g(f(x))* in fig.1.1 using a synthesizer as the generative function *g(x)*. Some function *f* approximates a mapping from audio space to the high dimensional parameter space *P*. The approximated point *f(x)* - a set of parameter values in *P* serves as a preset which instructs the synthesizer how to generate the approximation of *x*. Copyright of the OP-Z synthesizer sketch belongs to Teenage Engineering.

Researchers have previously attempted to use Artificial Intelligence (AI) to perform re-synthesis and commonly using Genetic Algorithms (GA), a type of search algorithms which encodes each instance as an individual of a population. Using evolutionary concepts such as *fitness*, *crossover* and *mutation* to improve the fitness of the population over a number of generations. GA has previously showed to be an effective approach for searching through vast spaces of non-linear parameter settings, i.e. hyper parameters for training deep learning models.[29]. GA have previously been used to automate preset generation with promising results, however at large computational cost per prediction. [28] [22]

In the recent decade, however, Machine Learning (ML) has emerged as a leading technique within Artificial Intelligence, proving its capabilities in modelling numerous complex real world tasks without inferred prior human knowledge - ranging from facial recognition to playing boardgames.[26] [19] Rather than searching through an unknown search space, ML aims at approximating a model of the unknown search space by learning a mapping from an input value *x* to an output value *y*. This property make ML require significant amounts of computing power data to train, however leveraging from relatively fast "one shot" predictions compared to the trial-and-error nature of GA.

This thesis explores Machine Learning as a tool for automated re-synthesis using a Frequency Modulation synthesizer. While this subsection explains the motivation for doing so, the next subsection expands on the problem of choosing between the different approaches.

1.2 Problem

Horner et al. [10] early suggested a GA based approach to synth parameter estimation by decomposing the estimation process into subproblems. Yee-King and Roth [28] suggest that their GA based *Synthbot* system could serve as an effective assistant to humans attempting re-synthesis tasks, using the euclidian distance of the Mel Frequency Cepstrum Components as fitness function. Lai et al. [12] find that a combination of the spectral centroid and the spectral norm provides relatively fast convergence and good accuracy. Tatar et al. [22] propose a multi-objective GA (FFT, STFT and Envelope distances) in combination with clustering of the pareto front to produce multiple candidate sounds, and show that this approach can be used to achieve human expert competitive level when automating the generation of a preset for a given sound.

However, the use of GA comes at a significant computational cost per prediction. Apart from the large search space, the process is slowed down further by having to produce and extract features of all candidate sounds for every individual of every generation of the algorithm. For instance, the *PresetGen* require an average of 34 minutes to predict an optimal preset for a re-synthesizing a single target sound using a cluster of 50 machines working in parallel. Arguably, such compute power will not be available in consumer electronics, such as synthesizers, in a foreseeable future.

Instead, it is possible to train a ML model to learn the mapping from sound to parameter space, potentially leveraging from fast predictions *and* high accuracy. For instance, Barkan and Tsisris [2] evaluate a number of deep models and approaches to the problem with success. By training on sounds generated by the synthesizer from a large set of random presets, the model could learn what combinations of parameter values to use to make the synthesizer produce almost any given sound which the synthesizer is practically able to produce.

Different approaches can be taken when designing an ML system for this task. One approach would be to could train a single model to predict all parameters concurrently given some input representation of an input sound. We refer to this approach as the *concurrent* approach.

However, such an approach demands some caution. Tatar et al. [22] show that the parameter wise similarity of two presets is not necessarily correlated to the similarity of the sounds produced using those presets. Since the synth-parameters may have a highly non-linear impact on the output sound two highly similar sets of synth-parameters may produce perceptually non-similar output sounds. It may also be possible to produce identical sounds using non-

similar presets. An example of such non-linearities which may be difficult to learn is when using multiple modulating oscillators in an FM-synthesizer. The impact of the parameters of one oscillator could be either amplified or completely silenced depending on the current state of a number of other parameters. Another difficulty could be to let the model implicitly decide the number of oscillators to use and how many to silence. Horner et al [10] argue that finding the correct frequencies and modulation indices of the modulating oscillators is the most crucial step in the process. The authors go as far as stating that contrary to a concurrent approach *"the decomposition of the matching process into subproblems is central to success"* [10].

So, a second approach would be to divide the process into subproblems, training several models which make predictions sequentially depending on the predictions of other models. By explicitly designing a system of models with some models trained to make some key decisions a better system could be obtained. The models would make predictions sequentially, with each model aware of a previous model's predicitions either by taking the previous predictions as input features or by including or excluding the use of some models instead of others based on the predictions of other models. For instance, the number of oscillators used in a synthesizer could be decided by a classifier, or parameters which influence each other to a large extent could be predicted by separate regressors. We refer to this approach as the *sequential* approach.

In conclusion, the motivation for automated re-synthesis is the synthesizer's vast and complex search space which is hard for even experts to navigate. The motivation for using machine learning is the low computational expense per prediction as opposed to previously successful but computationally expensive genetic algorithms. The motivation for comparing a concurrent and sequential approach is the knowledge gap that consists in whether the problem of estimating parameters for a FM synthesizer benefits from decomposing the process into smaller sub problems, or if the problem can be solved as well or better without explicitly designed sub problems.

1.3 Purpose

The purpose of this thesis is to compare two different approaches to predicting a large number of parameter values in a high dimensional parameter space, in an attempt to fill the knowledge gap described in the previous subsection. We compare an approach of predicting all parameters concurrently with another approach of decomposing the process into subproblems, through a sequence of models which make predictions based on previous predictions. The aim of

the thesis is to answer the research question:

In machine learning based re-synthesis, does decomposing the problem into subproblems improve the performance compared to estimating all parameter values at once?

1.4 Objectives

A number of things need to be achieved in order to answer the research question. First, it clearly needs to be defined how to quantify distance between a candidate and target sound. Second, the learning problem needs to be clearly defined. Third, we need to develop the two approaches for re-synthesis. Fourth, we need to evaluate these models in a way which reflects their ability to generalize on non-synthetic data. Finally, we need to analyze this data.

1.5 Sustainability, Social Benefits and Ethics

As so often within the field of technology in general and AI in particular, the automation of tasks which are a part of somebody's job can and should be discussed. So should this: if our synthesizers can tune themselves perhaps we will not need sound designers anymore. Instead, we would simply be able to mimic a sound designed by someone else without any knowledge of sound design. Although this technology is far from at that level, this ethical aspect is important to highlight. On the positive note, learning a mapping is often more compute and energy efficient than searching for it. Since solutions are often searched for through expensive GAs, this could have some positive impact.

Finally, to cite a TE employee, the contribution towards positive social and environmental of manufacturing amusing software and hardware products for music, is making people spend time on creating and playing music rather than impacting the world negatively, ultimately bringing joy and happiness to the world.

1.6 Methodology

The research question will be answered in two steps: implementation and experiments. First, the implementation of a concurrent and a sequential approach to re-synthesis. The implementations are trained using a large set of samples which are generated with the software synthesizer. Second, using the two implementations to re-synthesize a number of instrumental samples and

quantifying the performance as a number of sound similarity metrics. This approach allows us to develop the implementations in a step wise manner, gaining domain knowledge and understanding of the two different approaches which are useful in reasoning about and understanding of the experimental results. The performance of the two implementations is measured quantitatively rather than qualitatively in order to benefit from the possibility to evaluate over a larger set of samples rather than a smaller, potentially biased set of sound samples.

1.7 Delimitations

Due to time constraints, only a limited amount of work on evaluating and reflecting around different available metrics for modelling human perceptual sound similarity - a thorough evaluation of similarity metrics for domain specific audio would arguably be a wide enough scope for a thesis itself. Instead, I rely on metrics proposed in relevant related work showing promising results.

Furthermore, the number of parameters included in the learning problem is reduced in order to reduce the complexity. More parameters would likely make the learning problem significantly harder to learn (see *curse of dimensionality* in subsection 2.5.3) which could result in both approaches performing poorly reducing generalizability of the results of the thesis.

Finally, the purpose of this thesis is not necessarily to obtain the best results possible but to obtain knowledge in which out of the two approaches perform better. For this reason, models of significantly different depth (such as Convolutional Neural Networks) will not be explored and compared, although these models would probably yield better results. Possibly, the two approaches would behave differently and the conclusion would be another using deeper models.

1.8 Outline

chapter 2 presents, a deeper study of fundamental theory including the basic concepts in sound synthesis, the TE synthesizer and its parameters and Machine Learning theory. In chapter 3 the research paradigm and metods are explained, the learning problem is defined and the two approaches are designed. In chapter 4 the implementation of the two approaches is explained in further detail. In chapter 5, fist the training and validation results of the two approaches are presented. Second, the results from the evaluation are

presented. Finally, the results are analyzed and discussed. In chapter 6, the results are concluded and future work suggested.

Chapter 2

Background

In this subsection, the background introduced in section 1 is extended upon. Relevant theory on Synthesisers and Sound Synthesis is explained, followed by a specific explanation of how the TE software synthesiser used in this thesis works. In 2.4, different approaches to the quantification of sound similarity are explained. In **??** the concept of Evolutionary Computing is explained, with extra focus on Genetic Algorithms. In **??**, the theory of Artificial Neural Networks is explained.

2.1 Synthesisers and sound synthesis

Sound synthesis is the technique of generating sound, using electronic hardware or software. This section, explains a number of fundamental components and techniques in synthesisers and sound synthesis.

2.1.1 Oscillators

An audio oscillator produces a periodic output signal with frequencies in the audio range (about 16 Hz - 20 kHz), usually in the form of either Sine, Sawtooth or Square Wave as shown in fig. 2.2. A Low Frequency Oscillator (LFO) is an oscillator which outputs signals with low frequencies, usually below 20Hz. A LFO is barely hearable to the human ear, but is used to modulate other signals, as explained in subsection **??**. A synthesiser normally have a set of multiple oscillators and LFOs.

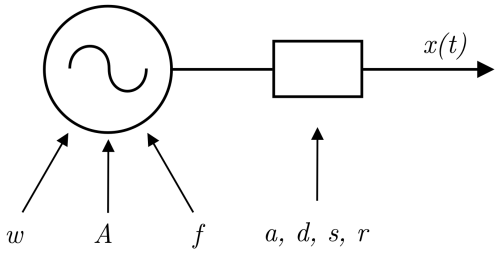

Figure 2.1: An oscillator is filtered through an ADSR envelope.

We may regard the output of an oscillator O, a function of time $x(t)$ for some waveform function w, peak amplitude A and frequency f. Below, the output equation is listet for each of the four waveform functions:

Sine Wave

$$x(t) = Asin(tf)$$

Sawtooth Wave

$$x(t) = \frac{2A}{T}tf, -T/2 \leq t < T/2$$

Square Wave

$$x(t) = \begin{cases} A, & 0 \leq t < \tau/2 \\ -A, & \tau/2 \leq tf < T - \tau/2 \quad ,A, \quad T - \tau/2 \leq tf < T \end{cases}$$

where τ denotes the puls width and is set to $\tau = T/2$ for a symmetric square wave.

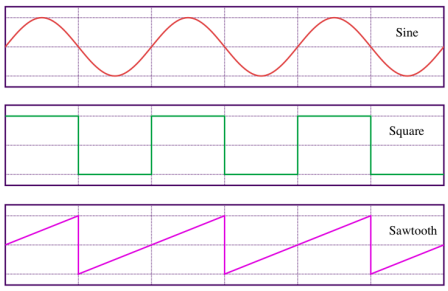

Figure 2.2: Sine, Square and Sawtooth wave forms.

2.1.2 Envelope

An envelope controls how the amplitude of a signal changes over time. The ADSR envelope consists of four parameters, see fig. 2.3, controlling the amplitude of the signal over time. The four parameters a, d, s, r control the *Attack, Delay, Sustain* and *Release* respectively. [24]

- Attack. The time taken for the signal to go from zero to peak amplitude (A), starting at $t = 0$
- Decay. The time taken for the subsequent run down from the attack level to the sustain level.
- Sustain. The level during the main sequence of the sound's duration.
- Release. The time taken for the level to decay from the sustain level to zero.

[24]

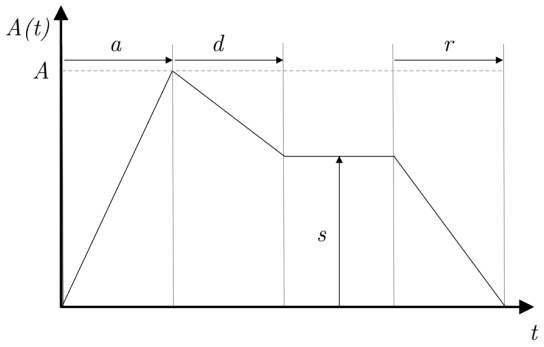

Figure 2.3: The impact of the ADSR envelope parameters on the amplitude of a signal

2.1.3 Filtering

A high-pass or low-pass filter reduces the power of low or high frequencies respectively. The parameter controlling the filter is called the *cutoff* frequency, c_h or c_l, effectively a threshold such that frequencies below or above the cutoff is reduced while the frequencies above or below the cutoff passes without modification.

2.1.4 Frequency Modulation Synthesis

Risset et al. [17] showed that the temporal evolution of the spectral components is of critical importance in the determination of the timbre of a sound. In 1973, John Chowning suggested that the already well known technique of Frequency Modulation (FM), previously used for transmitting audio signals over long distances in FM Radio, could be used to gain control of said spectral components. Chowning could show that the technique was able to yield nature like sounds in a less complex manner than before.[3] The technique of FM modulation became instrumental in the development of synthesizers in the 1980's such as the Yamaha DX7.

In FM, a modulating signal alters the frequency of a carrier signal by a rate which is the frequency of the modulating signal. The resulting signal of an oscillator O_m, modulating an oscillator O_c is given by

$$x_c'(t) = x_c(A_c f_c t + A_m I f_m x_m(t))$$

In FM synthesis, it is common to use more than two oscillators. The set up of the oscillators and how they modulate each other is referred to as an algorithm. In fig 2.4, four examples of algorithms for a synthesiser with six oscillators are shown.

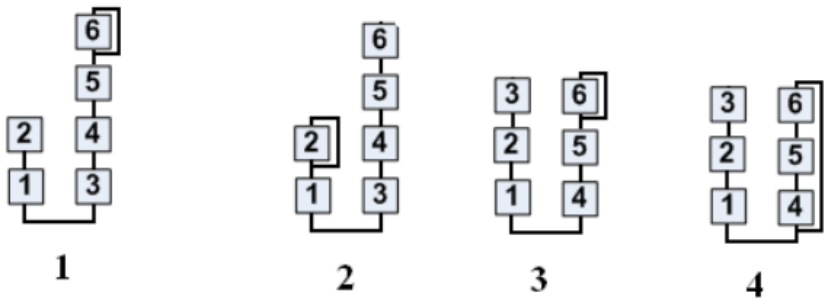

Figure 2.4: Four of the available FM-algorithms in the Yamaha DX7 synthesiser, a synthesiser with six oscillators.

2.1.5 Pitch in the MIDI Protocol

The Musical Instrument Digital Interface (MIDI) is a standard describing a communications protocol for electronic music. When used in a melodic context, the MIDI protocol can be used to describe a note being turned on and off, as well as a number of the note's characteristics such as pitch and velocity. [1]

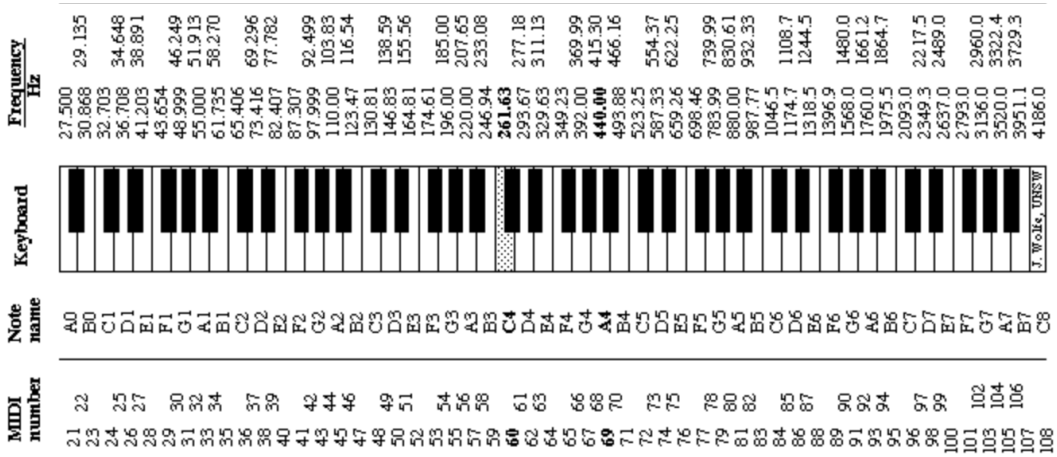

Figure 2.5: Table displaying conversion between MIDI Number, Note Name and Frequency of a pitch. Copyright belongs to Professor Joe Wolfe at the University of New South Wales and is used with permission.

The pitch of a note in MIDI is given by the note's MIDI number, an integer in the range [0, 127]. The scale ranges from the first note in the lowest octave (A0, with MIDI number 0) to the 128 half tones higher G note in the 11th octave (G11 with MIDI number 127). By convention, the frequency of note A4 (with MIDI number 69) is commonly set to 440 Hz. In fig. 2.5 a conversion table which follows this convention is shown. More formally, the MIDI number m of a frequency f is given by

$$m = 69 + 12 * log_2(f/440)$$

[27]

And conversely, the frequency of m can be obtained through

$$f = 2(m - 69)/12 * 440)$$

Given that there are 12 notes in an octave, a note with MIDI number m in octave O_i is transposed to octave O_j through

$$m_t = m + 12 * (O_j - O_i)$$

2.2 TE Synthesizer

The synthesizer to be used is a synthesizer software developed by TE. The software synthesizer creates sounds by digitally modelling a number of steps of an analogue synthesizer, including frequency modulation, filtering and delay. The synthesizer is similar to the software synthesizer in TE's *OP-Z* [*], but is developed specifically to be used in this thesis project: the software is simplified in order to protect intellectual property of TE and in order to reduce the complexity of the Machine Learning problem.

The synthesiser is configured through a number of parameters which determines how a sound is created. A set of parameter values for each of the synthesiser's parameters is called a *patch*. By feeding a patch to the synthesiser, the user controls the characteristics of the output sound.

2.2.1 Overview

- *Oscillators* Four oscillators, each creating a waveform signal $x_i(t)$.
- *Main mix* A combination of the signals of the four oscillators, filtered through a low/high pass filter and an envelope filter (see upper chain in fig 2.6).
- *Delay mix* A combination of the signals of the four oscillators is delayed and filtered through a low/high pass filter and an envelope filter. (see bottom chain in fig 2.6).
- *Output* The main mix is merged with the delay mix to form the output signal.

[*] https://www.teenageengineering.com/products/op-z

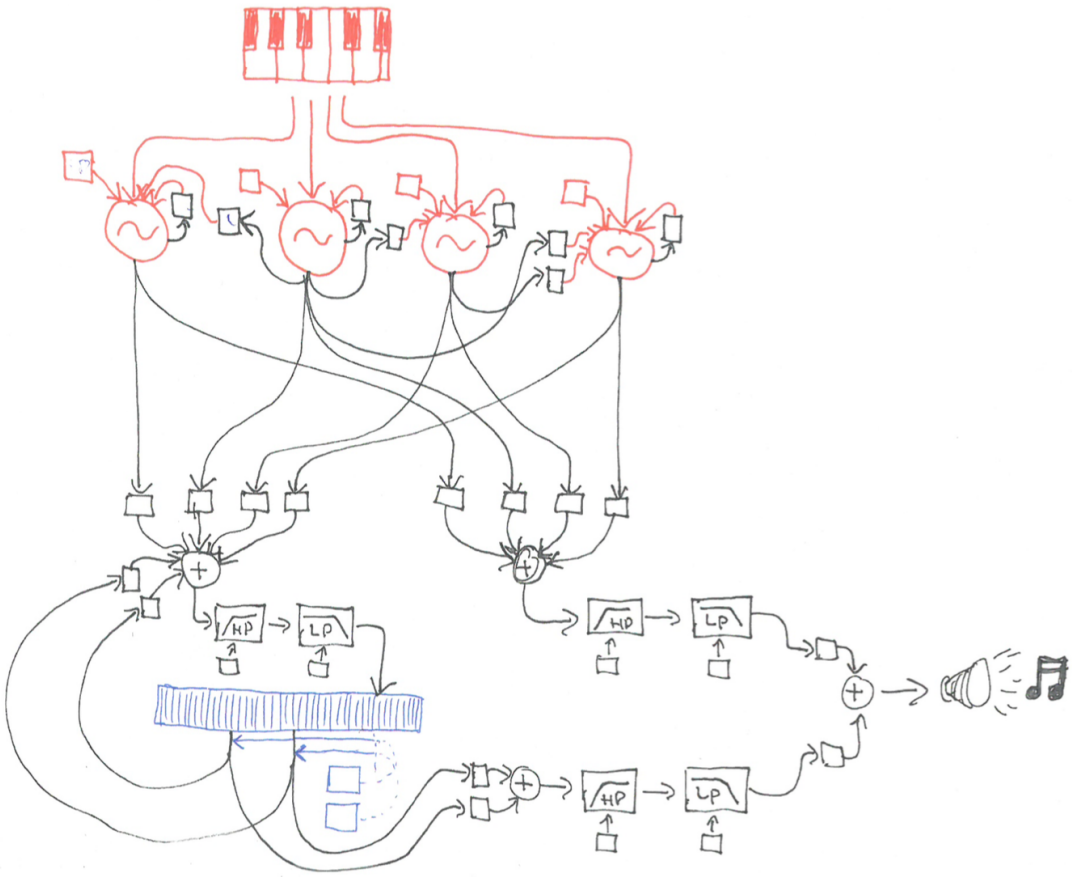

Figure 2.6: A model of the synthesizer. Playing a MIDI note triggers a set of parameters which control the four oscillators. Each oscillator can modulate the other oscillators and itself depending on the chosen modulation algorithm. The outputs of the oscillators are weighted and added once into a main mix (right) and once into a delay mix. The signal of each mix is filtered through a low-pass/high pass filter and an envelope filter. The weighted sum of the two mixes forms the output signal.

2.2.2 Meta Parameters

The following meta parameters are available:

- *Pitch*: [0, 128] The produced sounds' pitch, given as the MIDI number as described in 2.1.5
- *Duration*: [0, ∞) The duration of the created sound in milliseconds,

- *Algorithm*: [1, 8] One out of eight distinct fm-algorithms as described in 2.1.4.

2.2.3 Oscillator Parameters

For each of the four oscillators in the synthesizer, a number of parameters are available.

- *Frequency*: [-16, 16] The frequency shift of the oscillator with respect to the *pitch*. The frequency of the oscillator is given by the pitch frequency plus the frequency number multiplied by the frequency of the pitch. By other means, this is a linear manipulation of the frequency, as opposed to real octaves which is logarithmic.
- *Detune*: [-100, 100]
- *Attack*: [0, 1] The Attack of the envelope as described in section 2.1.2
- *Release*: [0, 1] The Release of the envelope as described in section 2.1.2
- *Modulation*: [0, 1] The amount the oscillator modulates another oscillator, where 0 is no modulation and 1 is "full" modulation. Which oscillator that modulates which is decided by the meta parameter *algorithm*.
- *Feedback*: [0, 1] The amount the oscillator modulates iself.
- *Mix 1 Amplitude*: [0, 1] How much is the output of the oscillator added to the mix 1 output?
- *Mix 2 Amplitude*: [0, 1] How much is the output of the oscillator added to the mix 2 (delay) output?

2.2.4 Mix parameters

For each of the two mixes, main and delay, the following parameters are available.

- *Cutoff*: [0, 1] Cutoff point of a filter as described in 2.1.3. 0.5 yields no filter, smaller than 0.5 yields a high pass filter and larger than 0.5 yields a low pass filter.

- *Resonance*: [1, 8] One of eight distinct resonance settings as described in 2.1.3
- *Envelope*: [1, 4] The envelope to apply to the mix. Setting the parameter to 1, 2, 3 or 4 corresponds to the Attach and Release parameter of oscillator 1, 2, 3 or 4.
- *Envelope Weight*: [-1, 1] -1 yields an inverse envelope filter, 0 yields no envelope and 1 yields full envelope filter.

2.2.5 Modulation

The oscillators in the synthesizer modulate each other according to a predefined modulation algorithm. In this thesis we limit the modulation to one modulation algorithm, where O_4 is the carrier, O_3 modulates O_4, $O_2 modulates O_3$ and O_1 modulates O_2 as described in fig. **??**. Each modulator has an *octave* parameter indicating the frequeny of the oscillator, envelope parameters *attack* and *release* as well as a *Mix1* parameter indicating the relative amplitude of the modulator.

2.3 Audio Feature Extraction

here exist within signal processing a variety of models for extracting the many perceptually important components of audio. Even in applying deep learning, where explicitly hand engineered features are usually disregarded in favor of implicitly learning the feature extraction as a part of the model, classic signal processing methods still play a significant role in the deep learning for audio domain.[16] While using the raw audio signal has shown promise [23] [11], older techniques such as the Cooley-Tukey Algorithm for extracting a signal's frequency components through the Fast Fourier Transform [4] is still an essential part to look into when addressing signal processing problems with ML. In this section, I explain some of the most commonly used transforms in signal processing. In section 2.4, I define a number of distance metrics using these transforms.

2.3.1 Raw Audio Signal

The raw audio signal is typically expressed as a one-dimensional array of amplitude values. The values are typically normalized to range from -1 to +1. In fig 2.7 a synthetic vocal sound represented as a raw audio signal.

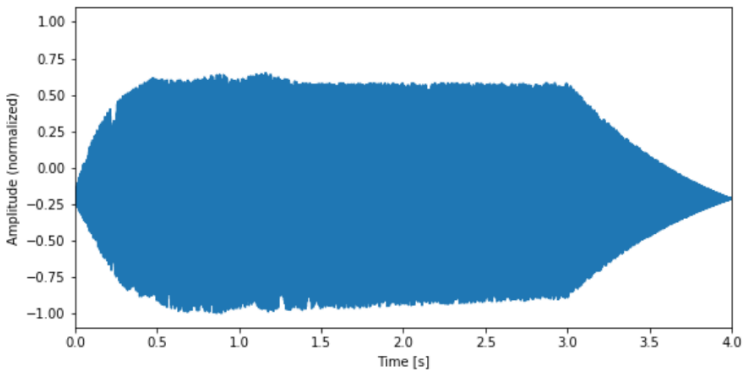

Figure 2.7: A raw audio signal representation of an instrumental sound.

2.3.2 Envelope

By applying the Hillberg transform to the signal, the analytic signal $A(t) = A_{re}(t) + A_{im}(t)$ of the signal is obtained. The estimation of the envelope, $e^*(t)$, is defined as the magnitude of $A(t)$, run through a low pass filter.

$$e^*(t) = lp(|A(t)|; f_c)$$

where f_c is some low frequency, i.e. 30 Hz.

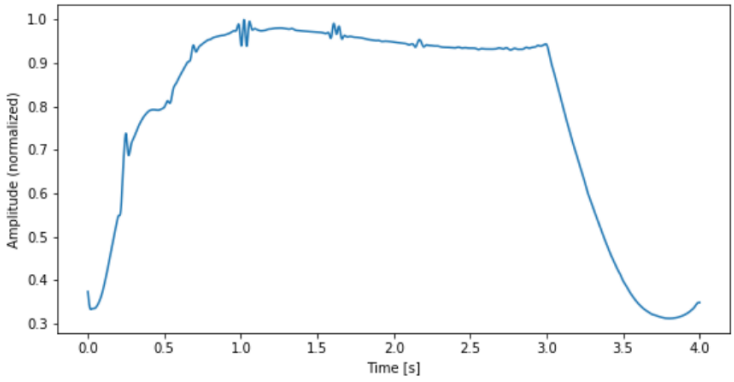

Figure 2.8: The envelope of a synthetic vocal sound as extracted through the Hillberg transform run through a low pass filter.

2.3.3 Fast Fourier Transform

The Cooley-Tukey FFT algorithm is used to obtain the array of Fourier coefficients $A_s(f)$, representing the amplitude of each frequency component for a

given signal s [4].

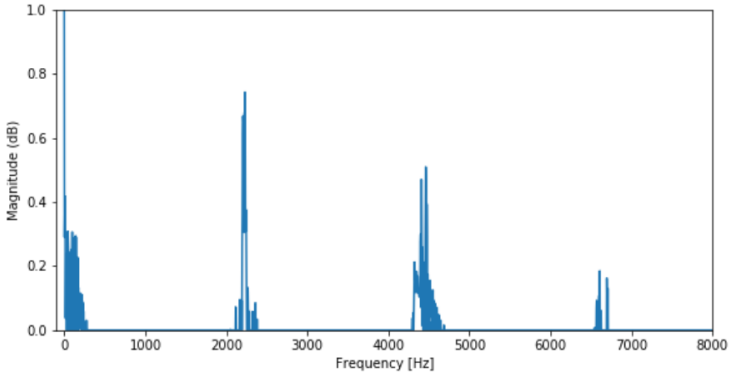

Figure 2.9: A representation of a synthetic vocal sound in the frequency domain, obtained through the Cooley-Tukey FFT algorithm.

2.3.4 Short Time Fourier Transform

The STFT spectogram is a sequence of FFTs over time, forming a spectrum of frequency component amplitudes which evolve over time.

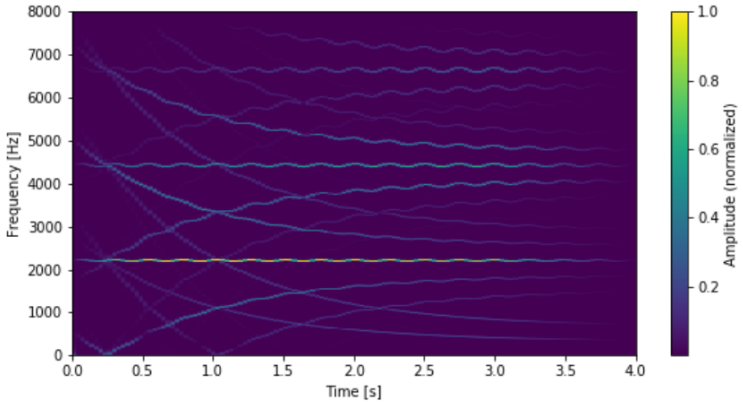

Figure 2.10: The STFT spectrogram of a synthetic vocal sound computed with a sampling rate of 16000 Hz, window size N_s of 1024 samples, and an overlap of 512 samples.

2.3.5 Mel Scale Representation

The Mel Scale was introduced by Stevens, Volkmann and Newman in 1937 [20]. As opposed to the linear Hertz scale, the Mel-scale aims accounts for

the perceptual phenomena that above 500 Hz increasingly large intervals are perceived to produce equal pitch increments, i.e. two notes with some interval i in the high frequency regions appear closer than two notes in the lower frequency regions with the same interval i.

A frequency of f Hertz can be converted to m Mel through equation 2.1, which is also displayed in fig. 2.11.

$$m = 2595 log_{10}(1 - \frac{f}{700}) \tag{2.1}$$

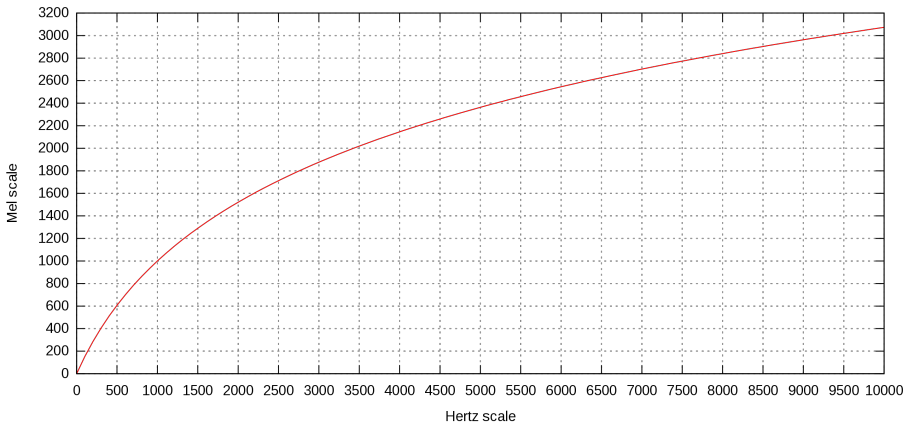

Figure 2.11: The mapping between the Mel and Hertz scales.

2.3.6 Log-Mel Spectrogram

to be added The log-mel spectrogram does similarly to the STFT model the temporal evolution of the frequency components of a sound. The log-mel spectrogram, however, has two key differences to adopt it to better model human perception: first, it maps the powers of the frequencies onto the mel-scale as described in 2.3.5 and second it expresses the powers of the mel frequencies on a logarithmic scale rather than linear. [16]

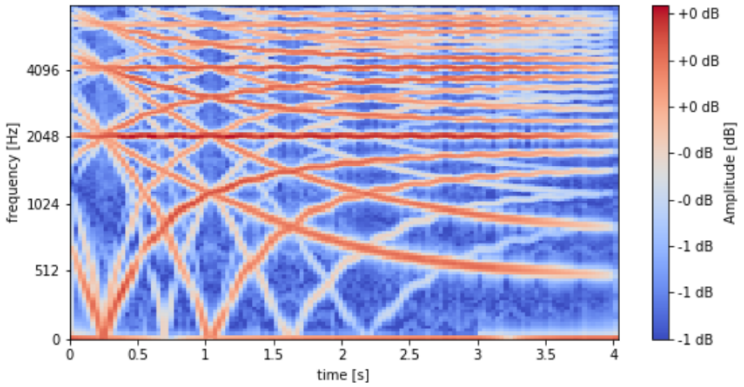

Figure 2.12: The log-mel spectrogram of a synthetic vocal sound computed with a sampling rate of 16000 Hz, window size N_s of 1024 samples, and an overlap of 512 samples

2.3.7 Spectral Entropy

The spectral entropy of a signal, measured in bits, describes the complexity of a spectrum. It is calculated by calculating the entropy of the Probability density function of the Power spectral density of the spectrum $S(x)$

The power spectral density of the signal computed by squaring the the amplitude by the number of bins:

$$P(x) = \frac{1}{N}|S(x)|^2$$

The density is normalized to a probability density function

$$p_i = \frac{P(x)}{\sum_i^N P(x)}$$

And the entropy calculated using the standard formula for entropy

$$SE = -\sum_i^N p_i ln(p_i)$$

2.3.8 Spectral Flatness

The *spectral flatness* or *Wiener Entropy* is a measure of the the noisiness/sinusoidality of a spectrum and is computed as the ratio of the geometric mean to the arithmetic mean of the energy spectrum [6]. In this thesis the log-flatness is used

in order to increase the dynamic range, making the measure range from minus infinity (a single sinusoid) to zero (complete white noise). In fig 2.13 the spectral flatness is displayed for two samples from the nsynth dataset [7].

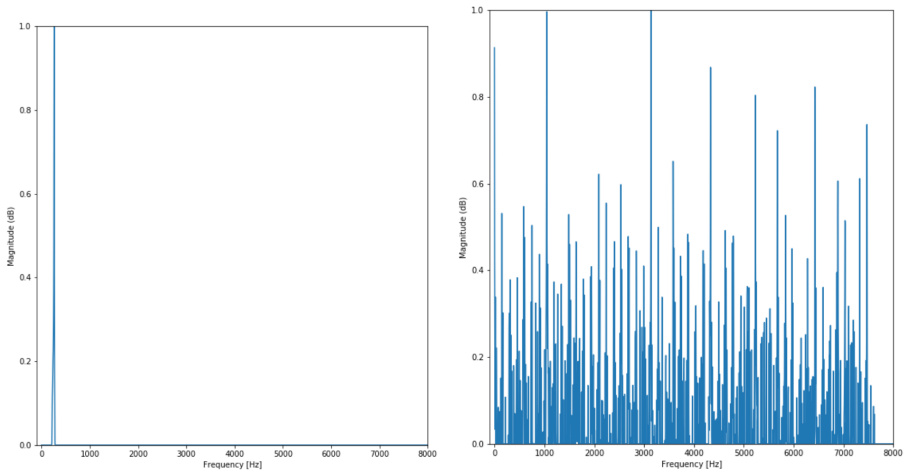

Figure 2.13: The frequency magnitude spectras of two sounds: to the left a synthetic bass with low spectral flatness (a large negative value), and to the right an electric guitar with high spectral flatness (close to zero)

2.4 Audio Similarity Metrics

Quantifying the similarity of a candidate and a target sound such that it models human perception is complex, application specific and somewhat subjective. Tatar et. al suggests a multi objective objective approach, measuring the similarity of two sounds by comparing 1. the euclidian distance of the magnitude frequency spectrums obtained through FFT over the entire sound without segmentation, 2. the euclidian distance of the spectral envelope obtained through STFT and 3. the euclidian distance of the envelopes, arguing that these metrics capture the similarity in spectral components, spectral envelope and envelope respectively. [22]. Arguably, measuring Euclidean distance for time series data may yield unintuitive results - for example the similarity between two identical series which are shifted slightly in the time domain, may be very large.

Yee-King and Roth [28] use the sum squared error of the Mel-Frequency Cepstrum Coefficient (MFCC) vectors to quantify similarity, arguing that since MFCC is largely pitch-indifferent and based on the perceptual mel scale model, it is a good model of perceptual similarity. [20].

Also in speech recognition, a domain where a word spoken at different speed or pitch still bears the same meaning, MFCC is well established [5], with the distance between the MFCC vectors commonly quantified with the Dynamic Time Warping (DTW) algorithm. [15].

2.4.1 Fast Fourier Transform

The FFT distance of a candidate sound c and a target sound t is defined as the euclidian distance of the magnitude of the two arrays A(c) and A(t) as obtained through the Cooley-Tukey algorithm as explained in 2.3.3 and then normalized such that max(A) = 1 and min (A) = 0:

$$d_{FFT}(t,c) = \sum_{n}^{N} \sqrt{(|A(t)_i| - |A(c)_i|)^2}$$

2.4.2 Short Time Fourier Transform

The STFT spectrogram S(k) is computed with a sampling rate of 44100 Hz, window size N_s of 1024 samples (23ms), and an overlap of 512 samples (11.5ms). * An example of the frequency magnitude spectrum over time extracted through STFT can be seen in the fourth plot in fig. **??**. The STFT distance of a candidate sound c and a target sound t is defined as the Euclidian distance of the two spectrums S(t) and S(c):

$$d_{STFT}(t,c) = \sum_{i=1}^{N_w} \sqrt{\sum_{j=1}^{N_s} (S(t)_{ij} - S(c)_{ij})^2}$$

2.4.3 Log-Mel Spectrogram

The log-mel spectrogram LMS(k) is essentially computed in identical manner as the STFT spectrogram. However with two differences designed to model the human perception of sound: the frequencies are mapped onto the mel-scale and the amplitude is projected on a logarithmical scale.

The log-mel spectrogram distance d_{LMS} is computed identically to d_{STFT}, but using the log-mel frequency spectrograms rather than the STFT spectro-

* These parameters are selected in order to replicate Tatar et Al.

gram.

$$d_{LMS}(t,c) = \sum_{i=1}^{N_w} \sqrt{\sum_{j=1}^{N_s} (LMS(t)_{ij} - LMS(c)_{ij})^2}$$

2.4.4 Euclidian Distance of Envelope

The envelope distance is defined as the mean of the sample wise euclidian distance of the target and candidate envelopes e_t^* and e_c^*, estimated through the Hillberg transform and a low pass filter in 2.1.2.

$$d_{envelope}(t,c) = \frac{1}{N} \sum_{n}^{N} \sqrt{(e_t^* n - e_c^* n)^2}$$

2.5 Machine Learning

Tom M. Mitchell [14] describe the field of Machine Learning as concerned with the question of how to construct computer programs that automatically improve with experience. Rather than performing a task based on some explicitly stated rules and conditions, ML allows a machine to *learn* how improve its ability to perform a task by experience. Formally, Mitchell defines the learning as:

Definition. *"A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". [14]*

In order to pose a good learning problem, we need to define three mentioned features: Task T, Performance measure P and the source of experience E. For example, consider the problem of recognizing hand written digits:

- Task T: Classify a handwritten digit in an image
- Performance measure P: the percent of digits correctly classified
- Training experience E: a database of handwritten digits with known classification.

The main categorization of learning problems is into either *supervised* or *unsupervised* learning. In supervised learning, the goal is to predict the output given a number of input features and can be seen as *"learning with a teacher"*,

where the "teacher" provides either the correct answer and/or an error associated with the predicted answer. In unsupervised learning, or *"learning without a teacher"*, there is no output value, but the goal is rather to describe the patterns among the input features. [9] In this thesis, the focus is at supervised learning.

2.5.1 Supervised Learning

For each task in supervised learning there is a set of variables, *inputs*, with some influence on one or more *outputs* or *response variables*. The goal of supervised learning is to use the inputs to predict the outputs. [9]

The output type is commonly divided into either a quantitative or qualitative measurement. The above mentioned problem of handwritten digits, where the output is one of 10 classes *{0, 1, ..., 9}* is an example of a problem with a qualitative (or *categorical*) output. By convention the prediction of quantitative variables is called *regression* whereas the prediction of qualitative variables is called *classification*. [9]

Take as example the linear model. Given an input $X = [X_1, X_2, ..., X_p]$ the output Y is predicted as

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j \tag{2.2}$$

where $\hat{\beta}_0$ is the *bias* of the model. Including a constant 1 in X and $\hat{\beta}_0$ in the vector of coefficients $\hat{\beta}$, the model can be written in vector form.

$$\hat{Y} = X^T \hat{\beta} \tag{2.3}$$

Given a set D of inputs $x_0, ..., x_i$ with known output values $y_0, ...y_i$ we can fit the model to D. The most popular way to fit the model is by minimizing the residual sum of squares with respect to β. [9]

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - x_i^T \beta)^2 \tag{2.4}$$

2.5.2 Regression

In regression, one or more continuous or discrete target variables are predicted. The error is defined as some loss function between the target. In this thesis we use the mean squared error to define the loss for N prediction $\hat{y}_i$ with true label y_i, as defined in 2.5.

Mean Squared Error

$$L = \frac{1}{N}\sum_{x}^{N}(y_i - \hat{y}_i)^2 \tag{2.5}$$

2.5.3 Generalization

One of the main goals of ML is to obtain a model which *generalizes* - i.e. makes correct predictions also on data which was not shown to the model during training.

A common problem in ML occurs when the dimensionality of the learning problem increases beyond just a few dimensions. When the dimensionality increases, the volume of the hypothesis space grows in such a pace that the training data become sparse. This phenomena is often referred to as the *curse of dimensionality*.

When a model is trained, a general model will become increasingly good at predicting the target variable correctly given not only training data but also previously unseen data. When the models ability to predict the target variable from unseen data starts decreasing, the model is said to be *overfitting*. For example, overfitting can occur due to the model being trained on too few data, being trained to many times on the training data or being trained in a too fast manner. In order to detect overfitting it is good practice to divide the available data into three subsets: a *train* dataset, an *evaluation* dataset and a *test* dataset.

- *Train*: The train data are used in training and their target variables are used as the ground truth for the model to learn the mapping from input x to target y. Usually about 70-80% of the data are in the train dataset.
- *Evaluation* The evaluation data are used in order to evaluate which hyperparameters make the model generalize well. The data are shown to the model which makes a prediction, and the target variables of the evaluation data are used to measure the general performance of the model. Usually 15-20% of the data are in the evaluation dataset.
- *Test* The test set is used to finally compare different models against each other, in a similar manner as the evaluation dataset, however testing on data which was not used in evaluation makes sure we have not chosen hyperparameters which makes the model loose generalizability.

In addition, techniques such as k-fold validation can be used in order to maximize the amount of data used in training, still evaluating the model with respect to its performance on *unseen* data.

2.5.4 Perceptron Algorithm

The perceptron algorithm, introduced by Rosenblatt in 1958 [18] is a linear classification algorithm, and the foundation of Artificial Neural Network (ANN) models. The perceptron is commonly modelled as a neuron which takes a n-dimensional vector of inputs $\vec{x}$ for a n-dimensional space, an n-dimensional array of weights $\vec{W} = [w_0, w_1, ..., w_n]$ and a *bias* w_0, and as output has some value Y which is a function of $\vec{W}$ given by the function:

$$Y = \vec{X}^T \vec{W} + w_0 \tag{2.6}$$

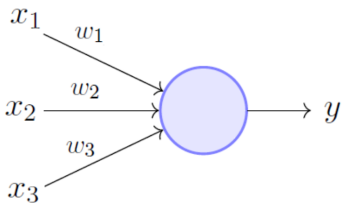

Figure 2.14: A model of a single perceptron with a three dimensional input space.

The perceptron is trained to find a decision boundary which separates the two classes as well as possible. The decision boundary can then be expressed as the linear n-dimensional hyperplane $\vec{x}$ which satisfies the equation

$$\vec{x}^T \vec{W} + w_0 = 0 \tag{2.7}$$

A separating hyperplane is found by minimizing the distance of the misclassified points (points on the "wrong" side of the decision boundary in 2.15).[9]

$$D(\vec{W}, w_0) = -\sum_{i \in M} y_i (x_i^T W + w_0) \tag{2.8}$$

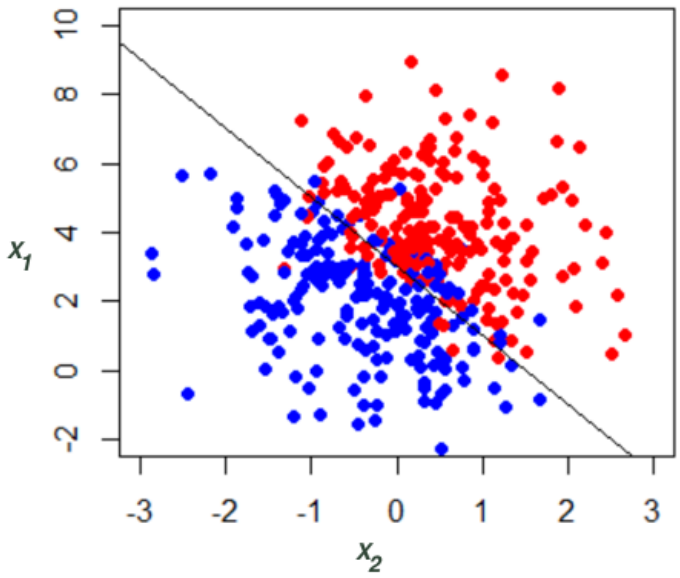

Figure 2.15: The straight line is the decision boundary between two classes in a two-dimensional space. A number of instances from both classes are one the "wrong" side of the decision boundary, and are hence misclassified.

2.5.5 Artificial Neural Network

Artificial Neural Networks is effectively a set of connected layers of perceptrons. They have proven a robust approach to approximating high dimensional functions of various types: real-valued, discrete-valued and vector-valued and can hence be used for regression and classification. [14] ANNs consists of three types of layers, see fig. 2.16:

1. *Input layer* Each of the neurons in the input layer can be seen as representing an input feature.

2. *Hidden layers* The hidden layer(s) is not visible to the outside, and can be set to an arbitrary size.

3. *Output layer* In the case of regression, the output layer has as many neurons as values predicted. The layer is then often combined with a sigmoid activation function which squashes the output to a prediction in the range [0, 1]. In the case of classification, the output layer has as many neurons as classes, with the softmax function applied to the

output vector making the output represent the predicted probability of each class.

Fig. 2.16 shows a neural network with three layers: an input layer,

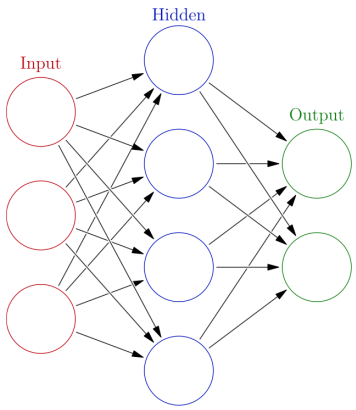

Figure 2.16: The multi-layer perceptron add caption

2.6 Related Work

2.6.1 Re-synthesis with Genetic Algorithms

Yee-King and Roth [28] propose the Synthbot, a tool which re-synthesis a given sound using any software synthesiser which complies with the Virtual Studio Technology (VST) protocol. The authors suggest a Genetic Algorithm to able to search the space of possible parameter settings of the given synthesiser. The authors argue that while similarity of the power spectrum as obtained through FFT or STFT does reward similar sounds, it regards sounds of similar timbre but different pitch to be completely different - for instance in the case of comparing two different notes played by the same instrument. Instead, the authors propose the use of the pitch independent Mel Frequency Cepstrum Coefficients (*MFCC*) to quantify the sound characteristic. The Synthbot is evaluated on two synthesisers: one FM synth with a single modulator and another subtractive synthesizer with 2 tuned oscillators and one noise oscillator.

Tatar et al. [22] also address re-synthesis through the use of a Genetic Algorithm, however they optimize and evaluate their solution for a single, more complex synthesiser: the OP-1 developed by TE. The authors propose and evaluate a Multi-Objective Genetic Algorithm to approximate a given target

sound by finding optimal synthesizer parameters. The fitness function considers three objectives to minimize: *FFT*, *STFT* and *Envelope*, modelling the frequency spectrum, the frequency envelope and amplitude envelope respectively. The use of three distinct objectives is motivated with the difficulty of choosing weights to aggregate the three into a single fitness function. Instead, the authors apply a K-means clustering algorithms to the solutions in the cumulative Pareto front, and for each cluster chose the centroid as the representative solution for the given cluster. The k solutions are presented to the user for evaluation. Finally, the authors evaluate the solutions qualitatively, by letting a number of expert sound designers attempt to re-synthesise a range of sounds using the same synthesiser. With a few exceptions the PresetGen outperform the human sound designers when compared using the three optimization goals of the GA.

2.6.2 Deep Re-synthesis

Barkan and Tsiris [2] compare several deep learning based approaches to automatic synthesizer parameter configuration. The best performing models are found to be Convolutional Neural Networks (CNN) - a special type of neural networks which use weight sharing in the early layers, effectively serving as filters which implicitly learns an optimal feature extraction. The work examines and compares two types of end to end-learning CNNs learning to predict the syntheiszer parameters given some representation of an input sound. The learning objective is defined as.... First, a CNN which takes the STFT spectrogram of a sound as input, second a CNN which takes the raw waveform as input. The authors find that while the STFT-CNN performs better, the waveform-CNN works surprisingly well. The study also concludes that large depth (number of layers) has significant positive impact on the predictive capabilities, and that larger models seem to generalize better.

2.6.3 Deep Generative Re-synthesis

Engel, Resnick et al. [7] propose a deeper approach to the problem of resynthesizing short instrumental sounds. Rather than learning how to interact with an engineered synthesizer - an explicitly programmed and to some extent *unknown* generative function - they utilize a trainable generative model which learns to produce the output sounds. Building on previous work by van den Oord et al [25] who created WaveNet, a deep neural network model for generating raw audio, they suggest an autoencoder with WaveNet as the

second, generative half of the autoencoder. First, this approach is beneficial because rather than projecting the target sound onto some possibly suboptimal parameter space according to the instructions of the synthesizer, the model can learn a more efficient encoding. Second, the model is not limited by the pre-engineered functionality of the synthesizer, but may instead learn a potentially more optimized manner of generating sounds. Notably however, training WaveNet is very computationally expensive process which take weeks even in a highly parallel setting.

2.7 Summary

To summarize, we are targeting a problem of approximating a mapping from sound space to synthesizer parameter space for a FM-synthesizer. An FM synthesizer creates rich frequency spectras by modulating a (sinusoidal) carrier frequency with other modulating frequency. By altering parameters such as frequency, amplitude and modulation index, a rich variety of sounds can be obtained. This and similar problems have previously been approached with Genetic Algorithms and Deep Learning. In order to measure the performance of a Machine Learning model performing this task, we quantify the distance between target sound t and candidate sound c with respect to four different metrics: euclidian distance of the frequency magnitude spectrums (d_{FFT}), euclidian distance of the temporal frequency magnitude spetrograms (d_{STFT}), euclidian distance of the log-mel scaled temporal frequency magnitude spetrograms (d_{LMS}) and finally euclidian distance of the envelopes ($d_{envelope}$).

Chapter 3

Methodology

This chapter gives an overview of the research method used in this thesis. Section 3.1 describes the research paradigm and the research methods used with the help Anne Håkanssons work *Portal of research methods and methodologies for research projects and degree projects* [8] which summarizes and concludes some research methods and their importance. Section 3.2 describes the method outline in further detail. Section 3.3 describes the learning problem given theory and delimitations mentioned in previous chapters. Section 3.4 describes the design of the two compared approaches. Section 3.5 describes the datasets used for training, validation and evaluation including generation of the self-generated dataset. Section 3.6 describes the use of techniques for measuring sound similarity, relating back to the sound similarites mentioned in the theory section 2.4. Section 3.7 describes the experiments conducted to compare the two approaches. Finally, section 3.8 describes the means by which the data gathered is analyzed.

3.1 Research Paradigm

Multiple approaches could be chosen in order to answer the research question stated in 1.2. For instance, a theoretical approach could have been chosen, consisting of a literature study aimed at answering the research question. Such a theoretical approach allows for significantly larger diversity in the approaches and datasets, possibly enlargening the scope of the research and minimizing the risk of biased data or human errors. However, this is approach was not chosen as this thesis aims to cover a knowledge gap in existing litterature. Moreover, the research partnership with Teenage Engineering offers a unique experience to conduct empirical research with an advanced software synthe-

sizer which is normally not available to the public or research community. For these reasons, the research question is instead attempted to answered empirically through an experimental research strategy, and the question is answered by designing and training two different approaches and quantitatively assessing their respective ability to automatically generate synthesiser presets.

3.1.1 Research Methods

According to Håkansson [8], research methods are the methods applied to the degree project in order to facilitate the research process, and provide procedures for *how* to do research, including initiating, carrying out and completing the tasks in research. Håkansson lists a number of the most common research methods, including *experimental research* which is described as a quantitative approach which *"establishes relationsships between variables and finds causalities between the relationships"*. [8] Given the mathematical nature of ML, a quantitative research method is suitable, and given the previously mentioned step wise manner in which the research will be conducted *experimental research* is chosen as the research method of this thesis. Given the perceptual nature of sound, it would arguably be interesting to also collect qualitative data as well. This could be done by allowing a test group to listen to sounds from the two approaches and comparing which perform better at re-synthesis. Allowing real humans to evaluate the approaches would indeed be superior to using sound similarity metrics since the metrics do not perfectly model human perception. However, it would be difficult to carry out such an evaluation with enough samples to yield a trustworthy test. Arranging such a test setup would likely require significant time, with the risk of obtaining results which would not be significant. Instead, a quantitative approach is chosen such that many samples can be evaluated, with awareness of the risk of the chosen sound similarity metrics not perfectly modelling human perception.

3.1.2 Research Approach

The research approach is used to decide how to draw conclusions from the collected data. Håkansson [8] lists two main approaches and one hybrid approach:

- *Inductive* After gathering enough data, the data are analyzed in order to gain knowledge and establishing different views of the researched phenomenon. Commonly used in qualitative research.

- *Deductive* Large data sets are used to verify or falsify theories or hypotheses. Commonly used in quantitative research.
- *Abductive* Uses both inductive and deductive approaches to establish conclusions through reasoning. The outcome is an explanation or reasoning, rather than test of an hypothesis.

Given the experimental yet quantitative nature of this thesis, the abductive research approach is chosen. More specifically, I choose to develop an an artifact per approach. This allows for obtaining domain knowledge in general and knowledge of the two implemented approaches in particular, which aids the reasoning around and understanding of the results obtained in the experiments conducted with the artifacts.

3.2 Method Outline

The research is conducted in a step wise manner, starting by applying theory in practice on toy problems, where after the problem difficulty of the number of predicted parameters are increased step by step until the learning problem is complex and large enough to satisfyingly being able to answer the research question. This method is adopted to in an agile pace establishing domain knowledge, without the risk of spending vast amounts of time implementing or training models which eventually potentially do not work for the given problem. Eventually, the domain knowledge is used to train two models for the given learning problem:

- *Concurrent approach* A model which is trained to predict all parameter values of the learning problem at once. A single regression model.
- *Sequential approach* A set of models which are trained to predict the parameter values of the learning problem in a step wise manner. A mix of classifiers and regression models.

The two models are trained on a dataset generated by the TE syntheziser. Finally, the performance of the models are evaluated by measuring their capability to re-synthesize a number of instrumental sounds from different instrumental sources in three different pitches. The capability is quantified with respect to four different targets: FFT-distance, STFT-distance, log-mel spectrogram distance and envelope distance.

3.3 The learning problem

As mentioned in the delimitations section 2.2 are fixed or disabled to reduce some complexity of the learning problem. The delay chain is completely muted, the modulation is set to 1, the FM algorithm is set to always be the same 0, the global envelope is always the same as the carrier oscillator's envelope and the weight of the envelope is always set to 1. The parameters to learn are listed in table **??** and their effect is visualized in figure 3.1.

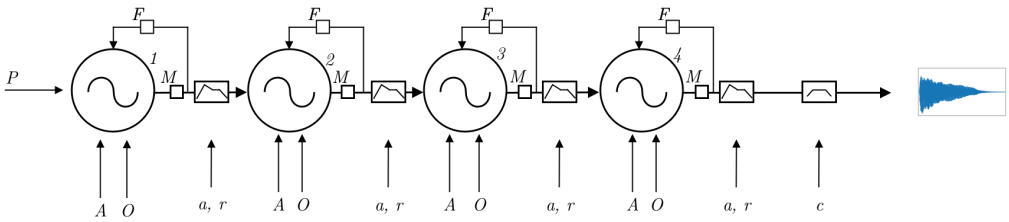

Figure 3.1: The impact of the trained parameters on the output sound. The modulation algorithm used is a straight line where osc 1 modulates osc 2 which modulates osc 3 which modulates osc 4. Each modulator has an *frequency* parameter O indicating the frequency of the oscillator, an *amplitude* parameter A indicating the relative amplitude of the output signal, a feedback modulation index F, envelope parameters *attack* a and *release* r. At the end of the algorithm there is a global envelope and a low/high pass filter which is controlled through a cutoff parameter.

3.4 Designing the competing approaches

The purpose of this thesis is to compare two approaches, the *concurrent* and the *sequential*. In order to compare the two approaches well, some caution needs to be taken when designing the models in order to facilitate for a fair comparison.

Choosing the size of a model, the number of parameters, is usually a trade off. A small model may be fast and computationally inexpensive to train and may generalize well, however it may be incapable of modelling the given learning problem. A large model is better capable to model high dimensional learning problems, but may also be too strong such that it learns to model the noise included in the data, making the model overfit as described in 2.5.3.

Very large models are also expensive to train and may require dedicated hardware such as virtual machines. The learning problem defined in 3.3 is high dimensional and models will likely benefit from size. Furthermore some architectures may be better at modelling certain problems, making it important to choose similar learning algorithms to facilitate for a just comparison.

In order to allow for a good comparison, both approaches will be ANNs, consisting of a roughly equal amount of parameters and trained for roughly the same number of epochs on the same data. As input, both models will take pitch extracted from a pitch prediction model and the STFT spectrogram of the input sound, as defined in subsection 2.3.4.

3.4.1 Concurrent Approach

The concurrent approach is an ANN with three hidden layers, taking the STFT-spectrogram downsampled by a factor of 16 and the predicted pitch as input.

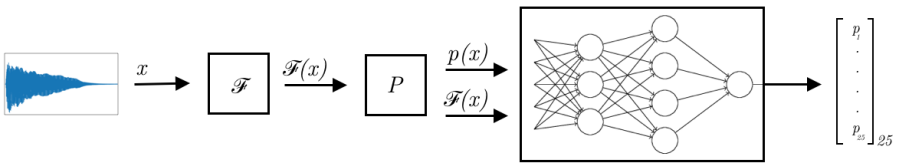

Figure 3.2: An overview of the concurrent approach of predicting the parameter values for resynthesis. The input sound x is passed through a feature extraction function f. The extracted features f(x) is used as input to a pitch predictor P to predict pitch p(x). f(x) and p(x) are used as input to a regression model which predicts all parameter values.

3.4.2 Sequential Approach

The sequential approach is designed as a sequence of models. First, the pitch and STFT-spectrogram is extracted. Then, a classifier classifies whether to use 0, 1, 2 or 3 modulating oscillators. If 1, 2 or 3 is predicted, the model trained to predict the parameters frequency, amplitude, attack, release and feedback of 1,

2 or 3 oscillators is used. If 0 is predicted, this step is skipped. Finally, a model predicts the cutoff filter and the global envelope given the pitch and previous predictions as input. All of the models are ANNs, however of various size and depth. This can be summarized by figure 3.3 in the following pipeline:

1. The carrier oscillator frequency is set according to the predicted pitch.
2. A classifier predicts the number of modulating oscillators (0, 1, 2 or 3).
3. Depending on the number of modulating oscillators, the frequency, amplitude, attack, release and feedback parameters of each modulator is predicted.
4. The global attack and release and cutoff parameters are predicted.

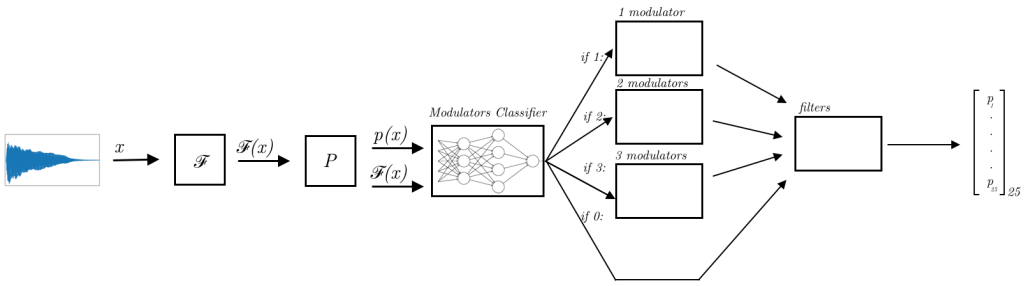

Figure 3.3: A sketch of the sequential approach of predicting the parameter values for resynthesis of input sound x. The pitch p(x) and features f(x) are extracted. p(x)

3.5 Datasets

Two datasets are used within this thesis. First, a dataset of sounds generated with the software synthesiser and second, the Nsynth dataset from Google's Magenta project. [7]

3.5.1 Self-Generated Dataset

The self-generated dataset is designed to cover the as much of the variety of the synthesizer's parameter space as possible. As discussed in **??**, we limit the dataset to only use only Mix1 in order to reduce complexity.

There are four modulators which can be configured through 8 different algorithms. We choose to use all four of the modulators, but always configured in the same modulation algorithm. The modulation algorithms can be seen in fig. **??**. We always use a carrier oscillator with *Octave* parameter set to 0. With equal probablility, we choose to use either 0, 1, 2 or 3 modulating oscillators. For each of the oscillators we choose values for the *Octave*, *Feedback*, *Amplitude* and *envelope* parameters. A set of global parameters are also chosen at random: Filter cutoff and *pitch* while *Mix1 envelope weight*, *Mix1 envelope index* and *Mix1 resonance* are fixed to a constant value.

The dataset is created by generating a set of random presets and feeding them to the TE software synthesizer. The dataset was produced according to the following procedure:

1. Generate a random patch
2. Mute all four oscillators $O_1 ... O_4$ by setting their *Mix1*, *Mix2* and *detune* parameters to zero.
3. Set the modulation *algorithm* to algorithm no 1
4. Set the *pitch* to a random integer in [21, 108]
5. Set the *Octave* of O_4 to 0.
6. Set the *Mix1* of O_4 to 1.
7. Set the *Attack* and *Release* of O_4 to random integers in the intervals [0, 0.5] and [0.1, 0.7] respectively.
8. Choose a random number of modulators in the interval [0, 3]
9. For each used modulating oscillator O_i:
   (a) set mix1 to a random number in [0, 1]
   (b) set octave to a random integer in [-8, 8]
   (c) set modulation to 1
   (d) Set the *Attack* and *Release* to random integers in the intervals [0, 0.5] and [0.1, 0.7] respectively.
10. Set the *cutoff* parameter to a random number in [0, 1]

# of modulators	# of samples	Proportion
0	66521	24.99%
1	66180	24.86%
2	66520	24.99%
3	66929	25.14%
Total	**267000**	**100%**

Table 3.1: Each sample in the dataset belongs to exactly one out of four classes based on the number of modulating oscillators used to produce the sound. It can be seen in the table that the dataset is balanced between the classes.

Parameter	mean	min	25%	50%	75%	max
Osc1Mix1	0.14	0.0	0.0	0.0	0.11	1.0
Osc1Frequency	0.01	-8.0	0.0	0.0	0.0	8.0
Osc1Feedback	0.63	0.38	0.5	0.63	0.75	0.87
Osc1Env1	0.3	0.0	0.15	0.25	0.44	0.50
Osc1Env4	0.42	0.09	0.26	0.42	0.58	0.70
Osc2Mix1	0.28	0.0	0.0	0.1	0.55	1.0
Osc2Frequency	0.0	-8.0	0.0	0.0	0.0	8.0
Osc2Feedback	0.63	0.38	0.5	0.63	0.75	0.87
Osc2Env1	0.28	0.0	0.14	0.28	0.42	0.50
Osc2Env4	0.41	0.09	0.25	0.41	0.57	0.70
Osc3Mix1	0.41	0.0	0.1	0.4	0.7	1.0
Osc3Frequency	0.0	-8.0	-3.0	-0.0	3.0	8.0
Osc3Feedback	0.62	0.38	0.5	0.62	0.75	0.87
Osc3Env1	0.27	0.0	0.13	0.26	0.39	0.50
Osc3Env4	0.41	0.09	0.25	0.41	0.56	0.70
Osc4Mix1	1.0	1.0	1.0	1.0	1.0	1.0
Osc4Frequency	0.0	0.0	0.0	0.0	0.0	0.0
Osc4Feedback	0.62	0.38	0.5	0.62	0.75	0.87
Osc4Env1	0.25	0.0	0.13	0.25	0.38	0.5
Osc4Env4	0.4	0.1	0.25	0.4	0.55	0.7
Mix1Cutoff	0.5	0.0	0.25	0.5	0.75	1.0
Mix1Resonance	0.0	0.0	0.0	0.0	0.0	0.0
Mix1EnvIndex	3.0	3.0	3.0	3.0	3.0	3.0
Mix1EnvWeight	1.0	1.0	1.0	1.0	1.0	1.0
Pitch	63.12	20.0	41.0	63.0	85.0	106.0

Table 3.2: Mean, minimum, maximum values of the dataset. When an oscillator is turned off the parameters are set to 0, causing gradually skewed distributions ranging from oscillator 4 (always used) to oscillator 1 (used in 25% of the patches.))

3.5.2 Nsynth Dataset

In a second phase, in order to increase complexity and evaluate how well the combination of deep learning model and software synthesizer can generalize, the Nsynth dataset is used. The dataset, consisting of sounds which are unknown to the TE software synthesiser, consists of 305,979 tones from 1006

instruments in 11 families, each with a unique pitch, timbre, and envelope. The method of sound production for each note's instrument - the instrument source - is categorized as either *acoustic*, *electronic* or *synthetic*, which facilitates for evaluating the model on different types of instruments. [7]. In table 3.3 the number of samples per instrument family and source.

Family	**Acoustic**	**Electronic**	**Synthetic**	**Total**
Bass	200	8,387	60,368	68,955
Brass	13,760	70	0	13,830
Flute	6,572	35	2,816	9,423
Guitar	13,343	16,805	5,275	35,423
Keyboard	8,508	42,645	3,838	54,991
Mallet	27,722	5,581	1,763	35,066
Organ	176	36,401	0	36,577
Reed	14,262	76	528	14,866
String	20,510	84	0	20,594
Synth Lead	0	0	5,501	5,501
Vocal	3,925	140	6,688	10,753
Total	**108,978**	**110,224**	**86,777**	**305,979**

Table 3.3: Number of samples by instrument family and source in the NSynth dataset.

For evaluation, a subset of the Nsynth dataset is used. For each of the instruments, the samples of the instrument in pitch 36, 60 and 84 is chosen. This yields a dataset of 2255 samples.

Family	Acoustic	Electronic	Synthetic	Total
Bass	74	2	393	469
Brass	0	117	0	117
Flute	0	46	15	61
Guitar	129	87	36	252
Keyboard	308	56	30	394
Mallet	38	199	14	251
Organ	309	0	0	309
Reed	0	118	3	121
String	0	178	0	178
Synth Lead	0	0	37	37
Vocal	0	24	42	66
Total	**827**	**858**	**570.0**	**2255**

Table 3.4: Number of samples by instrument family and source in the NSynth dataset.

3.6 Sound Similarity

In order to train the machine learning models, it is fundamental to define how to quantify the similarity between a target sound t and a candidate sound c. As explained in 2.4, the mix of metrics which model human perception can be subjective and vary depending on the application. The following distance measures are used:

- *FFT-distance* We use FFT-distance as one of the targets in order to evaluate how well the two different approaches are able to reproduce the frequency magnitude spectrums of target sounds through re-synthesis.
- *STFT-distance* We use STFT-distance as one of the targets in order to evaluate how well the two different approaches are able to reproduce the changes of the frequency magnitude spectrum and envelope over time.
- *Log-mel-spectrogram-distance* We use LMS-distance as one of the targets for similar reasons as the STFT distance, however through a metric which better models human perception.

- *Envelope-distance* We use envelope-distance as one of the targets in order to evaluate how well the two different approaches are able to reproduce the envelope.

3.7 Experiments

The two approaches are, as previously mentioned, trained on a train dataset and validated on a validation dataset. Both train and validation datasets are generated with the TE synthesizer, and are suitable for training since we know the ground truth of the parameters used to create each sample. However, one of the main goals is generalization - being able to reproduce or approximate sounds which were not originally produced by the TE synthesizer. By just validating on generated data, there remains a risk of overfitting both approaches.

Instead, we want to evaluate their ability to re-synthesize sounds which are not produced by the TE-synthesizer, and even sounds which are not originally synthetic but electronic or even acoustic. For this purpose, we evaluate upon a subset of the Google Nsynth dataset described in 3.4.

For each sound sample t in the evaluation dataset:

1. Get the Constant Q-transform spectrogram of t, Q_t
2. Extract the pitch from t, p_t given Q_t
3. Extract the STFT spectrogram of t, S_t
4. Predict the parameter values $\hat{P}_c$ using the *concurrent approach* given p_t and Q_t
5. Predict the parameter values $\hat{P}_s$ using the *sequential approach* given p_t and Q_t
6. Synthesize a candidate sound c_c from $\hat{P}_c$
7. Synthesize a candidate sound c_s from $\hat{P}_s$
8. Quantify the distance between c_c and c_s for each of the four distance metrics: FFT-distance, STFT-distance, LMS-distance and envelope-distance.

3.7.1 Hardware Environment

All models are trained on a virtual machine with 16 GB virtual RAM and an Nvidia Tesla CPU.

3.7.2 Software Environment

The machine learning parts of this project is carried out using the following python software packages:

- *Tensorflow 1.13* A machine learning library. [13]
- *Librosa 0.6.3* A signal processing library.
- *Python 3.7.3 wave file module* Modules used to read/write .wav files.
- *Scipy 1.2.0* For numeric operations and statistical testing.
- *TE Software Synthesizer* A software synthesizer developed by TE. Copyrighted and closed source.

3.8 Data Analysis

With the data obtained from the evaluation as described in subsection **??**, we compare the two approaches. Mainly, we seek to understand if one of the models perform significantly better than the other. Furthermore, we aim to gain knowledge of for what categories of sounds either approach is favorable over the other.

First, we compare the performance of the approaches through a z-test with 99.9% confidence. For each metric we test the null hypothesis that *there is no difference between the performance of the concurrent and sequential for the given metric.* Through doing these z-tests. We also display some examples of target and candidate sounds from both models.

We group the target sounds by spectral entropy and spectral flatness as defined in **??** and **??** respectively, and compare the model's performances in terms of STFT-distance in order to draw conclusions regarding the performance given sound spectrograms with various complexity. Finally, we group the target sounds by instrument family and instrument source in order to evaluate whether there is any difference in the approaches' performances in this regard.

Chapter 4

The Re-synthesizer

This section describes the implementation of the re-synthesizer, namely the prediction pipeline, each of the models in the prediction pipeline, the test pipeline and finally the implementation environment.

4.1 Re-synthesis Pipeline

The re-synthesizer is implemented to take a target sound as input, produce a candidate sound and measure the distance between the target and candidate sound. Step wise, the procedure is as follows:

1. Extract features from the target sound
2. Predict pitch parameter
3. Predict parameter values through concurrent or sequential approach
4. Synthesize a candidate sound with the synthesizer given the parameter values
5. Measure the FFT, STFT, LMS and Envelope distances

4.2 Pitch prediction

Both the concurrent and sequential approach use the pitch as an input feature. For this reason, we train a model to extract the pitch of the given sound, defined as the MIDI pitch of the carrier oscillator. The model is trained on the train part of the Nsynth dataset of MIDI instruments.

The model is an Artificial Neural Network (ANN) predicting one output variable given the Constant Q transform spectrogram of the target sound downsampled by a factor of 16 as input. Our efforts show that large models seem to generalize significantly better. The final model has three fully connected layers of 1200 nodes each, with a total of 3,691,201 trainable parameters.

4.3 Concurrent approach

The concurrent approach is implemented as an ANN predicting 18 output variables given the predicted pitch and the STFT spectrogram as input. The model has three hidden layers with 700 neurons each. In total, the model has 3,855,600 trainable parameters. The model uses the Relu activation function and is trained for 1300 epochs.

$$f_{concurrent}(S(X), pitch) = \begin{bmatrix} \hat{A_1} \\ \hat{O_1} \\ \hat{F_1} \\ \hat{a_1} \\ \hat{r_1} \\ . \\ . \\ \hat{A_3} \\ \hat{O_3} \\ \hat{F_3} \\ \hat{a_3} \\ \hat{r_3} \\ \hat{a_4} \\ \hat{r_4} \\ \hat{C_f} \end{bmatrix}_{18} \tag{4.1}$$

	Size	Trainable parameters
Input	4105	-
Layer 1	700	2874200
Layer 2	700	490700
Layer 3	700	490700
Output	18	12618
Total		**3855600**

Table 4.1: The number of parameters of the concurrent approach.

4.4 Sequential Approach

The sequential approach is as described in the method section a number of sequentially ordered ANNs for classification and regression. The approach takes the STFT spectrogram downsampled by a factor of 16 and the predicted pitch as defined in 4.2. In this section, the implementation of each model is described in further detail. In table 4.2 the implemented size of each model is listed, the number of parameters is in the same order as the concurrent model.

4.4.1 Number of modulators classifier

The Number of Oscillators-classifier was designed to be a classifier rather than a regression under the assumption that although the choice is between 0, 1, 2 or 3 modulating oscillators, the distance between the classes might not be linear. The model is an artificial neural network with two hidden layers. The output layer has four output nodes with a softmax activation function, and is optimized with respect to the categorical cross-entropy loss as defined in eq **??**. Given the STFT-spectrogram of the input sound X, the model predicts the probability of each class:

$$f(x) = \left[P(0|x), P(1|x), P(2|x), P(3|x)\right] \tag{4.2}$$

4.4.2 Predicting Modulator Parameters

Based on the prediction of the classifier in 4.4.1, it is time to chose the parameters controlling the either 0, 1, 2 or 3 modulators.

Three different models are trained for the purpose. Given the log-mel spectrogram and predicted pitch of an input sound X, the model predicts the

frequency O and the relative amplitude A, the feedback modulation F and envelope parameters a and r of 1, 2 or 3 modulators. The predicted parameters for a model given N modulators is:

$$f_N(S(X), p) = \begin{bmatrix} \hat{A_1} \\ \hat{O_1} \\ \hat{F_1} \\ \hat{a_1} \\ \hat{r_1} \\ . \\ . \\ \hat{A_N} \\ \hat{O_N} \\ \hat{F_N} \\ \hat{a_N} \\ \hat{r_N} \end{bmatrix}_{3N} \tag{4.3}$$

The models are trained through optimizing with respect to the loss function, the mean square error as defined in eq.2.5. All of the models are ANNs, however hyper parameters have been tuned manually through grid search for each model.

4.4.3 Filters: Envelope and Cutoff frequency

For a given sound x, the model is trained to learn the *attack* and *release* parameters of the global envelope filter and the *cutoff* parameter of the high/low pass filter. The model is an ANN with three nodes in the output layer. Given STFT spectrogram of x downsampled by a factor of 16, and the previously predicted oscillator variables, the model predicts:

$$y(x) = \begin{bmatrix} \hat{a} \\ \hat{r} \\ \hat{c} \end{bmatrix}$$

The model is trained on 80% of the self-generated dataset and validated on the remaining 20% of the dataset. For a batch D of training samples, each labelled with a 3 x 1 target tuple t the loss function is defined as the mean squared error, as defined in equation 2.5.

n mod	**Size**	**Trainable parameters**
Input	4105	-
Layer 1	150	615900
Layer 2	150	22650
Layer 3	150	22650
Output	5	755
Total		**661955**

1 mod	**Size**	**Trainable parameters**
Input	4105	
Layer 1	200	821200
Layer 2	200	40200
Output	5	1005
Total	-	**861400**

2 mod	Size	**Trainable parameters**
Input	4105	
Layer 1	200	821200
Layer 2	200	40200
Output	10	2010
Total	-	**861400**

3 mod	Size	**Trainable parameters**
Input	4105	-
Layer 1	200	821200
Layer 2	200	40200
Output	15	3015
Total		**864415**

filters	Size	**Trainable parameters**
Input	4105	-
Layer 1	100	410600
Layer 2	100	10100
Output	3	303
Total		421003

Total		**3670173**

Table 4.2: The number of trainable parameters of the models in the sequential approach.

Chapter 5

Experimental Results

In this chapter, we first present the training and validation results of the two respective models. We then present the results of the evaluation and show some examples of re-synthesized sound spectras. Finally, we discuss the results.

5.1 Training and validation

This subsection displays the training and validation results of the different models in the concurrent and sequential approaches as well as the pitch prediction model.

5.1.1 Pitch prediction

In fig. 5.1 the confusion table of the model illustrates the capabilities of the model. It can be seen that the model performs better in the mid frequency range, between MIDI numbers 40 and 80, and is less accurate given sounds of high frequency.

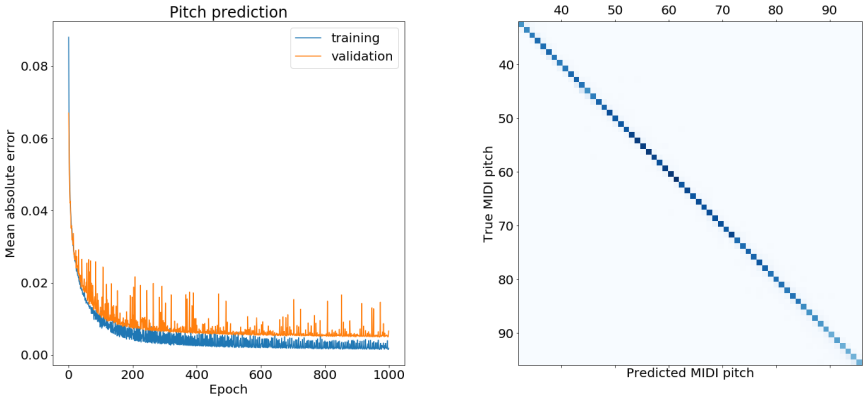

Figure 5.1: To the left, training and validation loss of the pitch regression model. To the right, confusion matrix of the pitch regression model on the validation dataset. The model is less accurate on sounds with pitch in the high and low frequency ranges, and as most accurate for sounds with MIDI number 40-70.

5.1.2 Concurrent approach

The concurrent approach is trained for 1000 epochs. The validation and training error does not show signs of overfitting.

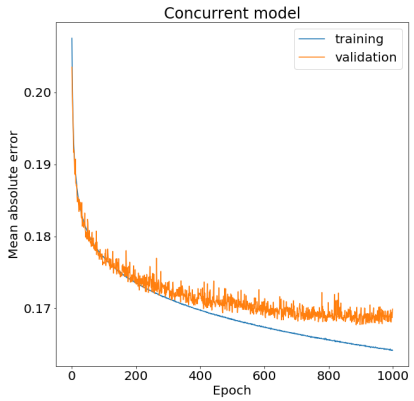

Figure 5.2: The training and validation loss of the full model.

5.1.3 Sequential approach

In this subsection the results of each part of the sequential approach is displayed.

Number of modulators

In fig 5.3 we can see that the model does not seem to overfit. It can be seen the model is accurate at detecting the use a single sinusoid without 0 modulating oscillators. However more instances with 1, 2 or 3 modulators are more often misclassified, and the model is slightly biased towards predicting more modulators.

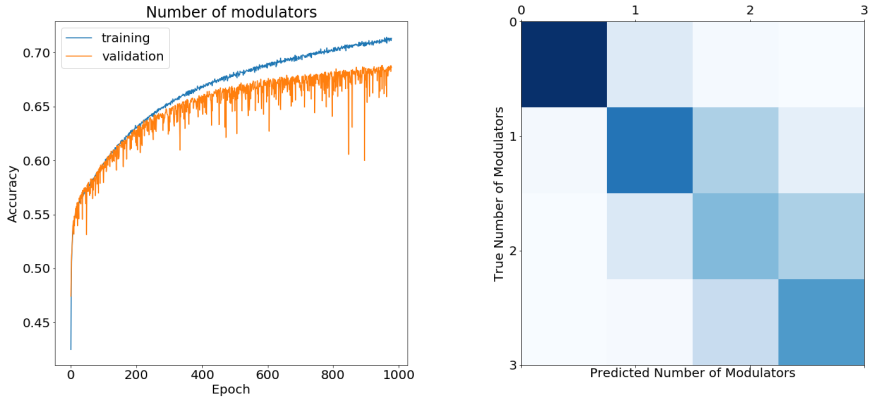

Figure 5.3: To the left, training and validation accuracy of the classifier. To the right, validation confusion matrix of the classifier.

1 modulator

In figure 5.4 the training loss and confusion matrix of the 1-modulator model is shown. The model does not seem to overfit, the validation error is 0.13. It can be seen that the model is more accurate at detecting modulators with lower frequencies and less accurate at detecting high frequencies.

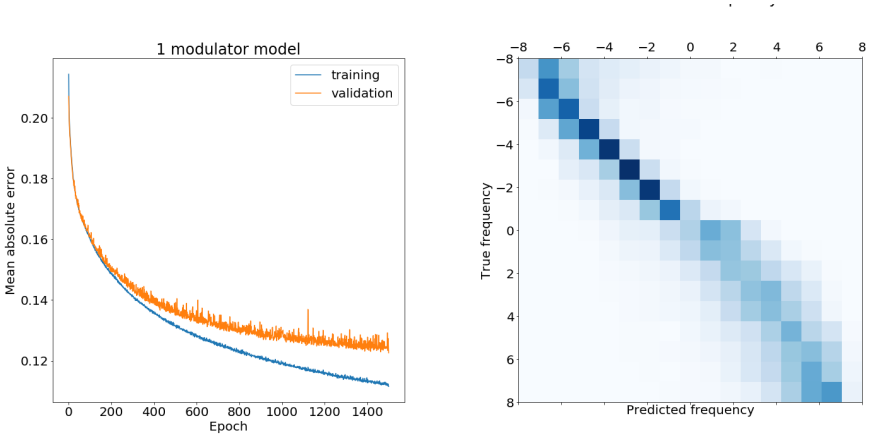

Figure 5.4: To the left the training and validation loss of the 1 modulator model. To the right the confusion matrix of frequency prediction by the 1 modulator model on the validation set. It can be seen that the model is better at detecting the pitch of a modulator in the lower frequencies, whereas less accurate in the higher frequencies.

2 modulators

In figure 5.5 the training loss and confusion matrix of the 2-modulator model is shown. The model does not seem to overfit. From the validation error at 0.185 and the more blurry confusion matrices it can be seen that the model is significantly less accurate than the 1-modulator model. The model is more accurate at detecting modulators with low frequencies.

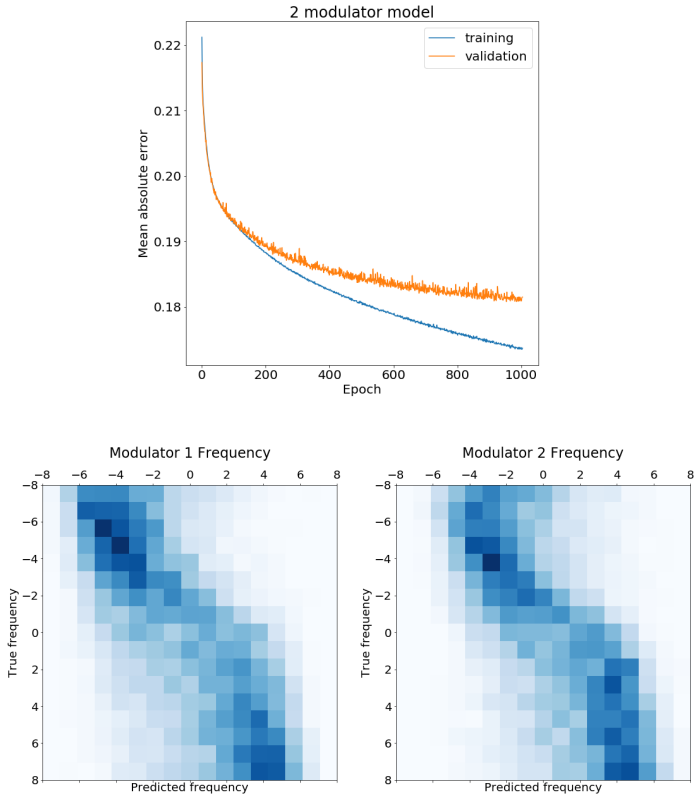

Figure 5.5: Confusion matrix of frequency prediction by the 2 modulators model on the validation set. It can be seen that the model is better at detecting the pitch of a modulator in the lower frequencies, whereas less accurate in the higher frequencies.

3 modulators

In figure 5.6 the training loss and confusion matrix of the 3-modulator model is shown. The model does not seem to overfit. From the validation error at 0.206 and the even more blurry confusion matrices it can be seen that the model is significantly less accurate than the 1-modulator and 2-modulators model. The model is not accurate at detecting modulators with either low or high frequencies, and the lion share of the predictions are between frequencie value -4 and 4.

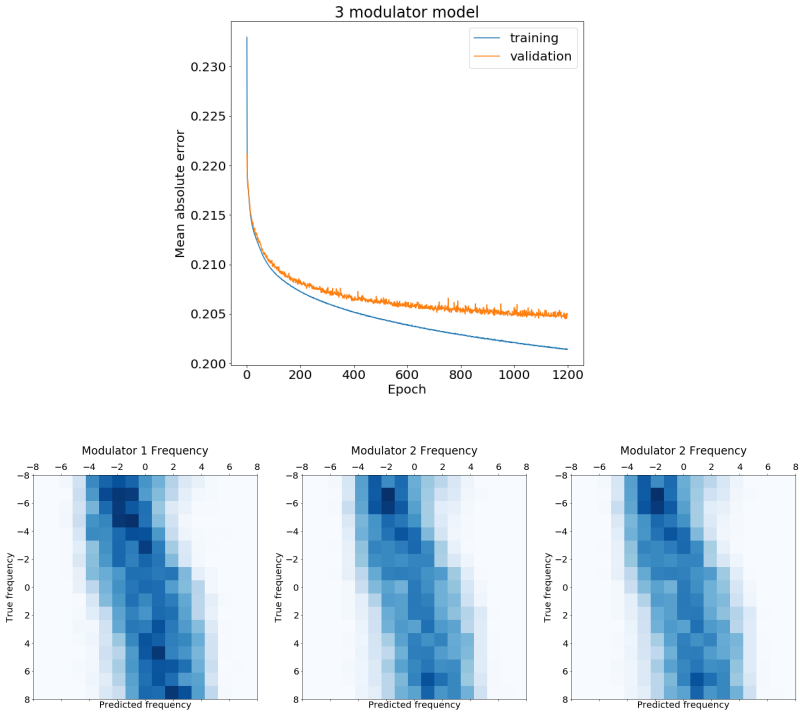

Figure 5.6: Above, training and validation loss of the 3 modulator model. Below, the confusion matrix of frequency prediction by the 3 modulators model on the validation set. It can be seen that the model is better at detecting the pitch of a modulator in the lower frequencies, whereas less accurate in the higher frequencies.

Envelope and cutoff

In figure 5.7 we can see the loss and confusion matrices of the envelope and cutoff model. The predicted values have been rounded to the closes decimal, however they are not rounded when fed to the synthesizer. It can bee seen that the model has captured the relationship between sound features and parameters to some extent, however it is far from perfect.

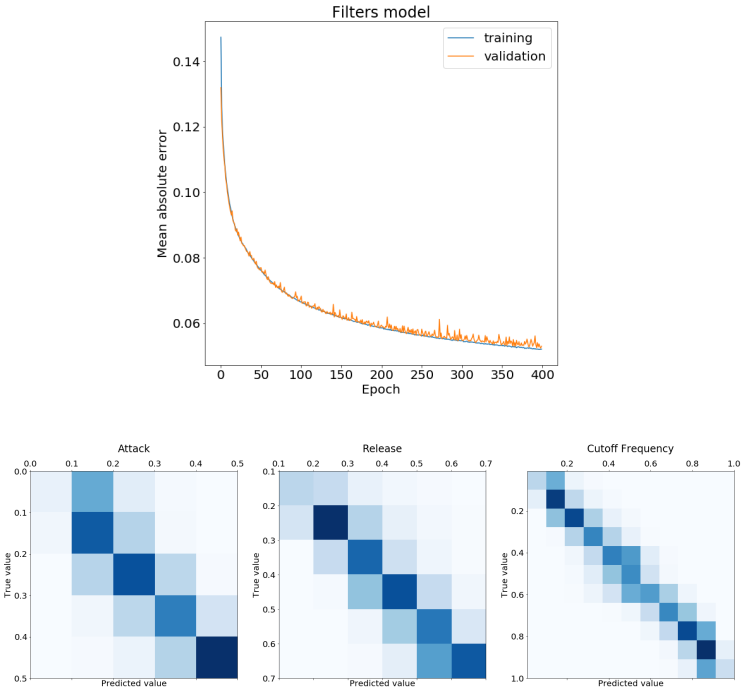

Figure 5.7: On top the training and validation error of the envelope and cutoff model. Below, the confusion matrices of the attack, release and cutoff.

5.2 Evaluation

In this subsection we describe the evaluation and performance of the two approaches. First the overall performance of the approaches are listed. Second, their ability to reconstruct frequency spectras are quantified. Third, their ability to reconstruct the temporal frequency spectras are quantified. Fourth, their ability to reconstruct the amplitude envelope are quantified. Finally, the observations are discussed.

5.2.1 Overall Performance

We compute the distance of each target and candidate sound for both of the approaches with respect to the four distance metrics.

	approach	d_FFT	d_STFT	d_LMS	d_envelope
min	**concurrent**	1.00	45.23	807.84	0.04
	sequential	1.25	27.03	422.40	0.05
mean	**sequential**	16.27	354.18	4602.14	0.42
	concurrent	16.99	497.66	4896.58	0.48
std	**concurrent**	23.51	318.57	1866.34	0.23
	sequential	23.82	281.00	2560.69	0.21
max	**concurrent**	515.33	6507.30	10834.02	0.91
	sequential	518.93	6450.60	11381.52	0.76

Table 5.1: Performance of the concurrent and sequential approaches over the whole evaluation dataset.

We conduct a two-sample z-test to assert whether the distributions for both approaches are in fact different. We conduct the z-test with 99.9% confidence, and for each metric test the null hypothesis that *there is no difference between the performance of the concurrent and sequential for the given metric.* In table 5.2 the results of the z-tests are displayed, indicating the it is possible to reject the null hypothesis in the case of STFT, LMS and envelope distance, but not in terms of FFT distance.

	d_FFT	d_STFT	d_LMS	d_envelope
p-value	3.05e-01	6.71e-58	1.02e-05	1.36e-18
Null hypothesis	not rejected	rejected	rejected	rejected

Table 5.2: Results from a z-test for each of the distance metrics of the null hypothesis that the distribution from both of the approaches for the given distance metric are the same. The test is conducted over all 2255 re-synthesized samples with 99.9% confidence. The null-hypothesis can be rejected for the STFT, LMS and envelope distances, but not for the FFT-distance.

5.2.2 Reconstructing the frequency spectrum

In fig. 5.8 the target sounds are grouped by Spectral Entropy and Spectral Flatness. The plots appear to show that the models yield near exactly the same results with respect to FFT-distance. This goes in line with the z-test, which could not reject the null hypothesis for the FFT-distance metric.

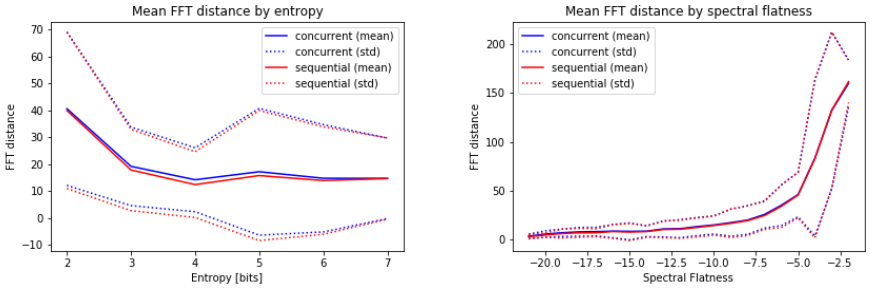

Figure 5.8: FFT-distance of target sounds grouped by Spectral Entropy and Spectral Flatness respectively. In the left plot, entropy ranges from low entropy (low amount of information) to high (high amount of information). To the right, the spectral flatness range from minus infinity to 0, with clean sinusoids being low on the scale and and flat frequencies spectrums (e.g. white noise) being close to zero. The mean and standard deviations of the FFT-distance of the concurrent and sequential models are plotted We can see that both models perform almost exactly similarly well in both cases.

5.2.3 Reconstructing the temporal frequency spectrums

As shown in table 5.1 we know that the sequential model outperforms the concurrent at reconstructing the temporal envolution of the frequency components. In fig. 5.9 the target sounds are grouped by Spectral Entropy and Spectral Flatness, and the STFT-distance and LMS-distances are plotted as functions of these attributes. We can see that the sequential model appear to perform better for most spectral entropy and flatness values. The relative difference is significantly smaller with respect to LMS-distance.

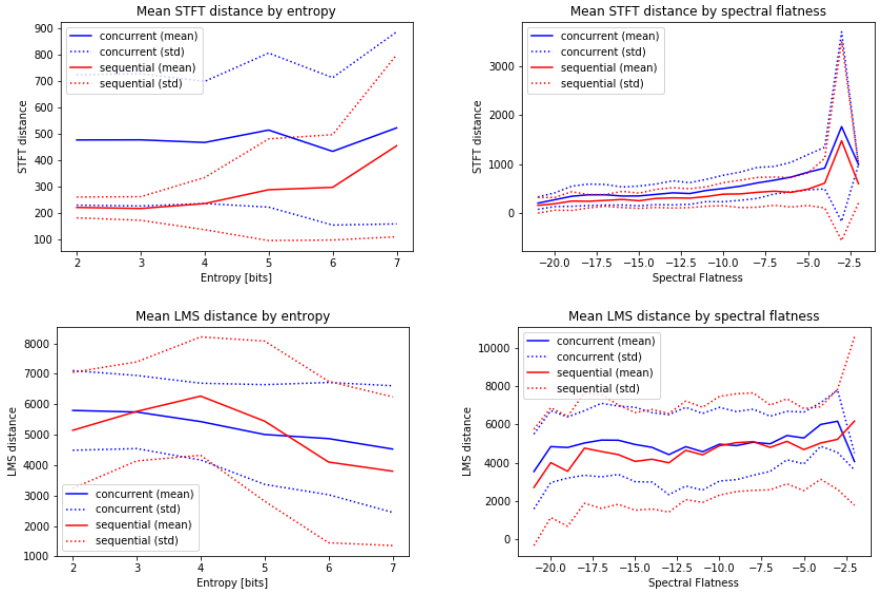

Figure 5.9: STFT-distance and LMS-distance between target and candidate sounds by the target sounds' Spectral Entropy and Spectral Flatness of each model.

5.2.4 Reconstructing the amplitude envelope

As shown in table 5.1 we know that the sequential model outperforms the concurrent at reconstructing the envelope. In fig. 5.10 the target sounds are grouped by Spectral Entropy and Spectral Flatness, and the envelope-distance is plotted as functions of these attributes. We can see that the sequential model appear to perform consistently better for all entropy and flatness values. The performance increases on sounds of high entropy but decreses on sounds of high spectral flatness.

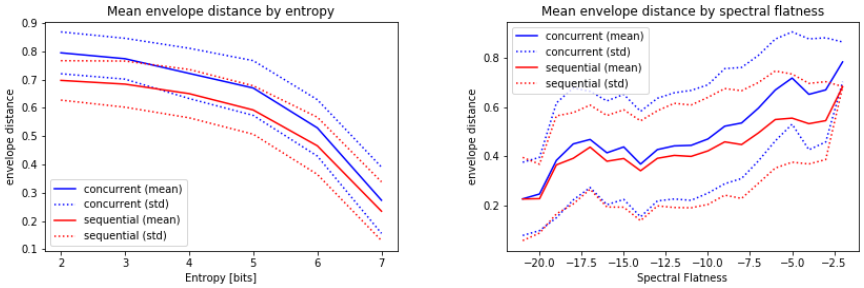

Figure 5.10: Envelope-distance between target and candidate sounds by the target sounds' Spectral Entropy and Spectral Flatness od each model. We can see that...

5.2.5 Performance by instrument family

In figure 5.11 the performance of both approaches are grouped by each sound's instrument family as labelled in the Nsynth dataset. We can see that the concurrent model performs consistenly better for all instruments with regards to the STFT-distance: all means are lower and all variances are generally lower.

However, with respect to LMS-distance the results are more mixed. The sequential model performs better for instruments like bass, brass, flute, organ and vocal while but generally has a high variance.

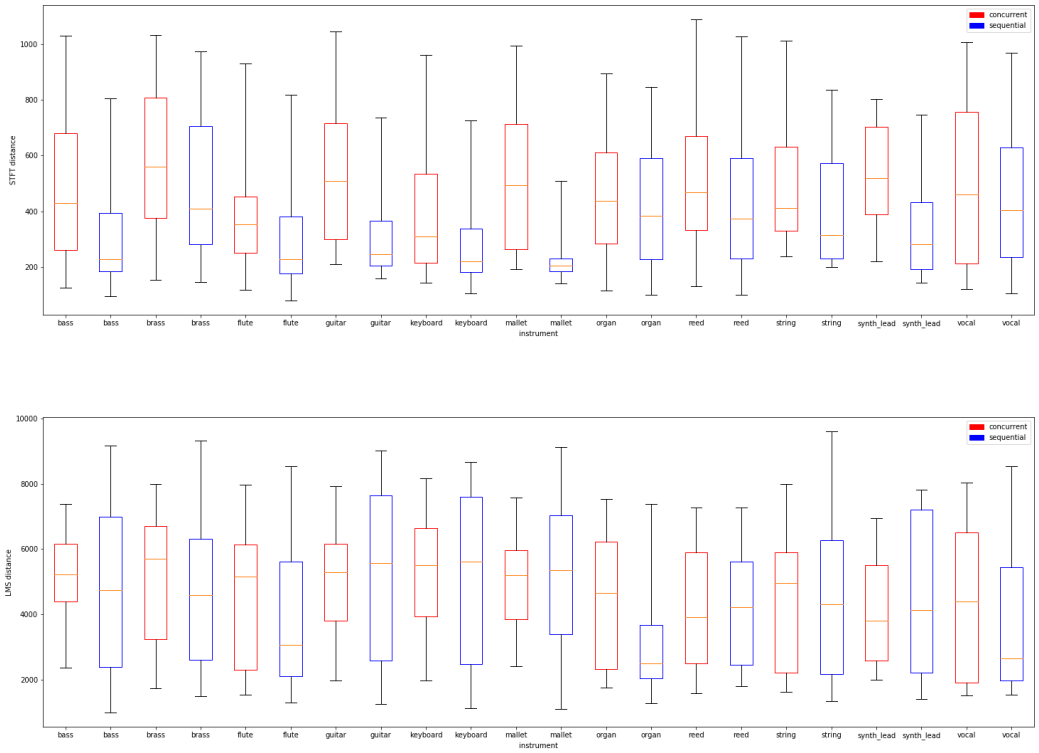

Figure 5.11: STFT and LMS-distance between target and candidate sounds, with target sounds grouped by pitch. The boxes represent quartile Q1-Q3 and the whiskers represent the 5%-quantiles. Outliers are ignored for readability.

5.2.6 Re-synthesized sound spectrograms

In fig. 5.12 the spectrograms of two re-synthesized are shown as example. More examples are shown in Appendix A.

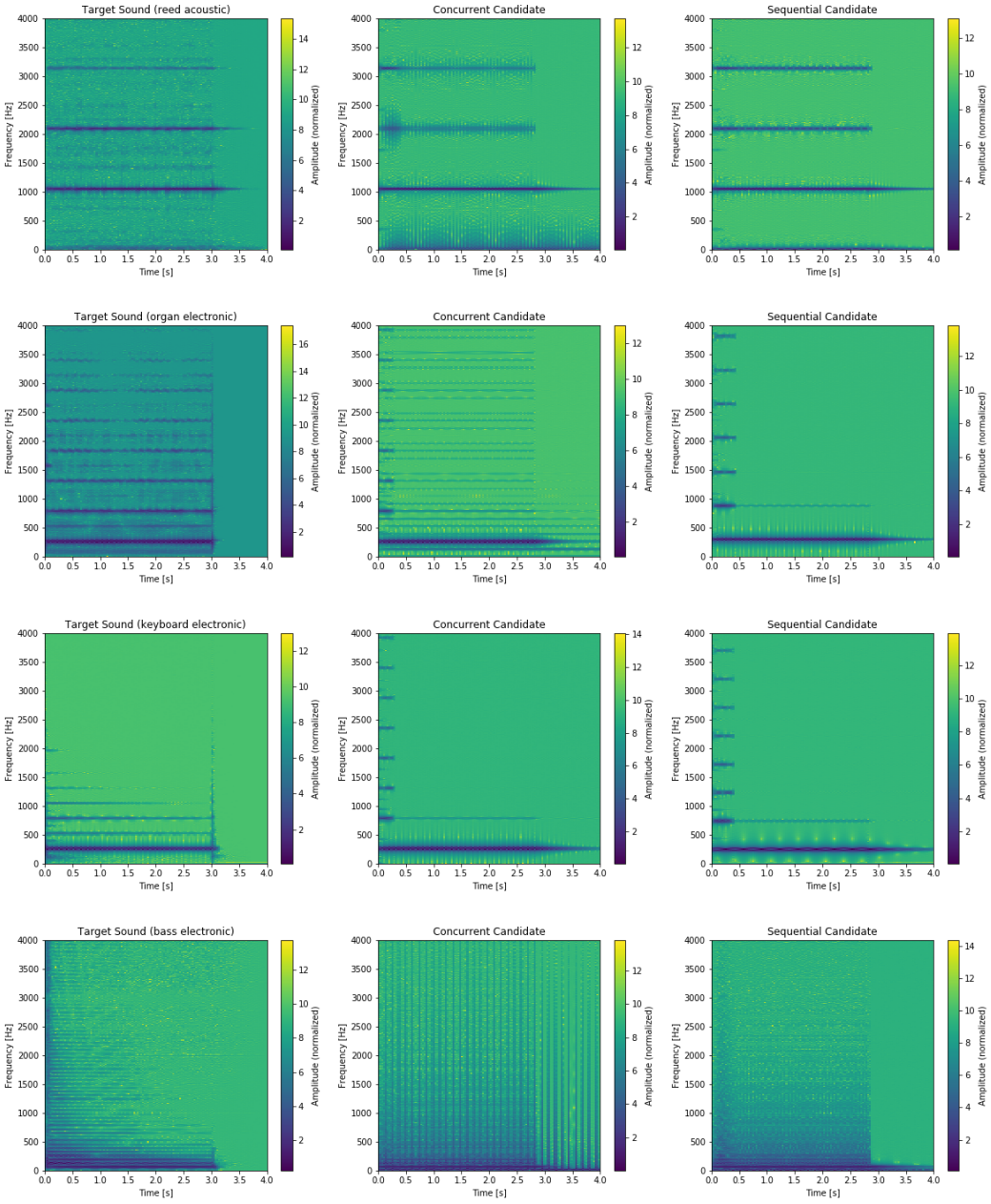

Figure 5.12: The log scaled STFT spectrograms of a number of re-synthesized sounds

5.3 Discussion

The results from the z-test in table 5.2 as well as the performance shown in table 5.1 suggest that the performance of both approaches with respect to the frequency magnitude spectrum are very similar, since the min, mean, max and standard deviation of the FFT-distance are very similar. Similarly, both graphs in figure 5.8 in subsection 5.2.2, show that the two models yield almost exactly the same results with respect to the FFT-distance. We can make the assumption that the two models have similar capabilities to choose the correct parameters to reproduce both simple and complex (flat) frequency spectrums. Likely, this is related to the ability to choose the frequency-related parameters: number of modulators, modulating frequency and amplitude, and also the cutoff filter. This is highly interesting since it indicates that the explicit design of a classifier which decided how many modulators to use likely did not improve the performance directly. This indicates that the large, concurrent model was indeed implicitly capable of choosing FM-related parameter values with the same precision as the explicitly design sequential model.

Furthermore, the figure with data grouped by Spectral Flatness shows that both models show a tendency at performing worse at reconstructing flat frequency spectras. This is expected, since noisy frequency spectras are hard or impossible to produce with just sinusoidal frequency modulation.

Table 5.1 as well as figure 5.9 in subsection 5.2.3 suggests that the sequential approach performs slightly better than the concurrent approach at resynthesizing sounds with respect to the temporal evolution of the frequency components. The only exception is with respect to the LMS-distance for sounds with 3-5 bits spectral entropy, and it is hard to reason why this would be the case, and this could be due to noise. Since the different approaches had similar succes at creating the frequency spectras of sound, but the sequential approach was better at mimicing the frequency spectra over time we can conclude that the sequential approach is better at predicting the envelope parameters of the oscillators. Possibly, the smaller sub-models of the sequential model benefited from small size when predicting the rather linear envelope parameters, whereas the large model may have overfitted to noise due to the many parameters.

Table 5.1 as well as figure 5.10 in subsection 5.2.4 indicate that the sequential model is slightly better at predicting the envelope. This result goes along the same lines as the above mentioned - the sequential model is better at predicting the envelope parameters.

Furthermore, the mean envelope distance is smaller for sounds with high

entropy for both models. Since sounds with high entropy generally have envelopes with high mean amplitudes, this could mean that the models are biased towards creating envelopes with high mean amplitudes. Furthermore, both models performance decreases as the spectral flatness increases. This could be due to noisy sounds having noisy envelopes, which could be difficult to model.

Chapter 6

Conclusions and Future work

In this thesis two approaches to machine learning based sound re-synthesis has been implemented, examined and evaluated. In this section, the results of the thesis are concluded and some suggestions for future work are outlined.

6.1 Conclusions

We can conclude that both approaches appear to perform similarly at the task of re-synthesis with respect to FFT-distance. That is, both models have the same capabilities at reproducing frequency spectrums. This points at the models performing similarly well at choosing the parameters parameters which control the frequency magnitude spectrum, such as the amplitude and frequency of the modulating oscillators. We can conclude that the concurrent approach when trained on a balanced data set is still general enough to implicitly decide how many modulating oscillators to use - this indicates there *may be no need to explicitly design* a system of classifiers and regressions to make key decisions with regards the frequency modulation.

However, the results in section 5 also indicate that the sequential approach seems to be better at reconstructing the temporal evolution of the frequency spectrum. This indicates the sequential approach might be able to choose the envelope parameters *attack* and *release* of each oscillator more accurately, which affects the modulation and subsequently the frequency spectrum over time.

This result indicates that the statement by Horner et al. [10] - that *"the decomposition of the matching process into subproblems is central to success"* - may no longer be valid. The new capabilities of machine learning and deep learning may be strong enough to implicitly learn to decompose these pro-

cesses into subproblems, potentially eliminating the need of human expertice or risk of human bias.

To conclude and clearly answer the research question stated in 1.3, the *sequential* approach in this thesis perform overall better than the *concurrent*. However, the results also suggests that the concurrent *did* manage to make FM-related key decisions as well as the sequential, however did not manage to do so with the envelope parameters.

6.2 Limitations

As stated in the delimitations section 1.7, some parameters were excluded to reduce complexity. However, the addition of more parameters could benefit one or the other approaches differently. For example more categorical parameters such as the choice of FM-algorithm could benefit the sequential approach with its explicitly designed classifiers. Furthermore, the results could be specific for an FM-synthesizer and this implementation in particular. We can not rule out that the results would have been different in another setting. Furthermore, it is hard to control whether some bias has been introduced in the time and effort spent optimizing the two different approaches. Training a Machine Learning model is in part a manual search for optimal hyperparameters. This search is greedy and may result in finding more optimized hyperparameters for one of the approaches, hence inducing a bias in the result.

Also, choosing appropriate metrics of sound similarity is of significance in order to train the system of models which yield perceptually similar sounds. However, the mix of which similarity metrics model human perception can be domain specific and potentially subjective. In this thesis, this was not fully accounted for and different metrics, such as comparing the Mel-Frequency Cepstrum Coefficients, could have yielded other results.

Finally, it is worth noting that with more time and better data infrastructure the approaches tested could have been deeper, more specifically using Convolutional Neural Networks (CNN) doing the feature extraction implicitly. It is hard to speculate in whether the results of this thesis would generalize to even deeper structures and algorithms - it would be of interest to attempt the same experiment but with deeper models.

6.3 Future Work

As described in the above subsection, first future work to perform would be to redo the same experiment but with deeper models like CNNs. CNNs are well known for performing well in high dimensional decision problems, and would likely improve performance significantly.

Futhermore, it would be intersting to redo the experiments with more parameters. In particular, the introduction of more class variables like the choice of modulation algorithm would put high demands on the implicit capabilities of the concurrent approach.

For re-synthesis in general there is a lot of interesting work to be done. Especially, optimizing a model with respect to the sound similarity between target and candidate rather than the similarity of the parameter values, as discussed in section 1.2. This cold be done by *training* a model with i.e. GA, an intersting approach since GA has already shown good performance for this domain. Stanley's work with deep neuroevolution [21] would be interesting to apply to the re-synthesis domain. Another approach, relating to the generative neureal re-synthesis work by Engel et al. [7] would be to explicitly implement a synthesizer in a machine learning optmimized and transparent manner (i.e. tensorflow). The synthesizer would then not be a black box, and it would be possible to optimize with respect to the sound similarity rather than the similarity of the parameter values, as discussed in section 1.2.

Bibliography

[1] MIDI Manufacturers Association et al. “The complete MIDI 1.0 detailed specification”. In: *Los Angeles, CA, The MIDI Manufacturers Association* (1996).

[2] Oren Barkan and David Tsiris. “Deep Synthesizer Parameter Estimation”. In: *CoRR* abs/1812.06349 (2018).

[3] John M Chowning. “The synthesis of complex audio spectra by means of frequency modulation”. In: *Journal of the audio engineering society* 21.7 (1973), pp. 526–534.

[4] James Cooley and John Tukey. “An Algorithm for the Machine Calculation of Complex Fourier Series”. In: *Mathematics of Computation* 19.90 (1965), pp. 297–301.

[5] Steven Davis and Paul Mermelstein. “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences”. In: *IEEE transactions on acoustics, speech, and signal processing* 28.4 (1980), pp. 357–366.

[6] S. Dubnov. “Generalization of spectral flatness measure for non-Gaussian linear processes”. In: *IEEE Signal Processing Letters* 11.8 (Aug. 2004), pp. 698–701. ISSN: 1070-9908. DOI: 10.1109/LSP.2004.831663.

[7] Jesse Engel et al. *Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders*. 2017. eprint: arXiv:1704.01279.

[8] Anne Håkansson. “Portal of research methods and methodologies for research projects and degree projects”. In: *The 2013 World Congress in Computer Science, Computer Engineering, and Applied Computing WORLDCOMP 2013; Las Vegas, Nevada, USA, 22-25 July*. CSREA Press USA. 2013, pp. 67–73.

[9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[10] Andrew Horner, James Beauchamp, and Lippold Haken. “Machine Tongues XVI: Genetic Algorithms and Their Application to FM Matching Synthesis”. In: *Computer Music Journal* 17.4 (1993), pp. 17–29. ISSN: 01489267, 15315169. URL: http://www.jstor.org/stable/3680541.

[11] Yedid Hoshen, Ron J Weiss, and Kevin W Wilson. “Speech acoustic modeling from raw multichannel waveforms”. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2015, pp. 4624–4628.

[12] Yuyo Lai et al. *AUTOMATED OPTIMIZATION OF PARAMETERS FOR FM SOUND SYNTHESIS WITH GENETIC ALGORITHMS*.

[13] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[14] Tom M. Mitchell. *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997. ISBN: 0071154671. URL: http://www.worldcat.org/oclc/61321007.

[15] Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. “Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques”. In: *arXiv preprint arXiv:1003.4083* (2010).

[16] Hendrik Purwins et al. “Deep Learning for Audio Signal Processing”. In: *IEEE Journal of Selected Topics in Signal Processing* (2019).

[17] Jean-Claude Risset. “Analysis of musical instrument tones”. In: *Phys. Today* 22 (1969), pp. 23–30.

[18] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.

[19] David Silver et al. “Mastering the game of Go without human knowledge”. In: *Nature* 550.7676 (2017). ISSN: 0028-0836.

[20] S. S. Stevens, J. Volkmann, and E. B. Newman. “A Scale for the Measurement of the Psychological Magnitude Pitch”. In: *The Journal of the Acoustical Society of America* 8.3 (1937), pp. 185–190. DOI: 10.1121/1.1915893. eprint: https://doi.org/10.1121/1.1915893. URL: https://doi.org/10.1121/1.1915893.

[21] Felipe Petroski Such et al. “Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning”. In: *arXiv preprint arXiv:1712.06567* (2017).

[22] Kıvanç Tatar, Matthieu Macret, and Philippe Pasquier. “Automatic Synthesizer Preset Generation with PresetGen”. In: *Journal of New Music Research* 45.2 (2016), pp. 124–144. DOI: 10.1080/09298215.2016.1175481. eprint: https://doi.org/10.1080/09298215.2016.1175481. URL: https://doi.org/10.1080/09298215.2016.1175481.

[23] Zoltán Tüske et al. “Acoustic modeling with deep neural networks using raw time signal for LVCSR”. In: *Fifteenth annual conference of the international speech communication association*. 2014.

[24] Mark Vail. *The synthesizer : a comprehensive guide to understanding, programming, playing, and recording the ultimate electronic music instrument*. New York : Oxford University Press, [2014]. URL: https://search.library.wisc.edu/catalog/9910194327002121.

[25] Aäron Van Den Oord et al. “WaveNet: A generative model for raw audio.” In: *SSW* 125 (2016).

[26] Mei Wang and Weihong Deng. “Deep Face Recognition: A Survey”. In: (2018).

[27] Joe Wolfe. *Note names, MIDI numbers and frequencies*. https://newt.phys.unsw.edu.au/jw/notes.html. Accessed: 2019-05-04.

[28] Matthew Yee-king and Martin Roth. “Synthbot: An unsupervised software synthesizer programmer”. In: *In Proc. International Computer Music Conference (ICMC-08*. 2008, pp. 184–187.

[29] Steven Young et al. “Optimizing deep learning hyper-parameters through an evolutionary algorithm”. eng. In: *Proceedings of the Workshop on machine learning in high-performance computing environments*. MLHPC ’15. ACM, 2015, pp. 1–5. ISBN: 9781450340069.