

https://dspace.org/

Automatic mixing of musical compositions using machine learning

Blanckensee, Darren

2020
https://hdl.handle.net/10539/32743

Downloaded from a DSpace repository. https://dspace.org/



UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

Automatic mixing of musical compositions using machine learning

---

A dissertation submitted to the school of Computer Science and Applied Mathematics, University of the Witwatersrand, in fulfilment of the requirements for the degree of Master of Science

Author: Darren Blanckensee

Supervisor: Professor Byron Jacobs

Johannesburg, 2020

Declaration

I, **Darren Blanckensee**, declare that this dissertation is my own, unaided work. It is being submitted for the Degree of Master of Science at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other University.

Student: ..... Date.....25/03/2021.....

Supervisor: ..... Date.....25/03/2021.....

Acknowledgment

I would like to thank my supervisor, Prof. Byron Jacobs for all the help he gave me at every step of the way. From sourcing the dataset used and suggesting online courses that proved to be extremely helpful to the feedback on the dissertation, and the help going back and forth with ideas and suggestions. Without him there would be no dissertation. I would also like to thank my family for helping me push through, keeping me focused and motivated, and putting up with all the late nights spent working. I would also like to thank the external moderators, who, through their feedback, helped me make important changes that resulted in a much stronger piece of work. Finally, I would like to extend my gratitude to the National Research Foundation (NRF) for their financial help, without which I would not have had the equipment and software I used to complete this dissertation.

Dedication

I dedicate this project to all the musicians out there who are following their dreams, even when there is so much in the way telling them they will never make it. And especially to those who also have to fight against prejudice just to get a foot in the door. Music is my life. Spending a year working on this has been a blessing to me because I have learnt so much about what I love and have in the process opened so many doors to new paths that I look forward to exploring. Hopefully this is the beginning of a journey that will allow me to open these doors for others who need them opened.

Abstract

Given the increasing availability of music production software and the many live musical performances that take place without the assistance of professional sound engineers, the need has arisen for automatic mixing software capable of producing high quality mixes.

The mixing of songs is the process of going from individually recorded tracks to a single song. This process involves the use of various effects which are essentially functions applied to the individual tracks in a particular order. After the effects have been applied to each of the tracks, the results are then added together to form the final mix of the song. This project uses machine learning to automate the mixing process. Although the mixing process generally involves multiple effects, this project considers only equalisation and volume control. Two systems are designed, trained and tested, one for equalisation and the other for volume control. Each system tests various architectures to see which performs the best. The performances of the equalisation and volume control systems are evaluated individually as well as together using a data set of professionally mixed songs. The outputs of the neural networks are compared with the professionally mixed songs from the dataset using spectrograms and waveforms of the final mixes produced by each.

It is shown that by using well designed neural networks trained to a sufficient level, the quality of the mixes produced by the neural networks approaches that of the professionally mixed songs.

Contents

1 Introduction **1**

1.1 Background 1

1.2 Audio Mixing 3

1.2.1 Compression 4

1.2.2 Equalisation 7

1.2.3 Volume Control 10

1.2.4 Reverberation 11

1.2.5 Panoramic Positioning 11

1.3 Project Goal 12

1.4 Training Technique 14

1.5 Evaluation Criteria 15

1.5.1 The Short Time Fourier Transform 15

1.6 Considerations 18

2 Literature Review **21**

2.1 Machine Learning 21

2.2 The History of Automatic Mixing 24

2.3 Automatic Mixing Guidelines 26

2.4 Cross-Adaptive Systems 28

2.5 Machine Learning in Automatic Mixing 29

2.6 Spectrograms as Evaluation Criteria 33
2.7 Machine Learning in System Identification 34
2.8 Machine Learning in Music Classification 36

3 Problem Statement and Objectives **37**
3.1 Problem Statement 37
3.2 Objectives 37

4 Dataset **41**
4.1 Formatting and Normalisation 42
4.1.1 Equaliser Input-Output Pair Formatting 50
4.1.2 Volume Control Input-Output Pair Formatting 51

5 Methodology **56**
5.1 Work Environment 57
5.1.1 Loss Functions 58
5.2 Implementation 64
5.2.1 Equalisation 65
5.2.2 Volume and Gain Control 73

6 Results **81**
6.1 Equaliser 82
6.2 Spectrogram Comparison 85
6.2.1 32 Band Multi-layered Densely Connected Architecture . 87
6.2.2 32 Band Custom Musically Informed Architecture 89
6.2.3 32 Band Custom Musically Informed Architecture with Subsequent Dense Layers 91
6.2.4 4096 Band Multi-layered Densely Connected Architecture 92
6.3 EQ visualisation 93

6.4 Volume Control 96
6.5 Combined System Results 99

7 Recommendations for Future Work **102**

8 Conclusion **104**

8.1 Applications 106
8.1.1 Open-Source Professional Mixing 106
8.1.2 Automatic Live Mixing 106

Appendices **116**

A Dataset Sounds **117**

Chapter 1

Introduction

1.1 Background

The increasing availability of music production software has led to a surge in the amount of music being created daily. In the past, music production took place in professional studios where the recording process was closely monitored and facilitated by professional sound/recording engineers. The songs produced in these studios went on to be mixed and mastered by professional mix engineers with expert knowledge in signal processing. Today, music is being produced in various settings ranging from professional studios to makeshift bedroom studios with USB microphones and laptops with digital audio workstation software. While this has led to enormous growth in the music industry, it also means that a large amount of the music that people are exposed to is not professionally mixed. This is because mixing a song in a way that ensures it is of a professional standard is relatively complex as it requires a wealth of knowledge about signal processing which is not common in the majority of music producers.

In the past, most professional mix engineers had been educated on the subject in some form or another. Tony Maserati, who has been credited on songs by artists ranging from James Brown, David Bowie and Whitney Houston to 2 Chainz, Ed Sheeran and Beyonce between 1987 and 2018, had a Masters in Arts from Berklee College of Music in Boston [Shea, 2014]. Bruce Swedien, another well-known mix engineer credited for working with jazz legends like Duke Ellington and Oscar Peterson, pop icons like Michael Jackson (Thriller) and Barbara Streisand, as well as crooners like Nat King Cole, had a degree from the University of Minnesota, where he majored in electrical engineering and minored in music [BRU, 2020]. Susan Rogers, a professor of music production and engineering at Berklee College of Music is credited as having mixed songs for legendary artists like Prince (Purple Rain) and Queen [SUS, 2020].

In the past, being a mix/recording engineer required both technical and musical knowledge in order to not only capture but produce the best quality sound from a given recording using the available equipment. Today, thanks to advances in technology and the abstraction of the music creation process, many musicians are choosing to make music in home studios. These musicians often choose to mix and master this music themselves, as hiring a professional mix engineer is often very expensive. In addition, digital audio workstations provide musicians with the tools they need to be able to mix and master their own music. This is good in that it means that the world is exposed to more music than ever before thanks to these technical advances. The only downside is that professional standard mixing requires technical knowledge that is not easy to obtain.

If it were possible for anyone to use an automatic mixing system that could

do the mixing for these musicians, it would mean that the quality of the music from a purely technical perspective would increase significantly. This would make for a better listening experience for the listeners. Everyone has listened to a song and had the snare drum make them blink because of how loud it was, or had their headphones distort when the kick drum and bass guitar are struck at the same time. Songs like this would clearly benefit from such a system. It is the goal of this project to construct a system like this and analyse its results to see if it is possible to train neural networks to learn to mix musical compositions across multiple genres containing multiple combinations of sounds and instruments.

The goal is not only to train the neural networks but also to compare them to the work of professional mix engineers similar to the people mentioned above. In the process of building this automatic mixing system, as much information as possible will be gathered about what the neural network learns on its own without explicitly being taught by assessing several useful metrics and deeply analysing the frequency content of the output mixes produced by the system.

1.2 Audio Mixing

Mixing a song refers to the process of going from multiple individual, separately recorded pieces of music (main vocal, backup vocals, drums, guitar, piano, synthesiser, etc.) to one final piece of music which contains all the individual elements, and whatever effects the mix engineer has chosen to use on them. These individual elements of a musical piece will be referred to as the individual tracks or stems going forward.

Mixing is different to mastering in that mixing is done on each individual track whereas mastering is done on the sum of all the mixed tracks. Given that the mix engineer has access to all the individual tracks, they have a significant amount of control over how they would like each of the tracks to sit in the mix and what tonal and spacial qualities they would like each track to have.

In order to carry out the vision for the song, the mix engineer takes into consideration a combination of suggestions from the artist and producer as well as their own experience and knowledge of best practices. There are many techniques that fall under the umbrella of audio mixing. Techniques such as compression, equalisation, gain control and panoramic positioning are all part of the process and are used to achieve particular goals.

1.2.1 Compression

In [Vickers, 2010] the 'loudness war' is discussed. The loudness war describes a period of time when it was thought that the louder a song was, the more it would stand out among others on radio and therefore the more it would sell. The loudness wars started in the late 1970's and early 1980's when there was a belief that louder signals on the radio were more attractive to listeners. The loudness war still continues today and is the reason why some songs are perceived as being significantly louder than others when listening at the same volume.

There are a number of methods a mix engineer can use to make a song as loud as possible. It is important, however, to ensure that in doing so, the final mix

does not clip. Clipping is what happens when a sound is too loud for whatever system is being used to reproduce it accurately. In most digital audio workstations today, there is a level at which any sound louder than that level cannot be reproduced accurately and would cause distortion. This reference level is assigned the level 0dB. When mixing it is important to ensure that the final mix never exceeds this 0dB reference level. This provides mix engineers participating in the loudness wars with a challenge. How does one increase the perceived loudness of a song as much as possible without causing clipping? That is where compression comes in.

Compression is a way of increasing the perceived loudness of a track without increasing its peak volume. This is done by reducing what is called the dynamic range of the track. The dynamic range of a track refers to the ratio between the softest and loudest parts of a track, and is measured in dB. For a song that commences with a soft solo violin but ends with a loud full orchestra playing together, the dynamic range would be significantly more than a song that only has a soft solo violin for the entire duration of the song. Reducing the dynamic range of a song means decreasing the difference in volume between the softest and loudest parts of a song. This often results in the softer parts of the song becoming much louder and the peaks (loudest parts) being somewhat reduced depending on the nature of the compression.

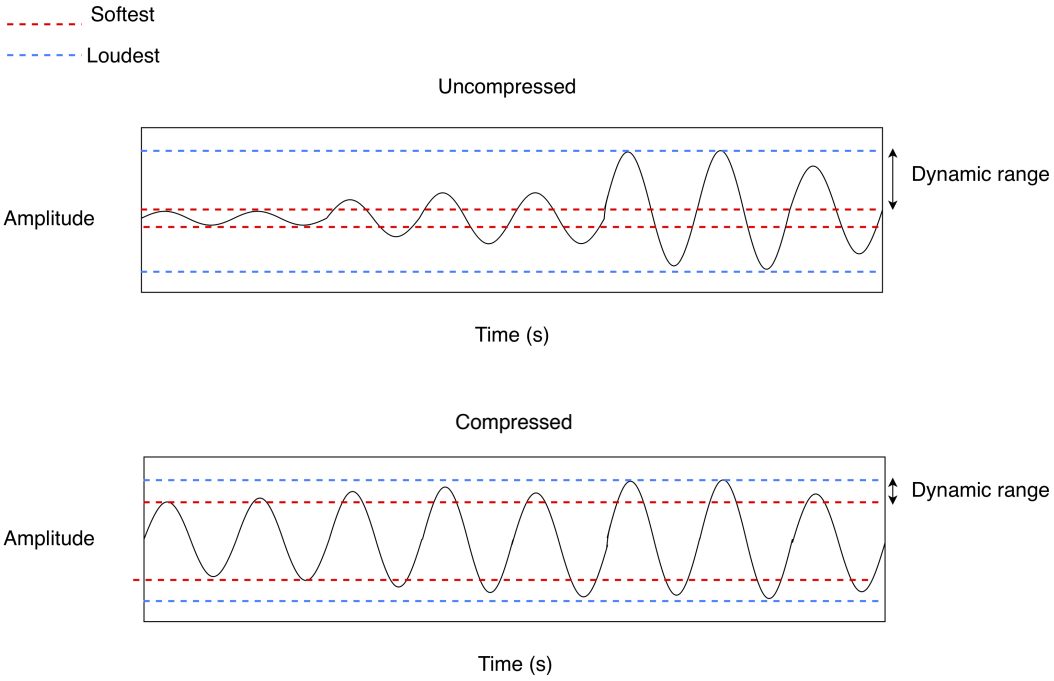


Figure 1.1: Trivial example of the effect of compression on dynamic range

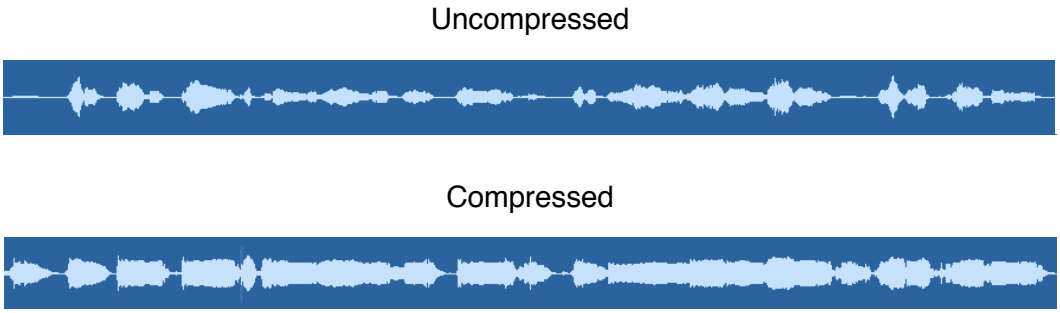


Figure 1.2: Example of compression on an actual vocal signal

As Figure 1.1 above shows, the overall volume of the song has increased but the peak volume has stayed the same. In doing this, loudness has been increased while avoiding clipping. This is what compression does, and as can be seen, this results in a reduction of the dynamic range. Figure 1.2 shows the effect that compression has on an actual vocal signal.

The loudness war has been criticised heavily by many over the years. While loudness does make a song stand out among other songs as desired, a number of

negative side effects are also associated with the loudness war. Some of these include listener fatigue and increasing what is referred to as the noise floor [Campbell et al., 2010]. This essentially increases the amount of unwanted noise in the final mix. Compression is a valuable tool when used for good reason and is still one of the effects that is often used when mixing music.

1.2.2 Equalisation

Equalisation refers to any manipulation of a signal's frequency response, most often using filters. The origin of equalisation lies in the attempt to reduce the high frequency losses experienced in telephone technology. When transmitted over long distances, the spectrum of the signal at the receiver did not match the signal at the transmitter because of the high frequency losses. To counteract this, the higher frequencies were amplified at the transmitter in order to cater for the expected losses and produce a matching spectrum at the receiver [Välimäki and Reiss, 2016].

The initial goal of equalisers was to accurately reproduce signals and conserve the frequency content. The first variable equaliser (equaliser where the boosts or cuts in frequency could be varied by the user) was used in 1920 for the equalisation of a motion picture [Välimäki and Reiss, 2016]. This began the journey into variable equalisation that has led to where we are today. In the music industry today equalisers are used for much more than reproducing sounds accurately. These days, equalisers are used both to solve technical issues and artistically to add another layer of creativity to a song.

Equalisers effectively provide mix engineers with the ability to increase (boost)

or reduce (cut) the presence of certain frequency bands in a particular sound. They can be used in many ways and for many different reasons. There is an on-going debate in the mixing world about whether engineers should prefer to boost or cut when designing the equaliser. Equalisers typically make use of combinations of various types of filters that, when used simultaneously, alter the frequency response of a signal in a particular way. This allows the mix engineer to control the frequency content of each sound in the mix and in so doing create an output mix that has been carefully constructed to amplify certain desired frequencies and to eliminate (to a certain extent) the unwanted frequencies on a stem by stem basis.

This was originally done by adjusting sliders that either boost or cut certain frequency bands for a particular sound. This led to what most mix engineers are familiar with today which is the graphic equaliser. An example how a graphic equaliser is used is shown in Figure 1.3 below.

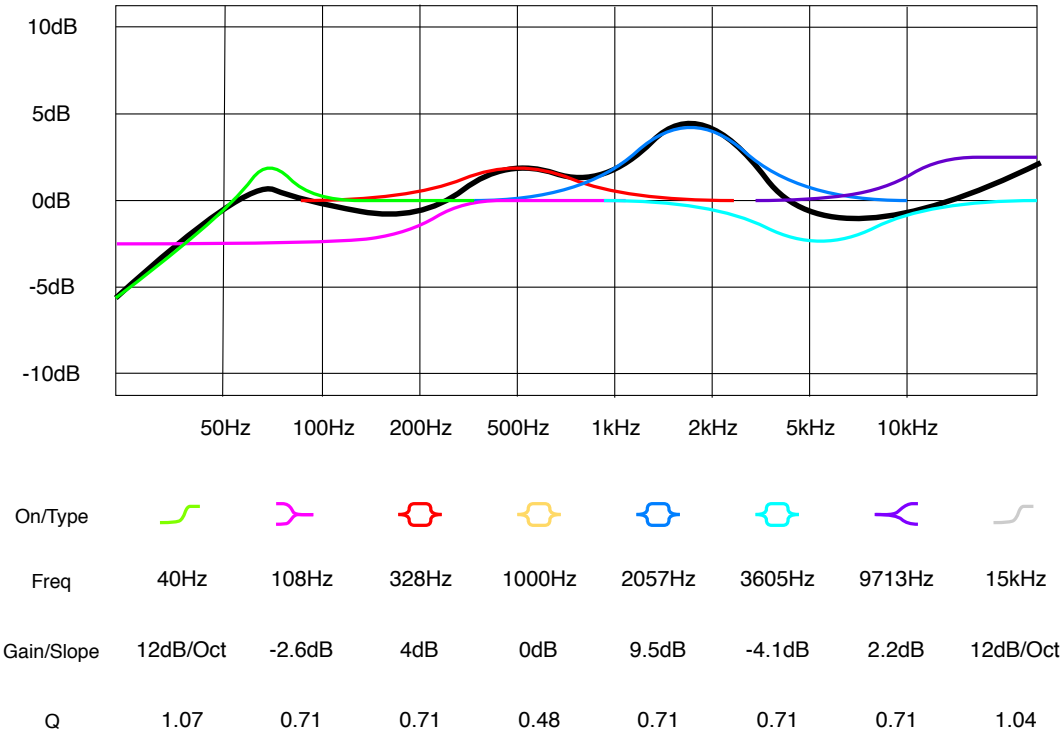


Figure 1.3: Example of a graphic equaliser

With graphic equalisers, mix engineers are able to adjust the gain of certain filters up or down. This relates to boosting or cutting the presence of the frequencies affected by each filter in the mix. For example, the green line on the equaliser is a filter that lets frequencies above 50Hz through and attenuates anything less than 50Hz with a slope of 12dB per octave. The red line is a filter that boosts frequencies between 200Hz and 2KHz and does not attenuate at all. These graphic equalisers have many variable parameters. Anything can be changed, from the bandwidth (Q/resonance) to the gain and even the center frequency of the filters. According to the standards provided by the ISO, the preferred number of frequency bands in an equaliser is ten [ISO, 1997].

Equalisers are used for many reasons. One of the most important of these is to reduce frequency masking. This refers to the masking of one sound by another that has similar frequency content but is more prominent. This will

be discussed in detail in the literature review as there are papers dedicated specifically to reducing frequency masking that have been reviewed. Equalisers are one of the most important tools in a mix engineers toolbox. This is because they can be used to control the frequency content of each stem in a song. This allows the mix engineer to shape the overall frequency landscape of the song, giving each instrument a space in the frequency domain that is not interfered with by another stem.

1.2.3 Volume Control

Volume control is one of the less complex but still extremely important techniques that mix engineers use when mixing music. Volume control relates purely to adjusting the volume of each individual stem in a mix in order to make sure that each stem can be heard as it was intended to be, and that no single stem or combination of stems is too loud or too soft. Volume control is very important when it comes to mixing a song because if the song is too loud and causes clipping there will be distortion and when the song is played back it will not sound the way it was intended to sound.

When mixing a song, it is common practice to leave some headroom between the loudest part of the mix and the 0dB limit. This is done so that whoever is mastering the song has room to work with, and is able to master the song without worrying that any increase to the volume could result in clipping.

Not only is volume control useful for preventing distortion, but it also allows the listener to be guided to listen to certain stems in different parts of a song more than others. This allows mix engineers to draw focus and influence the

listener in a very intentional way which can be used to evoke specific emotions.

1.2.4 Reverberation

Reverberation is an effect that effectively elongates a sound. It is often added to stems in order to give the illusion that the sound was recorded in a specific environment with desirable acoustics. For example, a song recorded in a home studio can be made to sound like it was recorded in a giant hall. Reverberation is one of the more artistic and less technical aspects of mixing a song. This is because different mix engineers and different listeners will prefer some types of reverberation over others and because the addition of reverberation does not affect the frequency content in the same direct way that equalisers do. Reverberation often leads to more frequency masking issues because the original sound is being repeated. For the purpose of this project reverberation has been omitted, firstly because this is often an artistic choice, and secondly in order to reduce the overall complexity of the project.

1.2.5 Panoramic Positioning

Panning or panoramic positioning allows the mixer to position certain stems in either the left or the right speakers in stereo audio systems. This is a very useful tool that often helps with frequency masking. Mix engineers can place two sounds that would normally cause frequency masking on opposing sides so that they will no longer interfere with each other. The listener will be able to clearly distinguish between the two. Panoramic positioning is often used to reproduce the environment in which the music was recorded. If, for example, in the studio the drum kit and electric guitar were on the left-hand side of the studio and the piano and vocalist were on the right-hand side, the mix

engineer might reproduce this by panning the drums and electric guitar to the left and the piano and the vocals to the right. Single instruments are often panned in multiple directions. For a drum kit for example, mix engineers will often pan the snare, tom toms and the hi-hats to the sides and leave the kick drum in the middle to capture what the drummer would hear while playing the drums. In this project, in order to reduce complexity and ensure easy integration with the various signal processing libraries, the dataset was converted to mono which meant that panoramic positioning was not possible.

In [Tot, 2018], the process of audio mixing is described as iterative and having three main phases; vision, action and evaluation. The mix engineer is responsible for listening to the song and using their extensive knowledge and past experience to develop a vision for how the final musical piece should sound. It is then their responsibility to take the correct actions, using the techniques mentioned above, to implement that vision. The mix engineer then has to evaluate the mix, not only from a subjective but also a technical point of view. What may seem non-technical is often quite the opposite. For example, ensuring that each individual track is clear, present and not 'swallowed' by the mix, is a topic that has been covered in a number of signal processing journals called frequency masking as mentioned previously.

1.3 Project Goal

The goal of this project was to use machine learning to perform automatic mixing of musical compositions, or as this paper will sometimes refer to them, songs. More specifically, the goal was to develop a system that takes indi-

vidual, unmixed tracks and is able to mix these tracks using equalisation and volume control to produce a single song or piece of music which is as close to a professional mix as possible with machine learning. This took the form of a complex, multiple input, multiple output (MIMO) system. The various tracks are used as inputs to the system. These inputs are then fed through the system and the various parameters, like frequency band gains and volume gains are produced as outputs. These parameters are then used to transform the input sounds to produce a final mix. A system where the output parameters, for a given input track, are dependent not only on that input track, but also on the other inputs to the system (the other tracks), is called a cross-adaptive system. The system that was designed, therefore, was a cross-adaptive, MIMO system.

After evaluating the complexities of the various effects that could form part of the neural network automatic mixing system, it was decided that the two most useful, but also most technically important effects in the mixing process, were volume control and equalisation. In this paper, two separate systems were designed and trained, one for volume control and the other for equalisation. These two neural networks were trained separately and combined in a final step once the training was complete on both. Each of the neural networks were trained on the same data, however the formats differed between the two networks. For the volume control neural network, the original time series data, or 'waveforms', from the datasets were used. For the equaliser, the waveforms were transformed to the frequency domain in order to be better able to manipulate the frequency content of the signals as equalisers do. The transformation used is explained in chapter 1.5.1 and the format of the input and output data used for each neural network is described in chapter 4.1.

1.4 Training Technique

Multiple papers covering automatic mixing as well as automatic mixing using machine learning are discussed in the literature review. One common feature in these papers is the use of surveys and/or mixing sessions that professional mix engineers participate in to develop a system that mirrors a professional engineer, whether through machine learning [Reed, 2000], or through using what are generally considered to be best practices [Pestana et al., 2014]. In these papers, groups of professional mix engineers are used in order to give the system enough data to make good decisions about what effects to use and how to use them. The more training data a system has, the better it will be at making decisions. For this reason, it was decided that the data that would be used to train the neural networks would come from the MedleyDB dataset [Bittner et al., 2014]. MedleyDB offers a set of 74 songs that come with all the unmixed tracks as well as a final professional mix of the song. The data was split in the usual manner, 80% for training and 20% for testing. It was decided that songs would be split into training songs or test songs. If a song appeared in the test set it did not appear in the training set. This choice was made because there is not enough difference between multiple sections of the same song to claim that good performance on an unseen section of a song that also existed in the training set means that the neural network generalises well. The dataset will be discussed in more detail in chapter 4.

1.5 Evaluation Criteria

In order to test the system and evaluate the quality of the mixes being produced, it is important to have a standard measurement that is purely technical, due to the fact that listening sessions (often used in automatic mixing papers [Reed, 2000]) introduce personal bias and subjectivity which make the results unreliable. In this project, two comparisons were made. The first was the comparison used for the volume control neural network. The volume control neural network's output was the mixed time series information (waveform) of the song. This waveform was compared with the waveform of the professionally mixed output using the mean squared error between the two. The smaller the mean squared error the more like the professional mix the output was. For the equaliser, the spectrograms of the final mixes produced by the neural networks and the spectrograms of the final mixes of the professional mix engineers were compared also using the mean squared error. The spectrograms contain the frequency content generated by the short time Fourier transform of the tracks.

1.5.1 The Short Time Fourier Transform

The short time Fourier transform is a mathematical tool used to transform a signal from the time domain to the frequency domain in order to get information about the frequency and phase content of a particular signal. For this project the discrete short time Fourier transform was used. The formula is as follows:

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

Where $x[n]$ is the signal being transformed, $w[n-m]$ is the time shifted window function, m is discretised time and ω is frequency. Where a normal Fourier transform gives the overall frequency content of a signal the STFT does the same but allows one to obtain information about how the frequency information of a signal changes over time due to the fact that a Fourier transform is being applied at multiple time points throughout the signal. This is done using the window function. As the window moves through the signal along the time axis Fourier transforms are done at each step and added to a results matrix.

In order to be able to perform the inverse short time Fourier transform to get the time series representation of the mixed signal output (from the frequency information that is the output of the equaliser neural network) it was important that an appropriate window function was selected. Without overlapping window functions that taper off at the edges the reconstructed sound would have had inconsistencies at the edges where the individual windows were stitched back together. For this project the window function used was the Blackman window function. The formula for the Blackman window that was used is shown below:

$$w(n) = 0.42 - 0.5cos(2\pi n/M) + 0.08cos(4\pi n/M)$$

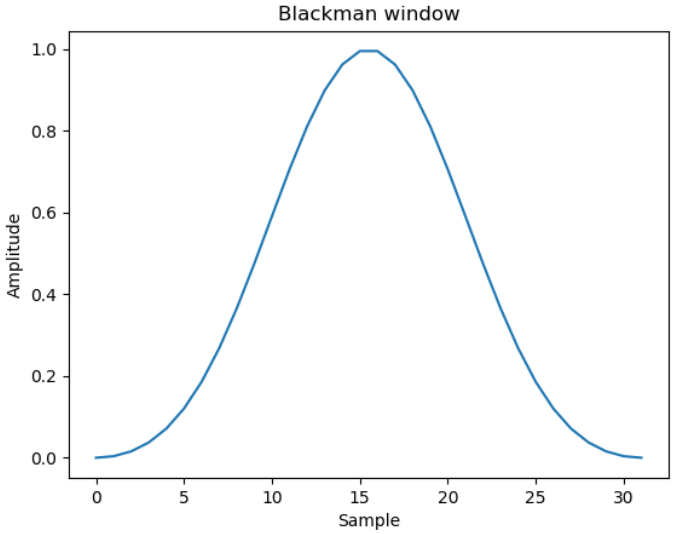


Figure 1.4: Time domain representation of Blackman window

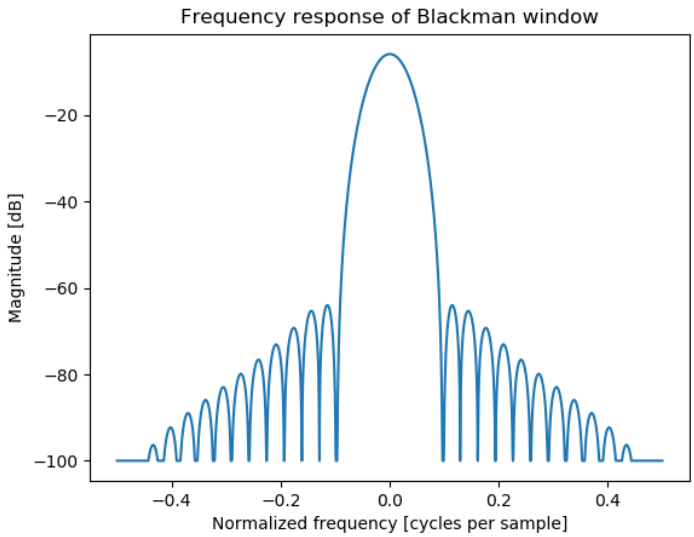


Figure 1.5: Frequency domain representation of Blackman window

The STFTs were used to produce spectrograms of the sound signals. Spectrograms can be calculated using the STFTs by taking the magnitude squared of the STFTs. Given the spectrograms of the final mixes produced by the system and those from the MedleyDB dataset, various techniques can be used to compare the songs numerically. The different methods considered are covered

in the literature review.

For the equaliser neural network, spectrograms were chosen over waveforms because the data of interest when it comes to equalisation in mixing is, for the most part, frequency based. Equalisers amplify and cut per frequency band, and filters are used to eliminate or allow different frequencies. While there are common effects like dynamic range compression that are not always frequency dependent, using spectrograms would not prevent one from being able to analyse that sort of data as the intensity of the sound is captured by the intensity of the spectrogram. Another type of measurement, the Mel-Frequency Cepstral Coefficients (MFCC) was also initially considered as an option. A brief comparison between MFCC and spectrograms is given in the literature review. Having compared the two, the choice was made to use spectrograms. The goal of this project is not to develop a neural network that is able to accurately tell whether or not a song has been professionally mixed, it is to develop a neural network that does the actual mixing.

1.6 Considerations

This project is essentially a system identification problem, where the system (the mixer) takes inputs (the unmixed individual tracks) and generates an output (the parameters used to mix the song).

When deciding how the data would be structured it was important to decide how to split up the songs. One consideration was whether the neural networks should be trained on songs as a whole or on much smaller sections of songs.

For example, do you train the network on an entire song before you expose it to the next song in the dataset, or do you split the song into different sections and randomise the order in which the network sees these sections?

In the very beginning stages of experimentation with the training data, it was noted that when training on an entire song before exposing the network to the next song, the networks would perform very well and minimise the loss and error metrics fast. But as the neural networks began training on the next song, the error and loss metrics jumped steeply. This indicated that this method was causing the neural networks to overfit the song they were being exposed to at that time, then almost discarding that training when exposed to a new song and overfitting that song instead. This was not at all ideal and luckily was spotted early in the experiment.

For this reason it was decided that the neural network would be trained on sections of songs and those sections would be randomised so as to minimise overfitting a particular song. There were two advantages to this method. The first was that when the network was trained this way the loss and error metrics gradually declined over the training period and there were no significant jumps when switching between different songs as there had been previously. The second advantage was that using small sections of the songs meant that this method would also allow for the system to perform live mixing as it would be able to take the incoming signal, compute the spectrogram and then, based on its training, determine what coefficients to use for the different effects. The way the dataset was split up is explained in detail in chapter 5.1. In order to implement the neural networks as effectively as possible it was important to form an understanding of not only the various automatic mixing systems out

there, but also of machine learning and automatic mixing in general.

Chapter 2

Literature Review

2.1 Machine Learning

Artificial intelligence, machine learning, and neural networks are terms that have been in use for many years, but have seen an increase in focus since the computational power required for their implementation has become more widely accessible. Artificial intelligence is an umbrella term under which the machine learning field of computer science and engineering falls. Neural networks are one of the tools used in this field to create algorithms that can interpret and learn from data in order to achieve goals through adaptation [Kaplan and Haenlein, 2019]. The concept of artificial intelligence has been known for a long time and while not all of the neural networks designed today have been programmed to take into account Isaac Asimov's three laws of robotics, all of them have helped advance the science to where it is today [Asimov, 1942].

In machine learning we have two main types of learning, supervised and unsupervised. Supervised learning takes place when we have data which allows us to form training examples with inputs and outputs which we use to train a

system. The goal of this type of learning is to be able to give the system new data and have it reach accurate conclusions [Simon et al., 2016]. Unsupervised learning occurs when the system is given data and asked to determine and find patterns, structures and relationships within the data that may be useful and may contain knowledge that is not obvious [Simon et al., 2016]. The type of learning used in this project is supervised learning because there is a dataset with a number of training examples that were used to train a system to mix songs and produce a professional quality output mix.

As machine learning evolved a new field emerged, deep learning. Deep learning refers to the practice of training neural networks with more than one hidden layer and is a methodology used in many applications due to its impressive performance in complex tasks, one example being computer vision [Serre, 2019]. Deep learning in the form of Convolutional neural networks has made major strides in the world of computer vision and image processing.

In situations such as described in this paper where supervised learning is used, the goal is often to train a neural network using data and have it recognise patterns and extract features from the data that allow it to generalises its predictions to data it has never seen before. There are two types of predictions neural networks can make, and the types of predictions being made allow us to determine whether these neural networks are performing classification or regression. Classification is the process of using information provided at the input to predict a label for the output that relates to one of a set of discrete classes. Regression is the process of predicting the value of a continuous variable from a set of input information [Stegemann and Buenfeld, 1999].

Neural networks often consist of an input layer followed by a number of hidden layers, and then an output layer. A layer is a set of cells or nodes and a cell or node is a computational unit that can perform linear or non-linear operations on one or many inputs and produce an output [Stegemann and Buenfeld, 1999]. Information is passed from layer to layer through weighted connections between nodes in each layer. At a node, the weighted inputs are summed with any biases and then given as the input to an activation function, the output of which is the output of the node. At the output layer the outputs of the nodes form the output of the neural network. In the training process, feeding the dataset forward through the network from input to output is called an epoch. The output produced by the neural network is then compared to the correct output given by the training data using what is called a loss function. The loss function produces a value that indicates how similar the output of the neural network is to the desired output. There are many different options when it comes to choosing loss functions. This paper uses the mean squared error which takes the average squared difference between the network's final mixes and the professional final mixes. The loss value produced by the loss function is then used to alter the weights used in the neural network using algorithms such as back propagation.

Back propagation makes use of an optimiser function where the goal is to feed the loss backwards through the network, performing partial derivatives on the way, in order to determine how each weight contributed to the loss at the output. In doing this one is able to adjust the weights in a way that will minimise that loss the next time data is fed forward through the network. In this paper both Adam and stochastic gradient descent (SGD) were tested. Adam was chosen due to the fact that it often converges faster than SGD but SGD

was used as well to compare the two. The weights are adjusted depending on a variable called the learning rate. While the optimiser tells us how the magnitude of the weights should change (larger or smaller), the learning rate determines how much the weights should change. As the neural network trains, it is important to ensure that the network is not overfitting the training data. We know a neural network is overfitting if the network's loss is significantly less for the training data than for data it has not seen (often referred to as validation loss). This means the neural network does not generalise well.

When training a neural network, one will often split the training data into data that is used for the actual training and validation data. The validation data is used to check for things like overfitting. As the neural network trains and the loss decreases the model becomes more and more able to make accurate predictions. Once training is complete, the model should be able to make accurate predictions on never before seen data and in so doing, produce a model that can be used in the real world on new data.

2.2 The History of Automatic Mixing

Automatic mixing is a relatively new topic of discussion in the academic world when compared to topics in other research areas of signal processing. According to [De Man et al., 2017] the term automatic mixing was first used in relation to the automatic gain control of multiple microphones recording a single performance. There are a number of reasons automatic mixing was needed in this situation such as the fact that when multiple microphones are recording the same performance there can be unwanted amplification of ambient noise picked up by all the microphones. In [Dugan, 1975] a number of solutions to

this problem are presented. One of these is to use separate microphones to determine the ambient noise volume levels (in dB). These levels are then used as threshold levels and only signals from the microphones where the input level is higher than the threshold are given a gain by the automatic mixer. The input levels of these microphones are mapped to gains by the automatic mixer in a way that maintains a chosen system-wide gain.

The second reason automatic mixing is needed in this situation is that when multiple microphones pick up the same source from different locations, the sound arrives at the different microphones at different times. Depending on the delay between what each microphone picks up, the signals received at the various microphones can be out of phase. When these signals are added to the overall mix it can cause undesirable amplification and attenuation of the original signal. This is called comb filtering and is a form of distortion. In [Perez Gonzalez and Reiss, 2008] this is handled by selecting one microphone as a reference signal and then using cross adaptive methods where the impulse responses of all the signals are taken and the placements of the peaks are compared in each. The algorithm then goes through all the impulse responses and finds the most negative delay (the signal that arrives first at the mixer), sets that as the new reference signal, then calculates the time delay between that reference signal and all the other signals in the mix. Each signal is then delayed accordingly to ensure that the signals in the mix are all in phase.

2.3 Automatic Mixing Guidelines

One trend that makes standardisation of automatic mixing difficult is the fact that over the years, mixing and sound engineering in general has become more artistic and less technical. This means that many decisions as to what the best practices should be have become personal and not necessarily based on any set of guidelines [Pestana et al., 2014]. This is addressed in [Pestana et al., 2014] where an attempt is made to use various different techniques to review audio mixing procedures to produce a set of best practices to be used especially in cases where machine learning will be implemented for automatic mixing. A number of the standard assumptions about what makes a good mix, that have been perpetuated either by literature, interviews with professionals or by previous implementations, were evaluated and tested in a few ways. One method was to analyse the features of number one hit singles to determine if these mixes had followed the practices outlined by the assumptions mentioned above. Features are essentially characteristics of a signal, for example the power spectral density (PSD), minimum and maximum amplitudes, and RMS value. Another way that these assumptions were tested was through mixing exercises by professionals. Through their evaluations, it was found that a third of the assumptions or guidelines that have been widely used in general practice were wrong when evaluating the number one singles and mixing exercises. One example is the assumption that mixers should prefer cutting over boosting when performing equalisation. Cutting and boosting in this case refers to how the equalisers are used on a per frequency band basis. Boosting amplifies certain desired frequencies whereas cutting reduces the presence of certain frequencies. This assumption is widely used and is often emphasised as one of the most important rules to audio mixing, however [Pestana et al., 2014]

found that the majority of professional mixes and number one singles had not followed this rule.

There are two main conclusions drawn from [Pestana et al., 2014]. The first being that one of the main areas of focus when it comes to automatic mixing needs to be that of frequency masking [Ramırez and Reiss, 2017], as this is a phenomenon that has an effect on a number of features in an audio mix. Frequency masking takes place when two signals with similar frequency content are present in the same mix at the same time. Frequency masking describes how the perception of one signal is affected by the presence of another. For example, consider a song that has a bass instrument as well as drums. The kick drum will often be masked by the bass instrument because they both have low frequency content. There are various techniques that deal with this problem that will be covered in this project. One technique is covered in [Hafezi and Reiss, 2015] where audio tracks are analysed using feature extraction which gives information about the magnitudes of the different frequency regions of the input tracks. Each input track is then compared to all the other input tracks in order to determine which tracks it masks. Once this has been determined, equalisation is applied to the frequencies of the given input track that are causing the masking. The equalisation is done by reducing the presence of the frequencies that are causing the masking in the given input track, which in turn allows the track that is being masked to be heard more clearly.

The second main conclusion from [Pestana et al., 2014] is the suggestion that any attempt at automatic mixing should consider that while there are set events that should trigger mixing decisions in a particular order, none of these decisions should be made until the overall combination of these decisions is known. This effectively means that automatic mixers cannot simply make de-

cisions based on a single sound or section without assessing how that decision would affect the mix as a whole. This leads to the need for a cross-adaptive system.

2.4 Cross-Adaptive Systems

Cross-adaptive systems have become commonplace in the world of automatic mixing. This is because each signal in a mix is affected by all the other signals, and changing something about one signal will have an effect on the perception of each of the other signals in the mix, and on the mix as a whole. Cross-adaptive signal processing is used in [Perez-Gonzalez and Reiss, 2009a] as a means to adjust the volume faders of different microphones in a live mix in order to ensure that the loudness of each signal in the mix is the same and tends toward the average loudness of the mix as a whole. The cross-adaptive method implemented in [Perez-Gonzalez and Reiss, 2009a] is used as the basis for the cross-adaptive method used in [Perez-Gonzalez and Reiss, 2009b] which uses the ratios between the loudness indicators of the individual channels and the overall mix itself. These papers use cross-adaptive methods to ensure that the loudness of all channels is the same. According to [Pestana et al., 2014], this is one of the incorrect assumptions about what makes a good mix. The papers are, however, not to be disregarded as their use of cross-adaptive methods can be modified to change the goal from being equal loudness to equal masking.

2.5 Machine Learning in Automatic Mixing

This project aims to use machine learning to form a cross-adaptive method that will be able to go from separate unmixed signals to a set of mixing parameters that can be used to produce a final, professional quality mix. One paper that has attempted to use machine learning for automatic mixing is [Reed, 2000], where inductive learning is used to train a neural network that uses nearest neighbour pattern recognition to determine how to best equalise each signal to achieve a certain timbre. Timbre is essentially the quality of a sound that is different from its pitch or intensity. It allows listeners to differentiate between different instruments or voices playing or singing the same note. In [Reed, 2000], the system is trained by getting a group of seventeen sound engineers (professionals and students) to equalise a number of signals with a given goal in terms of the type of tonal qualities the system should enhance. The system then stores the sounds, the goal and the equaliser values, namely which frequency bands were amplified or cut and by how much. These are then used later, when the system is given a particular sound and a particular tonal goal, to identify what equalisation values should be used by implementing the nearest neighbour technique.

One downside of this method is that the system will be reliant on a small group of 17 sound engineers. As the system is trained using that particular group of sound engineers' manipulations, it will be very good at mixing a song in the way that those engineers would mix it, but not necessarily in the best way possible. That is where this paper aims to improve the result by using the best practices from [Pestana et al., 2014] and a dataset provide by MedleyDB [Bittner et al., 2014] in order to produce a system that is an amalgamation

of hundreds of different professionals, including some Grammy Award winning sound engineers and their techniques. This is done with the aim of developing a system that can produce a professional mix using machine learning without having to be trained by a particular person.

While the use of nearest neighbour pattern recognition described in [Reed, 2000] is promising, it is designed to work on an individual signal without taking into account how the equalisation of that particular signal will affect a mix as a whole. For this reason, it is important to note that a significant amount of work would still need to be done in order to get that system to work on a full mix. Another difference between the project described in [Reed, 2000] and the model in this paper is that in [Reed, 2000] all of the mixing was done during the project meaning that all of the coefficients given to the effects were known, and could therefore be used to train the system. In this project none of these coefficients are known, meaning that it is not as simple as finding coefficients for a given function because the function itself is not known.

In Mimilakis et al. [2016], a deep neural network (DNN) was designed to perform dynamic range compression. In their paper this was done as part of the mastering process, however the concepts could be applied to mixing as well. Their model performs a short time Fourier transform in order to obtain the magnitude and phase data of the signals. Their network used the magnitude data to train their model while the phase data was left unchanged, but used to reproduce the sound at the output by performing an inverse short time Fourier transform.

The DNN used in [Mimilakis et al., 2016] consisted of seven layers of 260 fully

connected nodes with ReLu activations. The optimiser used was Adam [Mimilakis et al., 2016]. Their network was compared to the professionally mastered songs as well as to the masters produced by competing existing software, Auto Audio Mastering System [Velde, 2020], that claims to also perform automatic dynamic range compression. The comparison was done in a subjective evaluation performed by nine mix and mastering engineers with sufficient expertise. Overall, the professional mix was rated the best followed by the proposed DNN and the worst rating was for the existing AAMS software.

In [Scott et al., 2011], features were extracted from the waveforms and used to train a neural network that produced volume gains as an output in order to perform automatic multi-track mixing. The features used were spectral centroid (which is the weighted mean of the frequencies of a sound and is often used as an indication of the brightness of a sound), RMS energy, and slope/intercept from fitting a line to the spectrum [Peeters, 2004]. These features are extracted from the waveform and given as inputs to the neural network that then outputs the volume gains which are compared to the actual gains. One thing to note in this paper is that they did not have access to the true gains used and had to estimate them. The gain estimation process is described in their paper. The network architecture in terms of the number of nodes and layers used is not given.

It is important to note that both [Mimilakis et al., 2016] and [Scott et al., 2011] are designed to work as off-line systems as they require all of the time frames of the audio signals they are given to mix or master. This means that neither of these systems would be applicable in the case of live mixing.

In [Nercessian, 2020], a neural network is developed to perform EQ matching. This is the process of selecting the parameters of an EQ such that when given an input sound and a reference sound, it produces an equalised version of the input sound whose spectral qualities match, as much as possible, those of the reference sound. As mentioned in chapter 1.2.2, equalisers are usually made up of a combination of filters with various parameters. The goal of [Nercessian, 2020] was to design a neural network that could produce these parameters for a predefined number of filters. The paper used twelve filters where one of the filters is a low shelf filter, one is a high shelf filter and the remaining ten are peak filters. The neural network was designed to produce an output consisting of the frequencies that each of the filters should be set at, the Q value (shelving slopes and peaking quality factors) for each of the filters, as well as the gains for each of the filters.

The input to the network was the desired magnitude frequency response and the outputs were the settings mentioned above for the different filters. Their network had three hidden layers with 256 densely connected nodes in each layer, and utilised ReLu activation. The loss function used in [Nercessian, 2020] was the weighted combination of the MSEs of two different comparisons, the first being the magnitude frequency responses and the second being the EQ parameter values. This was then also optionally combined with a regulariser on the gains. Their model performed better than the approach from [Abel and Berners, 2004] which they refer to often in [Nercessian, 2020] as the baseline approach. The limitations of the solution in [Nercessian, 2020] are however that the filter types are predefined.

An interesting observation was made that [Nercessian, 2020] was written under

iZotope which is an audio software company that makes plugins used by the author as well as many award-winning mix engineers.

Another system that uses artificial intelligence and machine learning to master songs is the proprietary software, LANDR [LAN, 2020], however there are no publications that discuss the design or the type of neural network used, and there is no information regarding what dataset LANDR is trained on. LANDR may, however, prove useful when it comes to assessing the performance of the system that this project aims to create. Once again however LANDR is an off-line system and cannot be used for live mixing.

2.6 Spectrograms as Evaluation Criteria

As stated previously, in order to determine the quality of the mix without using surveys and subjective opinion, as is done in several of the papers mentioned above, it is important to choose a measurement that can accurately determine how similar two different mixes of a song are. In this paper, the two mixes are the output of the system and a professionally mixed song from the MedleyDB dataset. This project is not the first to compare spectrograms of songs. This has been done before in [George and Shamir, 2015] to compare the similarities between songs in different genres. This project will use the spectrograms in a similar way, however the songs being compared will be the two different mixes of the same song. In [George and Shamir, 2015] the songs are converted into their 2D spectrograms which are then analysed using the **wndchrm** scheme [Shamir et al., 2008]. **Wndchrm** is a system designed for biological image analysis and is used to analyse images of cells, tissues, organelles and many others. In [George and Shamir, 2015] **wndchrm** is used to extract more than

2000 different features from the song. These features can then be used to compare various songs and compute their similarities. A similar project where spectrograms are used as a method of comparison is found in [Scerri, 2016], where spectrograms and image processing techniques are used to identify songs without metadata by comparing their spectrograms to spectrograms of songs with metadata in a database. This project will compare the spectrogram arrays produced by the STFT algorithm using the mean squared error.

2.7 Machine Learning in System Identification

There is a significant amount of literature that deals with system identification using neural networks. In [Kumpati et al., 1990], multiple neural networks with different structures are used to identify not only single input, single output systems (SISO), but also multiple input, multiple output systems (MIMO). Most importantly in relation to this project, interconnections of recurrent and multilayer neural networks are developed with the use of back-propagation to determine the approximate parameters of the unknown functions. In [Kumpati et al., 1990], four models are used. In the first and second models the outputs are linearly dependent on the inputs. The key difference is that in the first model the output depends on the previous outputs, while in the second model the output depends on the previous inputs. In the third model the dependence of the current output on the previous outputs and inputs is non-linear but can be described by two separate functions, one for the inputs and one for the outputs. In the fourth model the current output depends on one non-linear function that includes a combination of the past inputs and outputs as parameters. While intuitively the fourth model seems like it would produce the most accurate results, it is found that often the third model performs just as

well and is easier to implement. Applying the logic of the third model from [Kumpati et al., 1990] to the problem presented in this paper would mean that the coefficients used on the different effects that would be applied to the tracks (the output of the system) would not only be dependent on the inputs (the spectrograms of the individual tracks) but also on the current output (spectrogram of the final mix). This allows for effects to be added in a way that considers the individual track that the effect is being applied to, but also considers the current state of the final mix, and the states of all the other inputs. This relates to what was mentioned above and in [Pestana et al., 2014] about how it is important to take the mix as a whole into account before applying any effects to the inputs, as opposed to applying effects based purely on the state of the current input.

In [Chen et al., 1990] the use of single layer neural networks in the system identification of non-linear systems is presented and evaluated. The loss function used is a recursive Gauss-Newton algorithm. Model validation was also performed using Chi-squared tests. It was found that initially using four nodes in the hidden layer meant that the network was not generating valid models. However, using five nodes in their single hidden layer caused the system to produce a valid model. In [Chen et al., 1990] both real world data and simulated data was used to test the neural network and it performed system identification successfully with both sets of data.

In [Genc, 2017] the use of deep convolutional neural networks for system identification is discussed. The use of deep convolutional neural networks for system identification is advantageous as it eliminates the need for prior knowledge about the structure of the equations for which the system has to determine

coefficients. The library used in [Genc, 2017] was the Keras library [Chollet et al., 2015] for Python. This library uses Tensorflow as the backend [Abadi et al., 2016].

2.8 Machine Learning in Music Classification

Music classification is a field that is widely covered in literature. The goal of these papers is usually either to classify music by genre or by time period. Upon review, it appears that the most popular neural network structure for music classification is the convolutional recurrent neural network (CRNN). This type of neural network is often used for speech recognition as well, and has even been used to classify bird sounds. While these are all sounds, the way the CRNN is used is as an image classifier. The images that are fed into the CRNN are usually either spectrograms as mentioned above or Mel-Frequency Cepstral Coefficients (MFCCs). In [Choi et al., 2017], a CRNN is used for music tagging which is similar to genre classification because tags can be genres, but they can also be moods or feelings associated with a particular song. In [Choi et al., 2017] the authors' CRNN is compared to three convolutional neural networks (CNN) and it is found that the CRNN outperforms the CNN in all the tests.

While this project is not a classification problem it is reassuring to see that spectrograms are often used when dealing with music and digital signal processing.

Chapter 3

Problem Statement and Objectives

3.1 Problem Statement

The problem that this project aims to solve is that there is a need for open-source automatic mixing software due to the number of musicians with access to recording equipment and software, without having the knowledge needed to produce professional quality mixes. For this reason, the purpose of this project was to develop a system that is able to take unmixed individual tracks and produce mix parameters that could be used to produce a professional quality mix. More specifically the objectives of this project are listed below:

3.2 Objectives

- To use neural networks to create a system that is able to take multiple audio input signals, which will be the individual tracks that make up a song, then analyse and perform computations on them in order to

produce a set of EQ and volume gain parameters which will be used to transform the original inputs to produce a single professional quality mix of that song.

- To train this neural network using the songs provided by MedleyDB - 80% of the songs in the dataset will be used for training purposes [Bittner et al., 2014].
- To test the performance of this neural network mixer - the remaining 20% of the songs in the dataset will be used for testing.

Each of these objectives involves a number of complex steps which need to be broken down. Keras (one of Python's many neural network API's) was used to create the neural networks. Using a musically informed architecture was the key to producing mixes that not only had low losses but actually sounded high-quality. At a high level, the architectures of the successful neural networks were such that the gains applied to each of the frequency ranges of each of the tracks depended on the spectral content of each input track in the same frequency ranges. For example, the gains of the higher frequency ranges for each track depended only on the high frequency content of the inputs and ignored the middle and low frequencies. The architectures used are discussed further in the implementation chapter 5 section 5.2.

In order to train the neural networks the input-output pairs were used on the GPU/TPU accelerated runtimes provided by Google Colab with sufficient RAM (32Gb) to deal with the size of the input-output pairs and the model architecture. The structure of the input-output pairs is described in the dataset chapter (chapter 4). The training was done on 80% of the dataset.

Once training was complete, the testing of the neural networks was done using the remaining 20% of the dataset by running Keras' predict function on the test data and comparing the spectrograms and waveforms produced by the neural networks and the spectrograms and waveforms of the professional mixes.
In order to create a neural network that goes from unmixed tracks to a mixed song, it is important to clarify exactly what effects will be used, as this will affect the nature of the system that the neural network aims to identify. There are multiple effects that can be applied to a track in the mixing process however the most commonly used are the following; equalisation, volume or gain control, compression, panning, modulation, delay and reverberation [Tot, 2018].

According to [Pestana et al., 2014] reverberation can be substituted for delays in order to reduce frequency masking. This means that the use of reverberation could be neglected. In this project modulation is also going to be neglected as it is often a personal and very artistic addition to a song and does not contribute to it being technically good or bad when talking about a professional mix. When mixing, it is important to remember that one of the most important goals of mixing is to address frequency masking [Ramırez and Reiss, 2017]. This is usually done using equalisers and phase manipulation. In order to limit the scope of this project while keeping the most important goals of mixing in mind the effects that will be used by the system are:

- Equalisation
- Volume or gain control

The equaliser is an effect that deals directly with how the different frequencies of a sound are manipulated. For this reason, it was decided that the equaliser

neural network would function in the frequency domain. Volume control affects sections of a stem as a whole as opposed to affecting distinct frequencies differently like the equaliser. For this reason it was decided that the volume control system would function in the time domain. Given the difference in the information required by these two systems it made sense initially to split them into two separate neural networks to train them separately. The reasoning behind this choice will become clearer in the next section when the descriptions of the formatting of the input-output pairs is discussed. It is noted here and in the Recommendations for Future Work chapter, that combining these two networks into one pipeline could possibly lead to even better results.

Typically for equalisation there are certain frequency bands to which either positive or negative gains are applied in order to boost or cut the presence of certain frequencies. There are usually between 8 and 32 frequency bands depending on the equaliser being used. For the equaliser neural network, each band of each instrument corresponds to a node in the neural network. For volume control the nodes in the neural network correspond to the gains applied to the different tracks in the song. The process behind deciding the number of frequency bands that would be used for this project is explained in chapter 4.1.

Chapter 4

Dataset

The dataset consists of 74 songs from various genres with varying combinations of sounds. For a full list of all the sounds present in the dataset see appendix A. The sounds in the dataset could be anything from vocals or instruments to sound effects (FX). The neural networks needed to be designed in such a way as to be able to take in as many or as few stems as necessary. There are 47 different sounds in the dataset. The neural networks were designed so that they always expect all of the 47 different sounds as inputs regardless of what sounds are present in the section of a song it is training on. Some of the songs will only have three stems for example, meaning the other 44 stems that the neural network expects as inputs will be supplied as zeroed input. This accounts for the varying number of stems per song. Songs are also varying lengths and so each song was split into slices of equal length. This procedure is discussed in more detail in chapter 4.1.

Each song in the dataset consisted of its final mix which is the professionally mixed output of the stems which are in turn a combination of the raw files. The relationship between raw files, stem files and mix files is shown below in Figure

4.1. Each song had metadata and annotations that provided extra information about what sounds were present and which raw files belong to which stems as well as some other useful information like what genre the song is classified as. The only metadata used by the neural networks was the instrument or sound type.

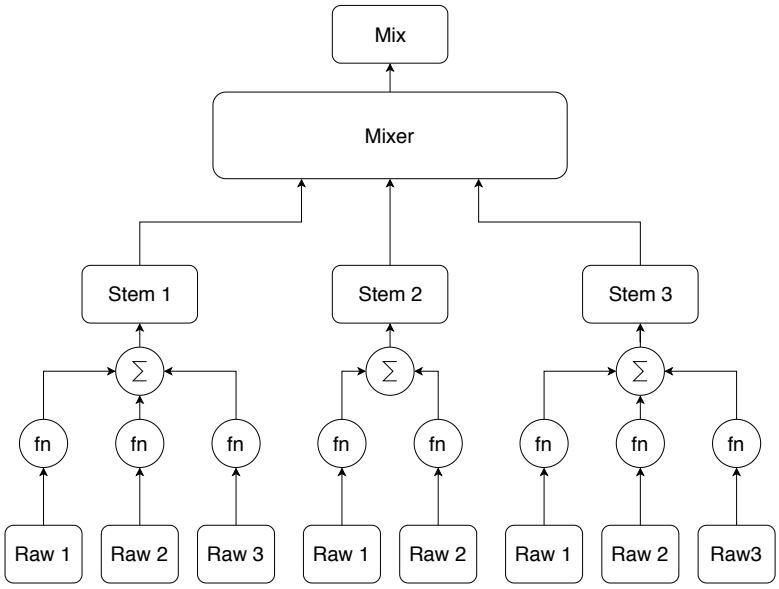


Figure 4.1: Layout of files in dataset

The metadata was used to determine what sound each stem file was classified as. This was done so that each of the individual sounds present would always be mapped to the same nodes so that the neural network gained an understanding of how to deal with a certain sound based on how it interacts with other sounds.

4.1 Formatting and Normalisation

Each stem and mix file was converted to the following standard format:

- Audio format: WAV
- Bit resolution: 32 bit
- Sampling rate: 44.1kHz
- Audio channels: Mono

While the volume control network used these waveforms as is, the short time Fourier transforms (STFTs) of these stems and mix files were taken to form the inputs for the equaliser neural network. When performing an STFT one needs to carefully consider a number of things:

- Fast Fourier transform (FFT) size
- Window type
- Window size
- Hop size

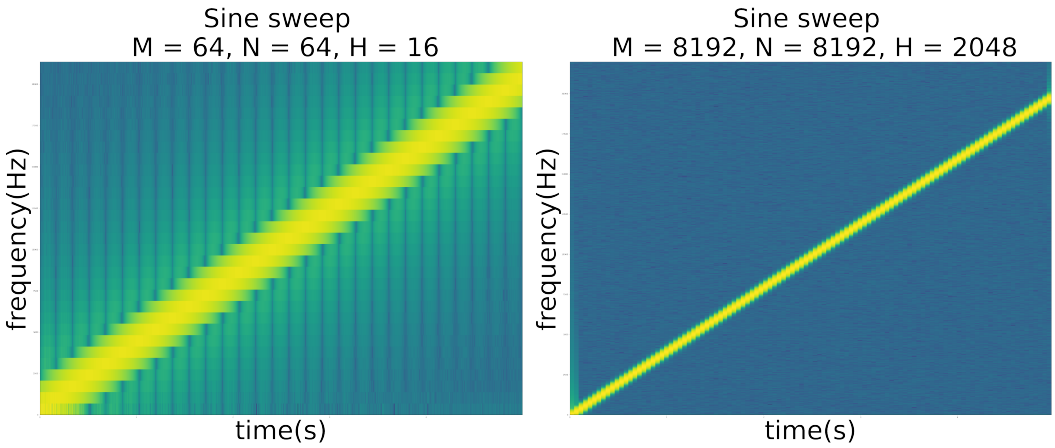
The FFT size specifies how many frequency bins there will be in the frequency domain and defines the frequency resolution. With a fixed sampling rate, a higher FFT size leads to higher frequency resolution which means a more accurate representation of the stems in the frequency domain. A sampling rate of 44.1kHz results in a reproducible spectrum of 0 - 22.05kHZ and the number of bands divide up that interval equally. According to [Nisar et al., 2016] which describes the well-known time-frequency compromise, a higher window size results in a high frequency resolution and a low temporal resolution while a lower window size results in a low frequency resolution and high temporal resolution. Frequency resolution is defined by the following equation [Semmlow, 2018].

$$f_{resolution} = \frac{f_s}{N}$$

Where N is the FFT size and f_s is the sampling rate. With our sampling rate set at 44.1KHz we can then calculate the required FFT size (N) based on what frequency resolution we require. Initially it was intended that the frequency resolution be as granular as possible. It was set to 1Hz because that would mean the equaliser neural network would be able to manipulate the values of each individual frequency independently. This would however require significant computational power when training the neural network and so was eventually deemed unnecessarily large.

One potential option was to use an FFT size of 64 which is equal to twice the standard number of bands in a typical equaliser (32 as mentioned in chapter 3.2). The reason being that after the STFT there would then be 32 bands of positive frequencies that could be adjusted. Another option was to use a higher FFT size of 8192 in order to get a better frequency resolution but not so high that it was computationally unrealistic. With an FFT size of 64 frequency resolution would be approximately $f_{resolution} = 689Hz$ whereas with 8192 the frequency resolution would be $f_{resolution} = 5Hz$. In order to choose the most acceptable FFT size, a number of experiments were conducted.

A five second sine sweep from 20Hz to 20kHz was created (listen to it here: https://drive.google.com/file/d/1Gtuf16wRZqTDP3lXu5cUgVVtTx5JBD96/view?usp=sharing). The STFT was performed on the sound with two different FFT sizes, 64 and 8192. A Blackman window was used with a window size equal to the FFT size and the hop sizes were set to a quarter of the window



(a) Low frequency resolution (b) High frequency resolution

Figure 4.2: Sine sweep spectrogram comparison

size (for the Blackman window used the recommended hop size is approximately a quarter of the window size for the desired overlap) according to the recommendations in [Serra and Smith, 2019] and [Heinzel et al., 2002]. The spectrograms of both are shown in chapter 4.2.

As can be seen in chapter 4.2, the difference in frequency resolution is substantial and it is clear that the higher frequency resolution option is a much more accurate representation of the frequencies present in the signal and how they change over time. Given that the equaliser neural network would operate in the frequency domain it seemed appropriate to use the option with a higher resolution and so the FFT size was set to 8192. The hop size had an upper limit as specified as a quarter of 8192 but there was no lower limit. For this reason other hop sizes were used and compared as well. A hop size of 512 produced the most visually appealing spectrogram 4.3.

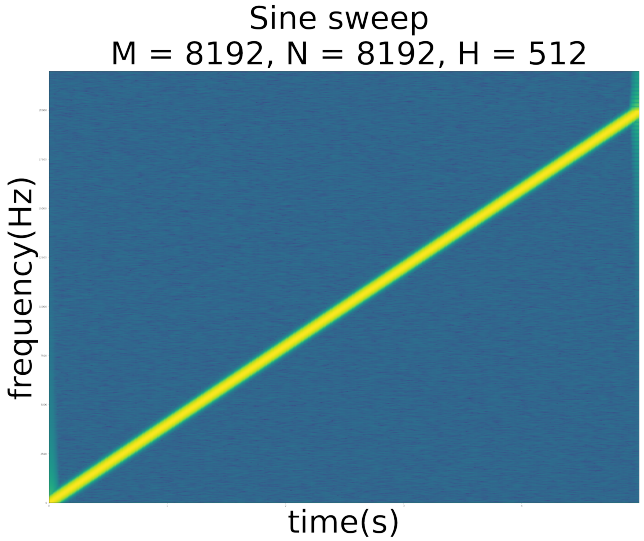


Figure 4.3: High frequency resolution and improved temporal resolution

It is however important to note that due to the window overlap percentage being significantly higher, there would be a higher overall value by which each data point would be multiplied in the windowing which results in over representation of the data. In order to test the effect of this, a random song from the dataset was chosen. Three STFTs and inverse STFTs were computed. The first two had the FFT and window sizes set to 8192 with hop sizes of 2048 (recommended maximum) and 512. The third with the FFT and window sizes set to 64 and hop sizes of 16 (recommended maximum). The spectrograms as well as waveforms were compared to see the effect of the smaller hop size.

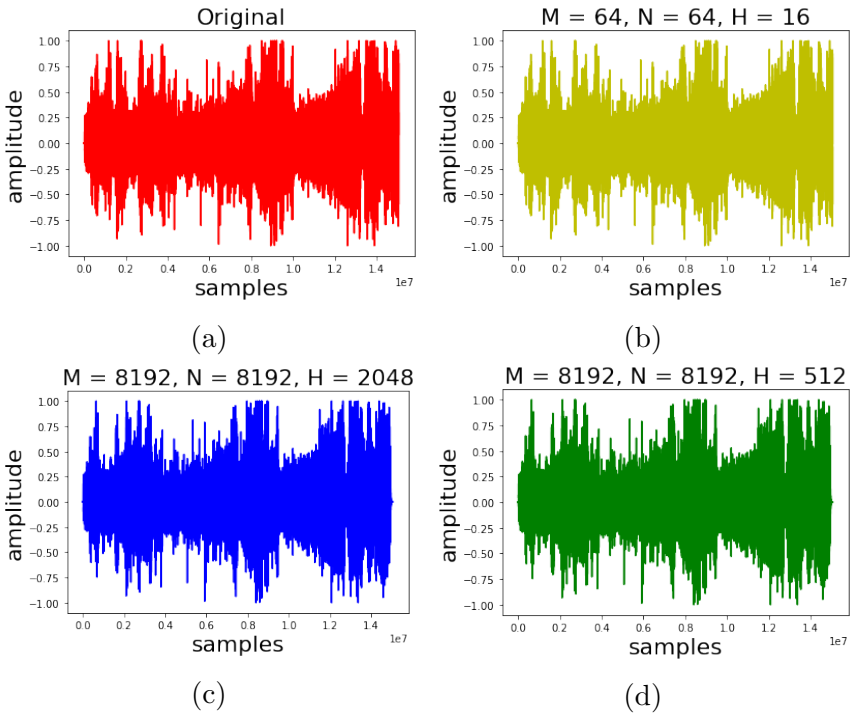


Figure 4.4: Waveform comparison between different FFT and hop sizes

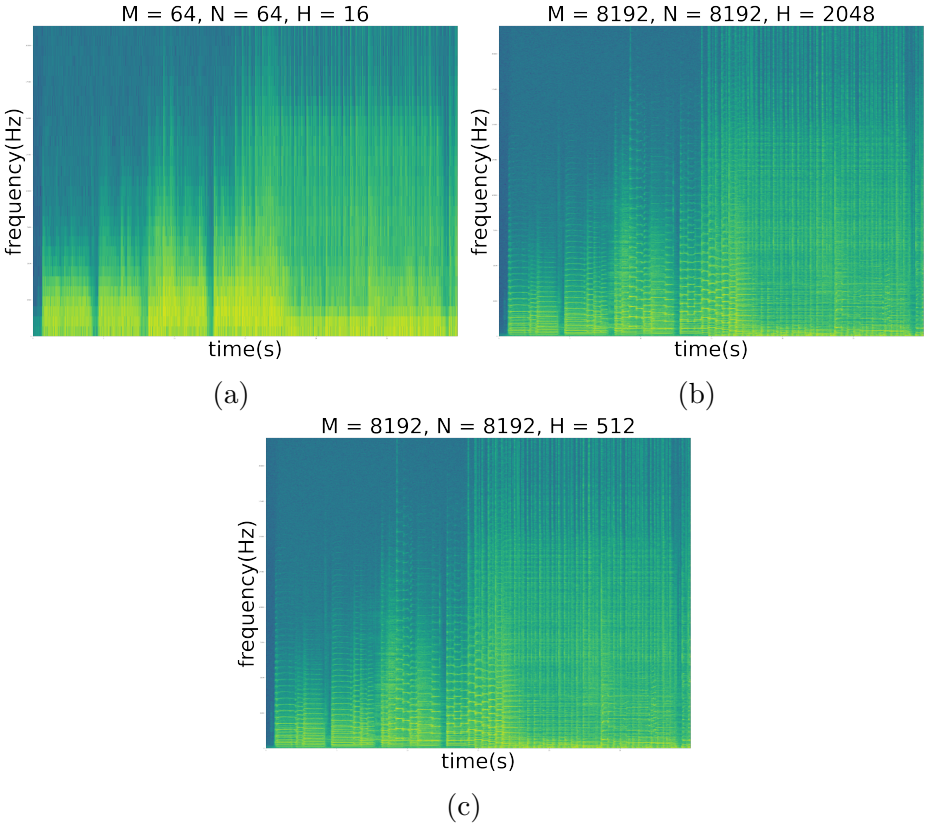


Figure 4.5: Spectrogram comparison between different FFT and hop sizes

Firstly, we see that there is no effect on the volume of the waveforms and no other noticeable effects on the contents of the waveforms themselves. We do however see very clear differences in the spectrograms. The difference between 64 bins and 8192 bins manifests itself, as expected, in the frequency domain and produces very different frequency resolutions. It was also noted, but not shown in the figures, that for a fixed FFT and window size, increasing the hop size above the recommended maximum produced substandard sounding audio after the inverse STFT was applied. This is supported by [Heinzel et al., 2002] and is because if there is insufficient overlap in the consecutive windows not all sections of the signal are equally represented. It was also noted that the as long as the hop size was less than or equal to a quarter of the window size, the audio was always reproduced perfectly and there was no auditory benefit to using a lower hop size. However, when comparing the spectrograms of both the sine sweep and the randomly selected song from the dataset, it is clear in chapter 4.3 and somewhat less clear but still noticeable in chapter 4.5c that reducing the hop size from 2048 to 512 produces a better looking spectrogram with a more accurate temporal resolution which results in a more accurate overall representation of the song.

Another factor to consider when choosing the FFT and window sizes for a song of fixed length is that performing an STFT with larger FFT and window sizes is faster than performing an STFT with small FFT and window sizes. This is not significant for off-line mixing of songs in a studio or at home where the amount of time taken to perform the STFTs would be made up for by the time you would save by using an automated mixing system such as this one. However, it plays a role in the latency that this program would introduce if being used for live mixing. For a 1 second long section of the random song

mentioned above using the Google Colab architecture mentioned in the work environment chapter, the STFT with FFT and window sizes of 8192 and hop size of 512 took approximately 178 ms, whereas the STFT with FFT and window sizes of 64, and hop size of 16 took approximately 84 seconds. This is a marked difference and for this and other reasons mentioned above, 8192 was originally chosen as both the FFT and window size and the hop size was set at 512. For the sake of comparison however, and due to the fact that large amounts of RAM would be required to implement a densely connected 4096 band equaliser network (there are half as many bands as the FFT size), the experiments were also run with a 32 band equaliser network. In this network the FFT and window sizes were 64 and the hop size was 16. This allowed for the comparison of the custom structures with deeper structures.

At a sampling rate of 44.1kHz, an FFT size of 8192 is equivalent to taking roughly 185 milli second slices of each stem while an FFT size of 64 is equivalent to taking 1.45 milli second slices of each stem. After the STFT is applied, the frequency information for each stem and corresponding mix was added to multiple $h5$ files as input and output pairs to be used in the training of the equaliser neural network. These $h5$ files are Hierarchical Data Format (HDF) files that store array like data. The h5py library was used to interact with these files. The HDF format was chosen for its structure and its ability to store large arrays without taking up as much space as other formats such as CSVs. The time domain representations of the stems and corresponding mixes were added to a separate $h5$ file as input-output pairs to be used in the training of the volume control neural network. The format of the input and output pairs is explained in the next sub-section.

4.1.1 Equaliser Input-Output Pair Formatting

To provide an understanding of how the data were split and used to create input-output pairs for the equaliser neural network, an example will be used with the help of figures 4.6 and 4.7. For a 3 minute song with three stems A, B and C and a sampling rate of 44.1kHz, there are 7 938 000 samples. The result of the STFT gives a number of spectrogram slices. Each slice is an individual spectrogram that contains the frequency content information (magnitude and phase) for each individual stem, as shown in figure 4.6. For each stem, A, B and C, as well as for the corresponding mix there are an equal number of slices. Each slice of each stem is a single row with either 4 096 or 32 columns depending on which model was being used. The value in each column refers to the magnitude of each particular frequency bin as a result of the application of the STFT. The network was trained using the magnitudes only, but the phase was used when performing the inverse Fourier transform in the loss function as explained in the Phase Concatenation for Loss Function section in 4.1.2.

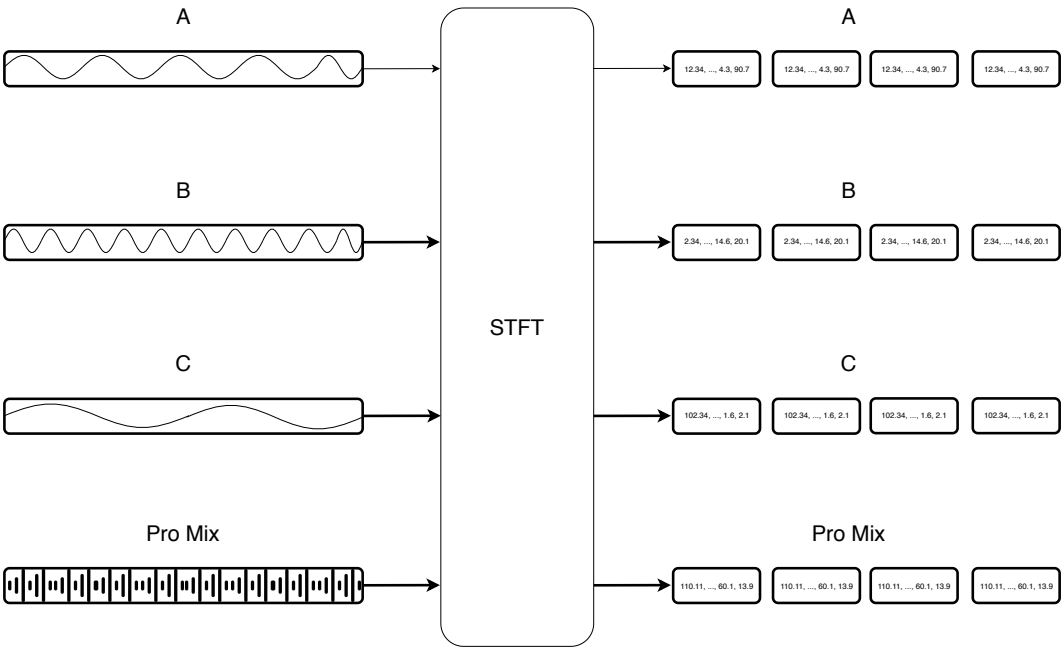


Figure 4.6: Input-output pairs for the equaliser network example

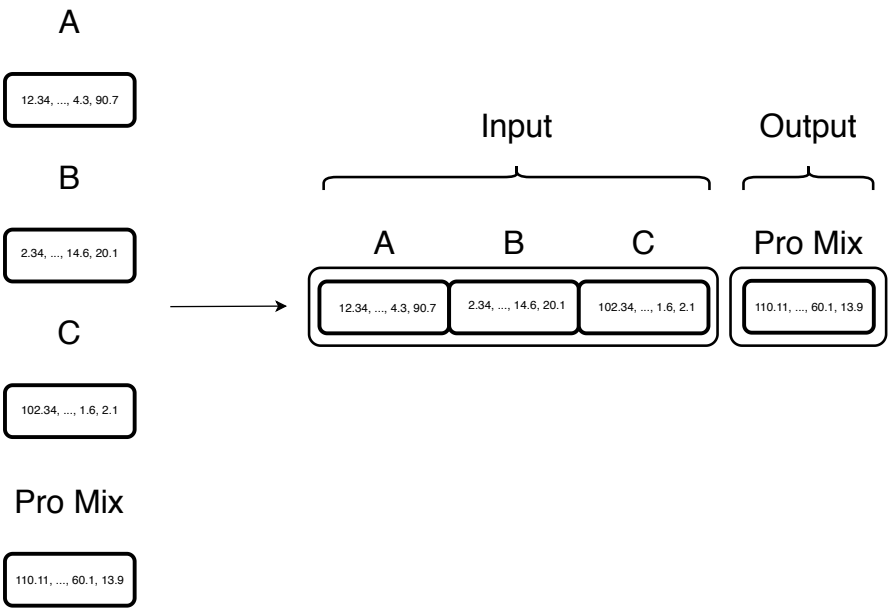


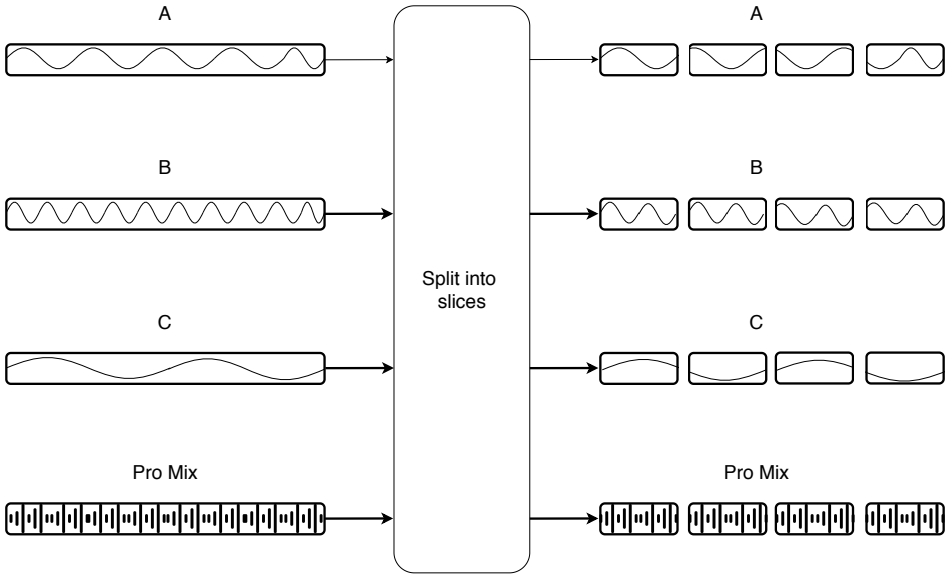
Figure 4.7: Input-output pairs for the equaliser network example

Each stem and the corresponding output were all rows with either 4 096 or 32 columns depending on the model. For this particular example, the resulting input-output pair would combine the stems as the input, and have the output mix as the output in the following way: input = [A, B, C] where A, B and C are all arrays with either 4 096 or 32 columns each depending on the model and where output is the output row with either 4 096 or 32 columns as shown in figure 4.7. Therefore the dimensions of the input for the 4096 band equaliser would be rows of 4 096 * 3 columns and the output would be rows of 4 096 columns. In the case of the 32 band model, the dimensions of the input would be rows of 32 * 3 columns and the output would be rows of 32 columns. The figures 4.6 - 4.7 above are to provide a visual explanation of this process.

4.1.2 Volume Control Input-Output Pair Formatting

For the volume model two neural networks were designed as with the equalisers. One that would accept 8192 samples and one that would accept 64 samples

so as to allow them to be used with the equaliser models where those are the dimensions of the outputs. Using the same example as above where the song consists of 7 938 000 samples, but this time using the waveform and not the STFT, each stem and the corresponding output mix was split into slices, each with either 8192 or 64 samples depending on the model. The slices were then used to create the input and output arrays where, as above, the input = [A, B, C] where A, B and C are all arrays with multiple rows of either 8 192 or 64 columns each and the output is the output mix array with rows of either 8 192 or 64 columns. Therefore, in this example, the dimensions of the input for the 8192 sample model would be rows of 8 192 * 3 columns and the output would be rows of 8 192 columns. The dimensions of the input for the 64 sample model would be rows of 64 * 3 columns and the output would be rows of 64 columns.



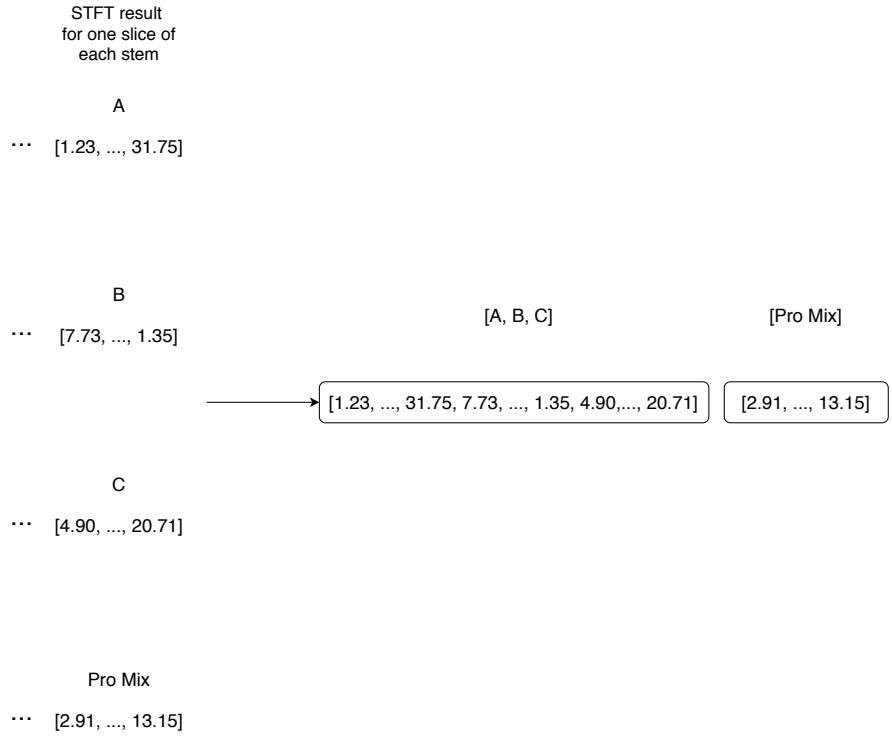


Figure 4.8: Input-output pairs for volume control network example

In both situations mentioned above, the sum of stems A, B and C before performing the STFT would give an unmixed version of the final song either in the frequency domain for the equaliser neural network, or in the time domain for the volume control neural network. This is useful when comparing how the neural networks perform because it allows for the output of the neural networks to not only be compared with the professional mix, but also with the completely unmixed sum of the stems. This is done in order to assess whether the neural networks attempt at mixing the songs is better or worse than no mixing at all. This was done using the output of the neural network and the unmixed output and comparing them to the professional mix and then assessing the loss metrics from these outputs. If the loss is less for the neural network's mixed outputs than for the unmixed output, then the neural network's mix is closer to the professional mix than the unmixed output. The lower the loss, the closer the neural network's mix is to the professional mix.

Phase and Magnitude Concatenation for Loss Function

The output of both of the neural networks are gains. In the equaliser's case they are frequency bin gains and in the volume control network's case they are track gains. Given that the outputs are gains and that the comparison required is not between the professional mix's gains (because these were unavailable) but rather the actual spectrograms and waveforms, it was important to be able to reconstruct the spectrograms and waveforms of the neural network's mixes in the loss function so that they could be compared with those of the professional mixes. In order to do this for the equaliser network, the network's output gains, the original input magnitudes and phase data as well as the phase data for the professional mix was required. For the volume control network, the output gains as well as the input waveforms were required.

In the case of the equaliser, the original magnitudes of each track were multiplied by their corresponding output gains from the network. The result of that calculation, along with the original phase data for each track, was used to synthesize the track's mixed waveforms. Once each track's mixed waveform was synthesized they would all be summed and that sum would be the neural network's final mix, which could then be compared with the professional mix.

In the case of the volume control network, the original waveforms of each track were multiplied by their corresponding output gains from the network. The result of which produced, again, the mixed track waveforms which were then summed to produce the neural networks final mix, which was then compared to the professional mix.

Given that this additional information was required for the reconstruction, it needed to be passed to the output layer so that it would be accessible in the loss functions. In order to pass this additional information to only the output layer but avoid it being used in the training process, Keras' layer concatenation functionality was used. At the output layer the neural network's actual gain outputs were concatenated with the additional information needed by each of the networks. Greater detail about how the loss function actually carried out these comparisons can be found in chapter 5.1.1.

Chapter 5

Methodology

The goal was to determine the best combination of weights for the equaliser and volume control neural networks through training on the MedleyDB dataset, then apply them to slices of each individual track in order to produce a final mixed song. Given that there were two neural networks each with its own responsibility (EQ and volume control), the question of ordering the volume and EQ networks within the system arises. By placing the equaliser first as shown in Figure 5.1, the system could suppress unwanted frequencies before ever applying any gains which means that only desired frequencies would be amplified [Pestana et al., 2014]. For this reason, the system shown in 5.1 was chosen. It is mentioned in the Recommendations for Future Work (chapter 7), that it would be interesting to see if swapping the order of the networks would improve or worsen the performance of the system.

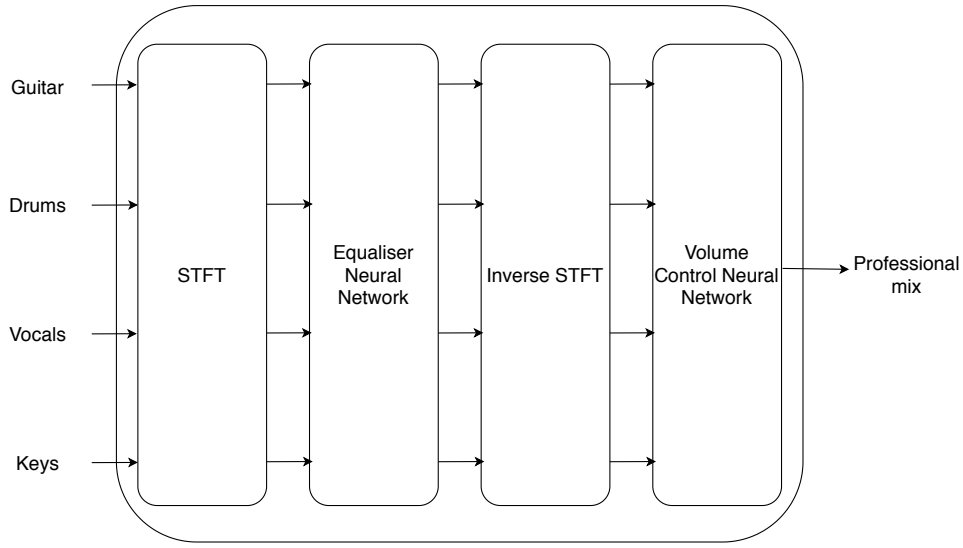


Figure 5.1: System Model - EQ first

5.1 Work Environment

All the code for this project was written in Python 3 using the Keras and Tensorflow libraries. Google Colab was used. Google Colab is essentially a Jupyter notebook stored in a Google Drive that is run on a customisable cloud-based run-time with certain hardware accelerators. For this project, the Python 3 environment with the GPU hardware accelerator and 32 GB of RAM was used.

Given that the full dataset for the equaliser network alone contains more than 1 TB of data to train on, the dataset had to be split into manageable batches, and the networks trained on each batch. Instead of relying on automated batch shuffling which could construct a batch of data that doesn't contain tracks with instruments representative of the entire dataset, manual batching was done. This was to ensure uniform sampling from the entire dataset and that each batch was sufficiently representative of the whole dataset. This was achieved by ensuring each batch consisted of at least one slice (as explained in chapter

4.1) of each of the songs in the training portion of the dataset. This meant that in each batch the whole training portion of the dataset was represented.

Using these batches and streaming the dataset off the hard-drive, it was possible to circumvent the hardware requirement of loading the entire dataset into memory. Each network trained on the same training data from the dataset in the same order and trained for 20 epochs. Training the networks for longer than 20 epochs was found to cause over-fitting and so twenty was chosen as the cut off.

Activation Function

ReLu activation function was used for all the hidden layers in the dense and custom architectures. ReLu was chosen because it was used in a number of the papers reviewed in chapter 2.5, namely [Mimilakis et al., 2016] and [Nercessian, 2020] and because in [Goodfellow et al., 2016] it is mentioned that the default choice for deep networks should be ReLu. The alpha value used was $\alpha = 0.1$ for all the networks. The individual networks had varying thresholds and maximum values which are discussed in section 5.2.

5.1.1 Loss Functions

For all neural networks mentioned in this project the loss functions were custom made. Experimentation showed that the Stochastic Gradient Descent (SGD) optimiser obtained lower losses when compared with other optimisers, therefore SGD was used as the optimiser with a learning rate of 0.00001.

Equaliser

For the equaliser network, the output of the network was the gains that should be applied to the various frequencies of the various stems. To be able to use these to compare the network's mix against the professional mix, the network's mix needed to be constructed in the loss function. As mentioned in chapter 4.1.1, the original magnitude and phase data for each stem was concatenated with the actual input to the network and only passed to the final output layer. This meant that in the loss function, the magnitude and phase data as well as the actual frequency gain neural network outputs were available.

Figure 5.2 shows exactly how the data was concatenated and how the loss function takes in all the data needed and compares the final mixes.

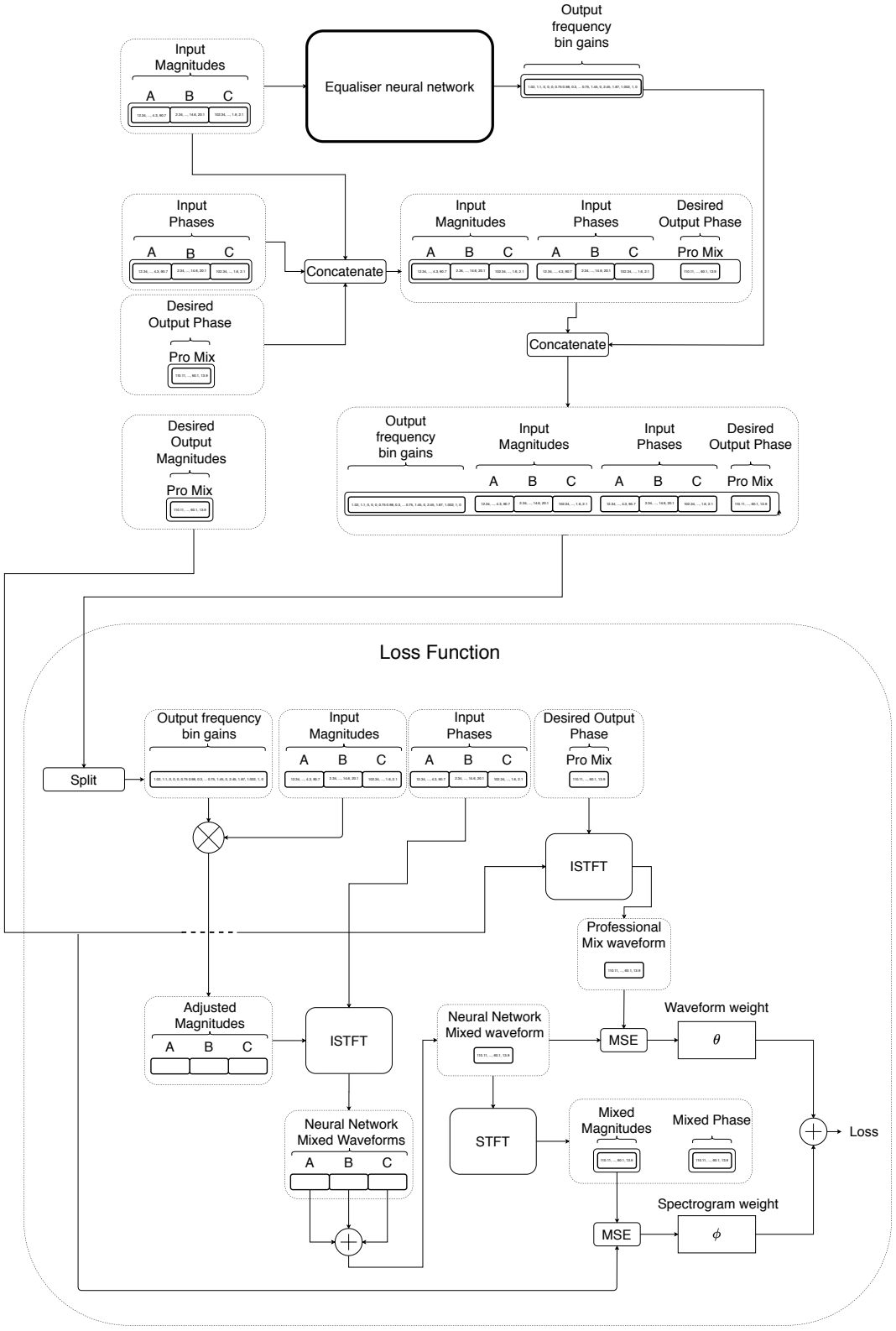


Figure 5.2: Equaliser Loss function explanation

In the loss function the magnitude data for each frequency of each stem was

multiplied by the output frequency gain for that stem at that particular frequency. The result was a set of magnitude adjusted stems. These were then used along with the corresponding phase information to synthesize the mixed stem waveforms which were then summed to produce the network's final mix.

This final mix waveform was then compared to the professional mix waveform using the mean squared error (MSE). This gave what will be referred to as the waveform loss (l_{wf}). Additionally, the mixes were then also both transformed once again into the frequency domain and the spectrograms were compared using MSE. This gave what will be referred to as the spectrogram loss (l_{sp}). The EQ network was trained to minimise the loss described by the following function:

$$loss = \theta l_{wf} + \phi l_{sp} \tag{5.1}$$

The coefficients θ and ϕ allow for the network's goal to be adjusted to either focus more on matching the professional mix's waveform (increase in θ) or the spectrogram (increase in ϕ). Various combinations of θ and ϕ were tested throughout the training procedure, and the rate at which the loss decreased as well as subjective assessments of the final mixes by the author were noted for each combination. Given these two observations the best combination tested has $\theta = 50000$ and $\phi = 1$. While 50000 may seem high it had to be so because the MSE, when comparing the spectrograms, was between four and five orders of magnitude larger than the MSE of the waveforms. In order to allow them to contribute equally to the loss, the waveform loss had to be multiplied by a large value so as to have it represented equally in the loss function. This approach where the loss function consists of multiple parts is similar to that

in [Scott et al., 2011].

Volume Control

For the volume control network, the output of the network was the gains that should be applied to the various stem's waveforms. To be able to use these, as described above, the final mix needed to be constructed in the loss function. As mentioned in chapter 4.1.2, the original waveforms were concatenated with the actual input to the network, ignored in the training process and only passed to the final output layer.

This meant that in the loss function the waveforms as well as the neural network's output gains for each stem were available.

Figure 5.3 shows exactly how the data was concatenated and how the loss function takes in all the data required and compares the final mixes.

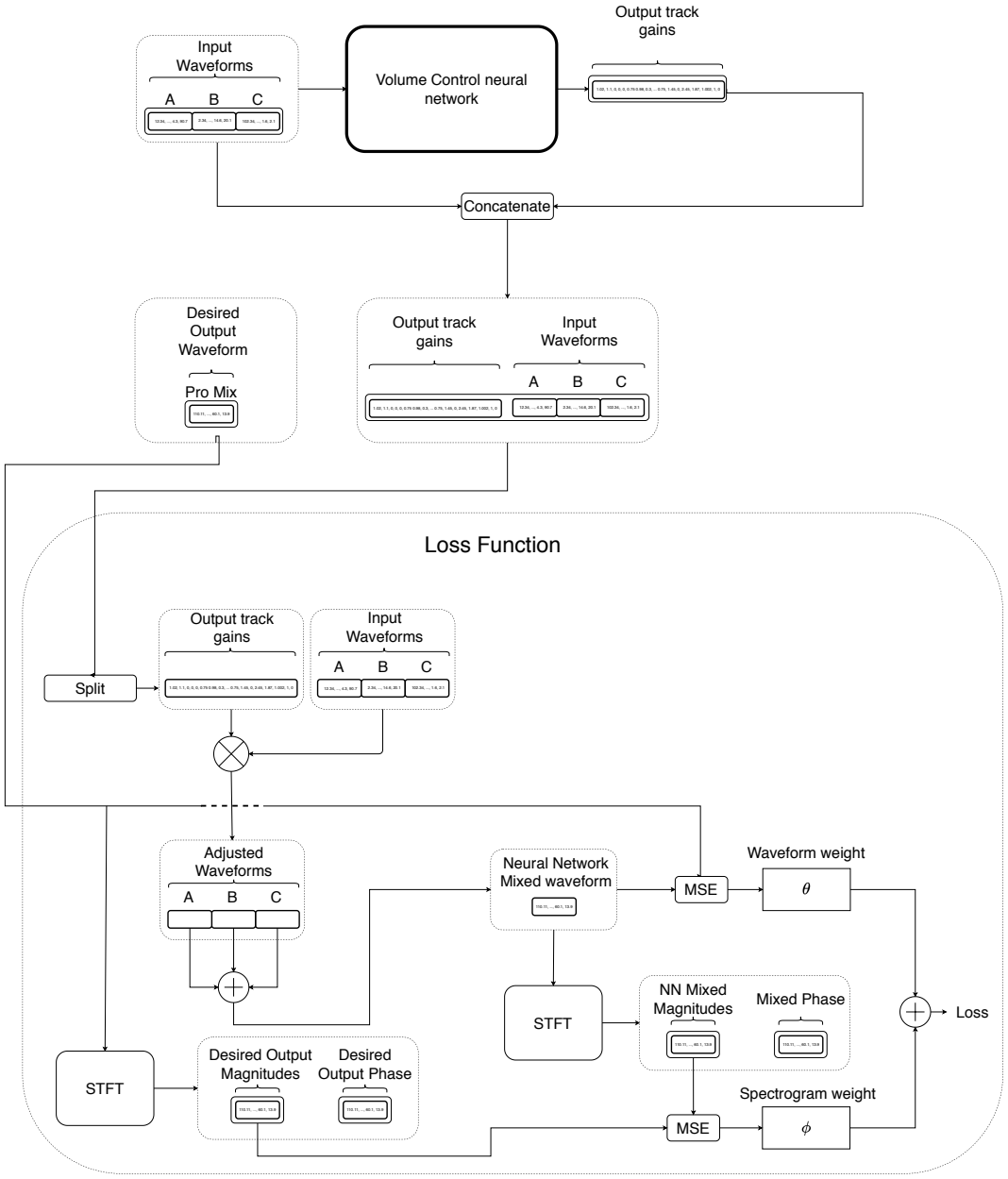


Figure 5.3: Volume Control Loss function explanation

In the loss function the waveforms of each stem were multiplied by the corresponding output gain. The result was a set of volume adjusted mixed stems. These were then summed to produce the network's final mix waveform.

As is done with the equaliser network, the waveform of the network's final mix is compared to the professional mix using MSE. The spectrograms are are also

compared using MSE. The loss function is then given by equation 5.1.

5.2 Implementation

In order to be able to test each different component of the mixer it was important to be able to take each effect that the mixer is comprised of and construct and test them individually.

Each of these individual effects was constructed with different objectives in mind, and therefore had different structures. When testing each effect it was taken into account that the effect being tested was most likely not the only effect that the professional mixers had used on any particular training example. This meant that it was not reasonable to expect the system to get zero loss/error because this would not be possible using only one effect when the original mix used multiple effects. The goal of the system in training was however to minimise the loss/error and get as close to the final professional mix as possible. This is analogous to a professional sound engineer being given the goal of mixing a song as competently as possible, while being limited to only the application of an equaliser, and the changing of the gains of the stems, for example.

This system would be considered a success **if the mixes produced by the networks were significantly closer to the professional mix than the unmixed sum of the stems**. This would be measured, as discussed above, by assessing the mean squared error of the final neural network mix and the unmixed sum of the stems when compared to the professional mix.

Each component of the mixer (EQ and volume control) and their structures are explained in the sections that follow. Once all the individual features had been constructed, trained and tested individually, they were combined and this combination formed the final mixer which was then also tested.

5.2.1 Equalisation

The goal of the equaliser was to take spectrogram slices of the song as inputs which contain the frequency information needed to do equalisation, and apply gains to each frequency band of each track. The gains applied to each frequency on a particular track would depend not only on the frequency information of that track at that time, but also on the frequency information of the other tracks producing input at the same time. This was done with the intended outcome being that if the equaliser neural network takes into account what frequencies are present in the mix from the other tracks, it would be able to minimise frequency masking and give each stem its own space in the mix.

In the MedleyDB dataset each song has the final mix (combination of all tracks with effects) and the individual stem tracks that made up the mix. The goal was to train the neural network so that given any group of stems and being told what instruments or sounds each of them were, a final professional quality mix could be produced.

Four architectures were tested and compared for the equaliser networks. The architectures used are listed below:

- 32 band, multi-layered, densely connected neural network (19 257 216

trainable parameters)

- 32 band, custom designed, musically informed neural network (72 897 trainable parameters)
- The same 32 band, custom designed, musically informed neural network as above, but with four additional densely connected layers (trainable parameters)
- 4096 band, multi-layered, densely connected neural network (24 860 975 trainable parameters)

A fully densely connected 4096 band model with 4096 nodes for each instrument was not feasible as it would have meant 192559 nodes per layer which, if seven layers were to be used for a fair comparison, would mean a total of 37079161040 trainable parameters. Each architecture is discussed below.

32 Band Multi-layered Densely Connected Architecture

In [Ramırez and Reiss, 2017] Martinez Ramirez and Reiss used Deep Auto Encoders (DAEs) with three hidden layers of 1024 nodes each. The frame and hop sizes used in [Ramırez and Reiss, 2017] were 2048 and 1024 respectively while for this deep, densely connected network, frame sizes of 64 and hop sizes of 16 were used. This was so that more hidden layers could be added without using excessive amounts of RAM. This model used seven hidden layers as in [Mimilakis et al., 2016] with as many nodes per layer as there were inputs to the system. The initial structure used in this paper can be seen in Figure 5.4

below.

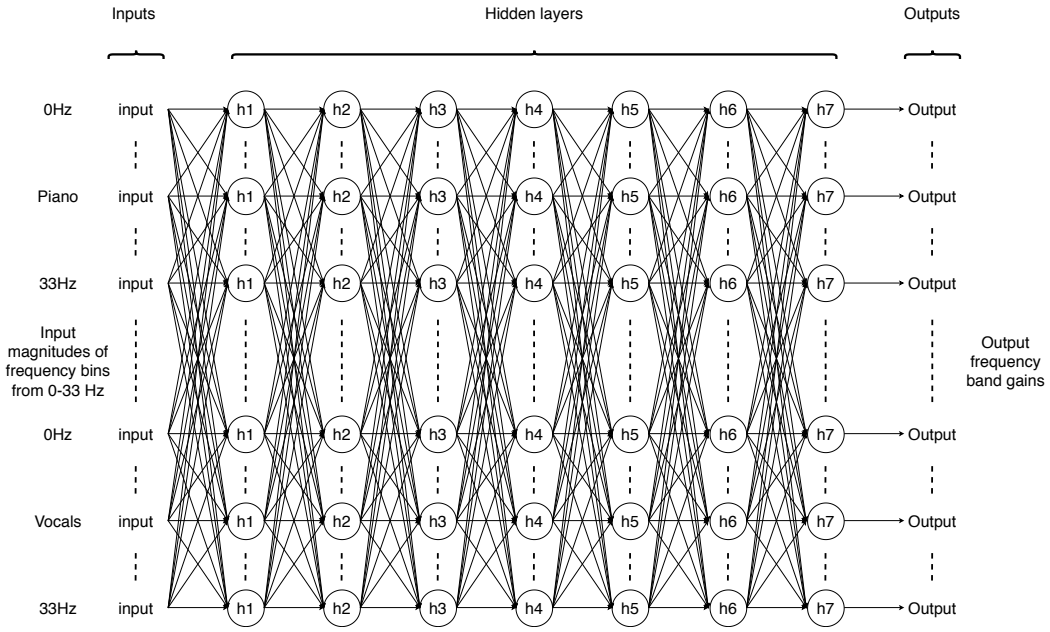


Figure 5.4: Simplified representation of the multi-layered densely connected equaliser model used in this paper

The densely connected architecture described above meant that the output gain for each frequency band would not only depend on the same frequency bands of the input tracks, but also on all other frequency bands of all the other input tracks. This means that, for example, the gain for the 30th band of the piano track in a particular song would not only be dependent on the magnitudes of the 30th bands of all the other instruments, but also on the magnitudes of every single band of every single track including itself. Considering frequency masking only happens when the same frequency bands of different instruments interfere with each other. This amount of interdependence seems unnecessary but given the past success of deep densely connected neural networks in the papers reviewed in chapter 2, this approach was one that had to be tested.

This method gives the neural network complete control over how it weighs the contributions of each input frequency band to each of the final output frequency gains.

32 Band Custom Musically Informed Architecture

In the previous architecture the output gain for each instrument frequency bin is dependent on the magnitude of every other instrument. This means that the low frequencies of the bass, for example, would be affected by the high, mid and low frequencies of every instrument including itself. While this seems to allow for greater control, it does not make much sense musically. Frequency masking occurs when similar frequency ranges of different instruments interfere with each other.

For this reason, and because of the author's own mixing experience, a new network was designed that took this into consideration. This neural network was designed while keeping in mind the general practice of mix engineers. When mixing music, mix engineers must ensure that stems with similar frequency content are given their own space in the mix, and that sounds that occupy the same ranges of frequencies do not interfere with or mask each other. This is the crux of mixing musical compositions. In order to achieve this, engineers will listen to the song they are mixing and go through the different frequency ranges to identify which stems are most present in each, then adjust them through an equaliser if need be.

A model was designed that ensured that, for a chosen instrument, any particular frequency bin gain at the output depended only on the magnitudes of

that same frequency bin from all of the tracks in the mix. This meant that the low frequencies of the bass would only be affected by the low frequencies of itself and the other instruments (to continue the example above).

A simplified representation of the architecture is shown in Figure 5.5.

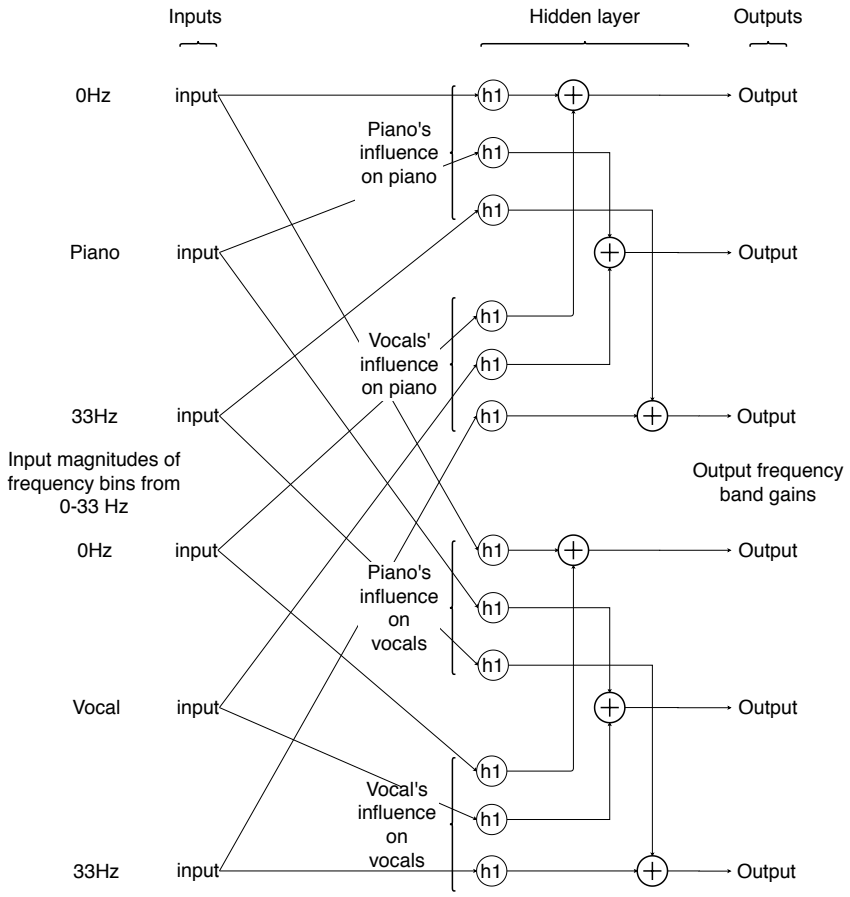


Figure 5.5: Simplified representation of the 32 band musically informed custom architecture equaliser model used in this paper

As shown in Figure 5.5, This neural network's hidden layer has far more nodes than the previous architecture. The previous architecture has $33 * 47$ nodes in each hidden layer whereas this architecture has $33 * 47 * 47$. The reason for this was so that each output gain for each instrument could depend only on the corresponding input magnitudes for all the other instruments. To illustrate this concept and explain why this many nodes were required, consider the fol-

lowing example. In Figure 5.5, if the Figure is taken literally and the implied missing frequencies are ignored for the sake of this example, then instead of 32 bands, there are only three bands and instead of 47 instruments there are only two instruments (piano and vocals). The first output band gain for the piano instrument is dependent only on the weighted sum of the magnitude of the piano's first band and the magnitude of the vocal's first band. This example only has two instruments and already the first output band gain for the piano alone is dependent on two input magnitudes. To then extend this logic to the full 47 instruments, the first output band gain of the piano only would be dependent on all of the 47 first band input magnitudes and similarly the piano's second output band gain would be dependent on all of the 47 second band input magnitudes. This applies to all 32 bands of all 47 instruments in this network, therefore requiring this network to have $33 * 47 * 47 = 72897$ nodes, where the network in the simplified example in Figure 5.5 required $3*2*2 = 12$ nodes.

This network, while having far more nodes per hidden layer than the previous example, only has one hidden layer. As the Figure shows there is only one hidden layer which is responsible for transforming the magnitudes at the inputs into gains. While this network is musically informed which was considered an advantage, its simplicity may be a disadvantage. It was a concern that the network would not be able to manipulate its weights in such a way that would produce satisfactory results.

32 Band Custom Musically Informed Architecture with Subsequent Dense Layers

In order to find a middle ground between the densely connected architecture and the musically informed architecture in Figure 5.5, this architecture was designed. This architecture aimed to build on the rationale of the musically informed architecture while taking advantage of the strengths of deep neural networks.

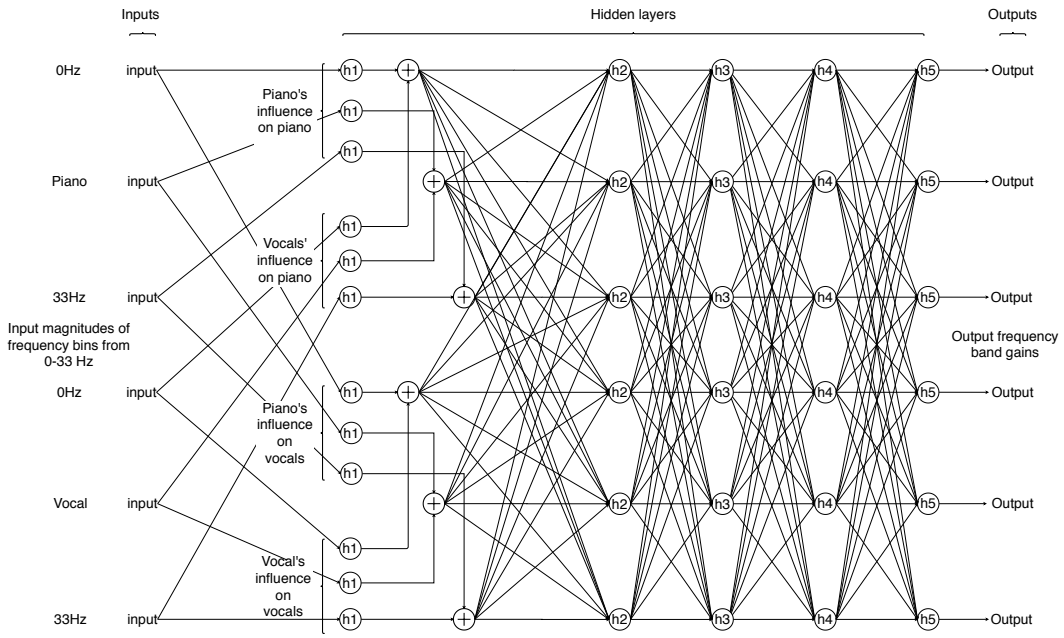


Figure 5.6: Simplified representation of the 32 band musically informed custom architecture equaliser model used in this paper with extra dense layers

As shown in Figure 5.6, this architecture starts in exactly the same way as the custom musically informed architecture in Figure 5.5 however what was previously the output in Figure 5.5 becomes the input to a set of four densely connected hidden layers.

4096 Band Multi-layered Densely Connected Architecture

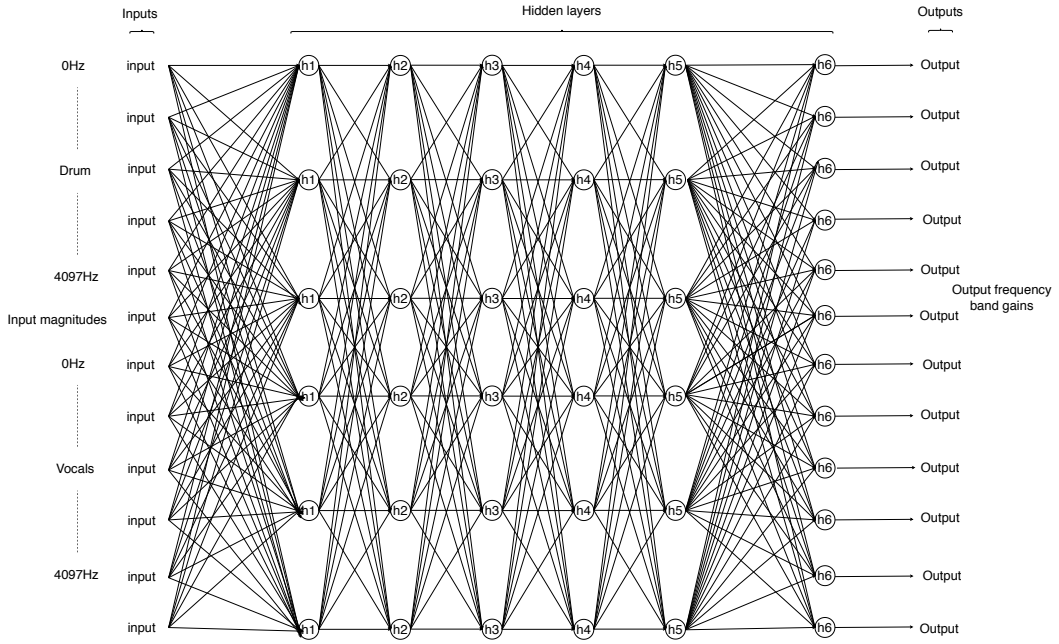


Figure 5.7: Simplified representation of the 4096 band equaliser model used in this paper

As mentioned above, to implement a 4096 band densely connected neural network with as many nodes per layer as the inputs to the system, for a seven hidden layer network to be compared to that of the 32 band model, there would have been 37 079 161 040 trainable parameters which is just not feasible. For this reason, this model needed to be limited in the number of nodes per hidden layer. The architecture that was created took in the 192 559 inputs. The first hidden layer and the next four layers all had 64 nodes each, and then the output layer had 192 559 nodes. These formed the output gains for each band of each of the 47 instruments.

This network has 24 865 135 trainable parameters and allows for very granular controller over the frequencies which was originally thought to be a good thing. The model has each frequency band at the output depending on a large

number of other frequencies from the other instruments so it is expected that identifying and fixing frequency masking will at least be possible.

Constraints

While none of the equaliser networks' weights were constrained, the *threshold* and *max_value* parameters for the ReLu activation's used for each of the networks were varied throughout the experimentation process. The values that performed best for the equaliser networks were $threshold = -2$ and $max_value = 2.5$ for all hidden layers. The threshold being negative allows the presence of frequencies in one stem to reduce the presence of similar frequencies in another stem. This was key for the equaliser networks as this is what allows for the prevention of **frequency masking**. Having the threshold at or above zero for the equaliser networks hindered the neural networks' performance significantly. This was a promising indication that the equaliser networks may be successfully handling frequency masking. This is discussed more in the results section. The output layer then had a ReLu activation with $threshold = 0$ and $max_value = 2.5$, just so that the actual gains at the output were never negative

5.2.2 Volume and Gain Control

Volume control of stems is something that is commonly used by mix engineers and is of fundamental importance. Recording artists and musicians often record the individual stems in such a way that they are not at a consistent volume level. Their concentration is often on producing stems that sound the way they find pleasing, and not on volume. This can result in various combinations of sounds at different volumes when the stems are sent to the mixing engineer. For this reason, it is important for the mixing engineer to adjust the levels

of the stems in order to allow everything to be heard as it should be without exceeding recommended volume levels or having sounds that are so soft that they are difficult to hear. As discussed in chapter 1.2.1, mixing engineers went through a stage called the "loudness wars" starting in the 1960s ,where mixing engineers competed to see who could produce the loudest mixes without going over 0dB. 0dB is the reference level that relates to the volume at which anything higher is not able to be reproduced without causing what is known as clipping. Clipping is to be avoided at all costs because it causes distortion and a loss of information which results in reduced sound quality[Kendrick et al., 2015].

Volume control is done by applying a gain to a signal. This gain is measured in dB where a positive gain means making the sound louder and a negative gain means making it softer. For this project, the gain that was applied was achieved by reading the sound in as a waveform array and multiplying all the values in the array by a given number where a number above 1 meant making the sound louder, and a number between 0 and 1 meant making it softer. If multiplied by 0 the sound would be eliminated but it is unlikely that this would ever be done.

The goal of the volume control network was to take waveform slices of the song as inputs and apply gains to each track. The gains applied to each track would depend not only on that track's waveform, but also on the waveforms of the other tracks producing input at the same time.

Two architectures were tested and compared for the volume control networks. The first was a multi-layered, densely connected neural network taking in wave-

forms 64 samples long. The second was also a multi-layered, densely connected neural network, but this time taking in waveforms 8192 samples long. The reason for the choice of these two architectures was so that the equalised outputs of the 32 and 4096 band equaliser neural networks could be used as the inputs for these 64 and 8192 sample volume control networks.

64 Sample Multi-layered Densely Connected Neural Network

In order to perform volume control on the results of the 32 band equaliser neural network, this network was designed considering that the mixes produced by the 32 band equaliser network would be 64 sample waveforms. There were 47 instruments so the input dimensions of this network were $47 * 64 = 3008$. The network consisted of seven densely connected hidden layers each with 3008 nodes per layer. The output of the network was the 47 different track gains. The architecture is shown in Figure 5.8.

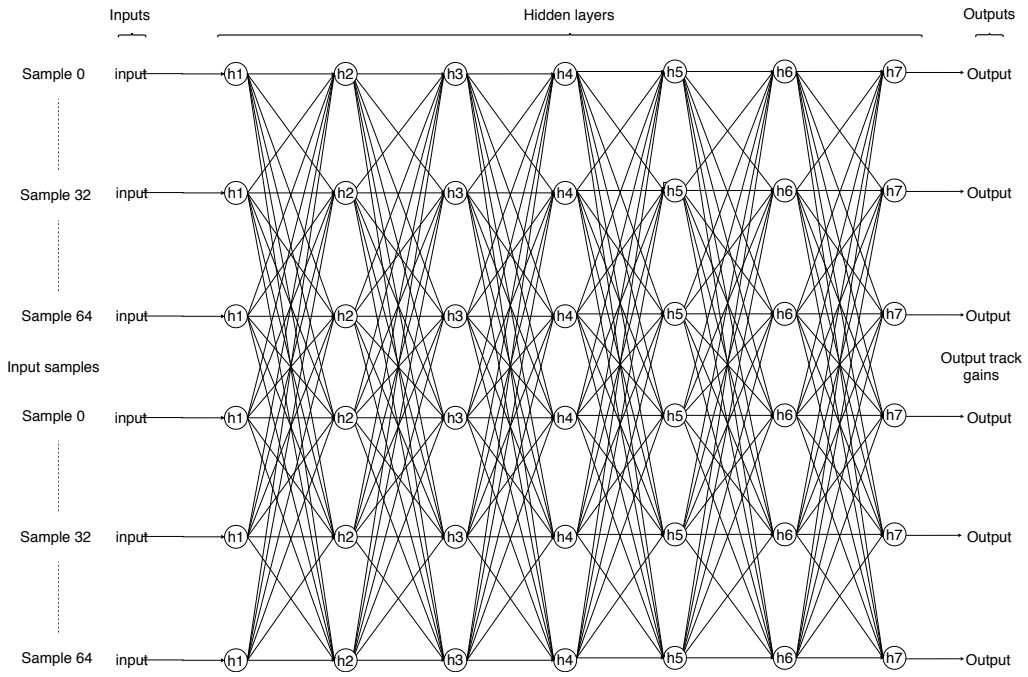


Figure 5.8: Simplified representation of the densely connected 64 sample volume control model used in this paper

This architecture was designed to be used on the waveforms produced by synthesizing the equalised tracks produced by multiplying the 32 band equaliser gains by the magnitude spectra of the input tracks. The reason for it accepting 64 samples was because that is how many samples would be in the resulting synthesized tracks. This is a simple fully densely connected deep neural network which has shown time and time again to work well in the past.

8192 Sample Multi-layered Densely Connected Neural Network

In order to perform volume control on the results of the 4096 band equaliser neural network, this network was designed considering that the mixes produced by that network would be 8192 sample waveforms. There were 47 instruments so the input dimensions for this network were originally going to be $47 * 8192 = 385024$. This network however was impractical due to the number of trainable parameters it would have had, so some adjustments had to be made.

For this network, a similar approach to that of [Scott et al., 2011] was used where some form of feature extraction was done prior to sending the data to the neural network. For this model the feature extracted was the normalised mean of the absolute values of the waveforms. This was used to attempt to get a value that would represent how loud a particular track was compared to another. If one track was louder than another, the mean of the absolute value of the waveform of the louder track would be higher than the softer one. When normalised between 0 and 1 the louder track would have a value closer to 1 than the softer track.

These normalised track metrics were then passed to the neural network. The

use of these metrics meant that the network required significantly fewer nodes per layer, and could therefore be much deeper. Only 47 nodes were required per layer. The architecture of this network is shown in Figure 5.9

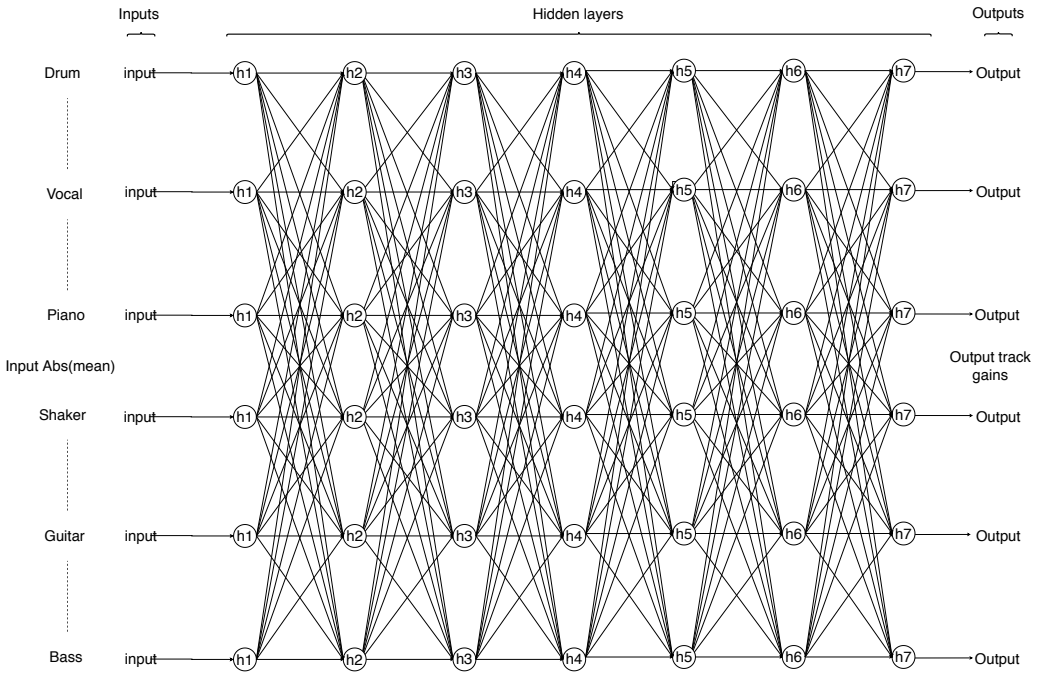


Figure 5.9: Simplified representation of the densely connected 8192 sample volume control model used in this paper

This network had seven hidden layers with 64 nodes in each layer. The output layer had 47 nodes and represented the output track gains which were to be applied to each of the input tracks.

Constraints

As with the equalisers, none of the volume control networks' weights were constrained. The *threshold* and *max_value* parameters for the ReLu activation's used for each of the volume control networks were varied throughout the experimentation process. The values that performed best for the volume control networks were $threshold = -0.2$ and $max_value = 1.5$ for all hidden

and output layers. The reason a negative threshold was allowed was so that it would be possible for the network to produce negative gains which would imply flipping the phase of the stem. Phase flipping or phase adjustment in general is common in music mixing. As discussed in the literature review and in [Perez Gonzalez and Reiss, 2008], phase adjustments are often made when the same sound is recorded by multiple microphones placed different distances from the source. The reason the phase of the sound may need to be adjusted is to reduce phase cancellation which can happen when the same sound recorded by one microphone or input is out of phase with the same sound recorded by another microphone or input.

Figures 5.10 and 5.11 below illustrate phase cancellation, how it occurs, and how to deal with it. In this example, the sound being recorded is a pure sine wave $sin(2\pi fx)$ where x is time, f is the frequency of the wave (2000Hz in this example) and the sine wave sound starts playing at $x = 0$. The wavelength is determined by

$$\lambda = \frac{c}{f}$$

In this example c is the speed of sound (343 m/s) and f is 2000 Hz, making the wavelength $\lambda = 0.1715\ m$. The orange wave is the sine wave as recorded by microphone A which is right up against the source of the sine wave and therefore there is no delay. The blue wave is the same sine wave recorded by microphone B situated across the room, exactly $\lambda/2\ m$ away from the source of the sound, for example. The sine wave reaches microphone B exactly 0.00025 seconds after it reaches microphone A because microphone B is further away and the sound takes time to travel.

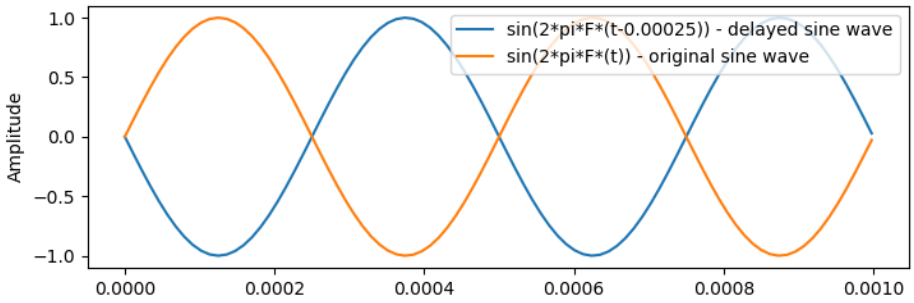


Figure 5.10: Phase cancellation example

As shown in Figure 5.10 the waveforms are identical but because of the 0.00025 second delay between microphone A and B, which causes a 180 degree phase shift, if these two sounds were to be added at the mixer, the result would be zero. This is because the two sounds are perfectly out of phase. In [Perez Gonzalez and Reiss, 2008] this is remedied by choosing a reference signal to be the signal that appears at the mixer first (in this case the orange wave), then taking the impulse response of it and all the other signals (in this case the only other signal is the blue wave), comparing the peaks and then determining the delay of the other signals. The reference signal and all other signals are then delayed accordingly to ensure that all the signals are in phase. The solution mentioned above achieves the desired result but it requires a lot of precision, and in most cases this level of precision is not required. The reason for this is that there is another solution to this problem that is simpler and works well in most cases where phase cancellation occurs. This simpler solution is to reverse (or flip) the phase of either one of the signals but not both.

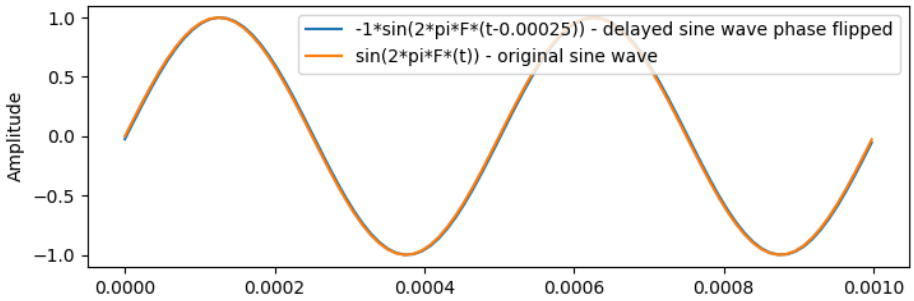


Figure 5.11: Phase cancellation solution example

Flipping the phase is the same as multiplying the sound wave by negative 1. As can be seen in Figure 5.11, this produces sounds that are in-phase and removes the phase cancellation. Flipping the phase however does not always produce perfectly in-phase signals, but in cases where phase cancellation is occurring, it will reduce the phase cancellation even if the resulting sounds are not perfectly in phase.

The threshold being -1 meant that the neural network was not prevented from doing phase flipping, however just producing negative gains is not sufficient proof that the neural network has actually learnt to identify and handle phase cancellation. The dataset specifies which songs contain stems with bleed meaning it is possible to test and be sure whether the neural network can identify and deal with phase cancellation. For this to be true the network would have to produce negative gains only for those tracks in the songs that have bleed. These tests are discussed in the results section.

Chapter 6

Results

To assess the mixing abilities of the equaliser and volume control neural networks, spectrograms and waveforms of the neural network mixes and the professional mixes were compared. To be able to evaluate the performance of the two networks it was important to get a baseline 'unmixed' result to have something, other than the professional mix, with which to compare the mixed result.

An unmixed result is obtained by simply summing all the stems of a particular song. This represents what the mixing engineer would have before applying any effects or making changes to any volumes. The mixed output of the neural networks is also compared to the professionally mixed output provided by the dataset. In addition to the unmixed and professionally mixed outputs, white noise was also generated and used to compare the outputs. The white noise was used as a representation of the worst-case scenario in which the idea was to mirror a situation where the neural networks completely remove any sense of musicality from the song and produce only noise.

In the training procedure multiple metrics were used to compare the perfor-

mance of the neural networks on the test set with the professionally mixed result namely:

- Loss
- Mean squared error
- R2 score

The metrics that were used most often in the neural networks cost functions were the mean squared error and the loss. These metrics were used to compare the training progress of both the equaliser and volume control neural networks.

In the testing stage the metric used to compare neural network mixes and professional mixes was the mean squared error metric as explained by equation 5.1. All results presented in this section, excluding the loss graphs which come from the training period, are for the test set and were obtained after the networks had been trained and combined.

6.1 Equaliser

The following table shows how the equaliser networks performed on the test set by comparing the unmixed sum of stems, the neural network mixes and the professional mixes for each network architecture. All values are the MSE values produced by equation 5.1.

For the sake of spacing, the network architectures have been given aliases.

Network Architecture	**Alias**
32 Band Multi-layered Densely Connected Architecture	32Dns
32 Band Custom Musically Informed Architecture	32MI
32 Band Custom Musically Informed Architecture With Subsequent Dense Layers	32MIDns
4096 Band Multi-layered Densely Connected Architecture	4096Dns

Table 6.1: Equaliser network aliases

Song	**Unmixed**	**32Dns**	**32MI**	**32MIDns**	**4096Dns**
The Silver Sea	470	503	461	449	1867
Keep Me Running	192	261	76	12	60
The Alchemist	1764	4204	3048	2056	4539
Chicken Fried Steak	275	400	397	201	912
Honey Brown	524	1106	523	239	1198
Return To Venezia	12	69	24	6	55
Mendelssohn Piano Trio 1 Movement 2	41	73	33	7	68

I'd Like To Know	2739	4818	3367	2364	4769
Improvisation 3	7	6	8	1	14
Reign Check	901	1469	805	346	1703
Prisoners Cinema	6232	3060	1067	198	5569
Miss Thing	1839	661	414	233	881
Schubert Movement	91	264	86	12	201
I Promise	1279	1117	904	795	1195
Perfect Day	2002	1908	1542	737	2881

Table 6.2: EQ MSE comparison results table

The results show that, overall, the 32 band deeper networks performed better than the more shallow 4096 band network, even though the frequency resolution with an FFT size of 64 was significantly lower than the 4096 model. In general, the musically informed models performed better than the purely densely connected models except in the case of the song **Keep Me Running** where the densely connected 4096 band model performed better than the 32 band musically informed model.

The best model by far is the 32 band musically informed model with additional dense layers. This model makes use of the strengths of deep neural networks while maintaining a musically informed structure. For this model, as explained in 5.2.1, frequency ranges in the different instruments can only affect

those same frequency ranges in each of the other instruments. This structure makes sense and mirrors what a mix engineer would do. Mix engineers will identify problematic frequency ranges where frequency masking is occurring and adjust those frequencies with EQ in order to minimise the masking and make sure both instruments can be heard and have space in the mix. Mix engineers would rarely, if ever, take into consideration the higher frequencies of a vocal when trying to fix frequency masking in the lower frequencies of the piano for example.

The 4096 band model performed the worst. There are multiple possible causes for its failure, the most interesting however being the very reason this model was originally included, the frequency resolution. With the network being able to change the magnitudes of so many bins it meant that each instrument's spectral identity could be very different from the original. It is similar to having a multitude of extremely narrow EQ bands. It is possible that because their frequency resolution was only 5 Hz per bin that the lack of consistency across sequential bands caused the synthesis to be sub optimal compared to the 32 band. The 32 band model affected bins of 689 Hz at a time thus giving the result a more consistent sound.

6.2 Spectrogram Comparison

In this section the spectrograms of the outputs of the different neural networks are compared to the professionally mixed output. An example from the test dataset is selected and discussed in Figures 6.1 - 6.5 below.

The professionally mixed output can be seen in Figure 6.1.

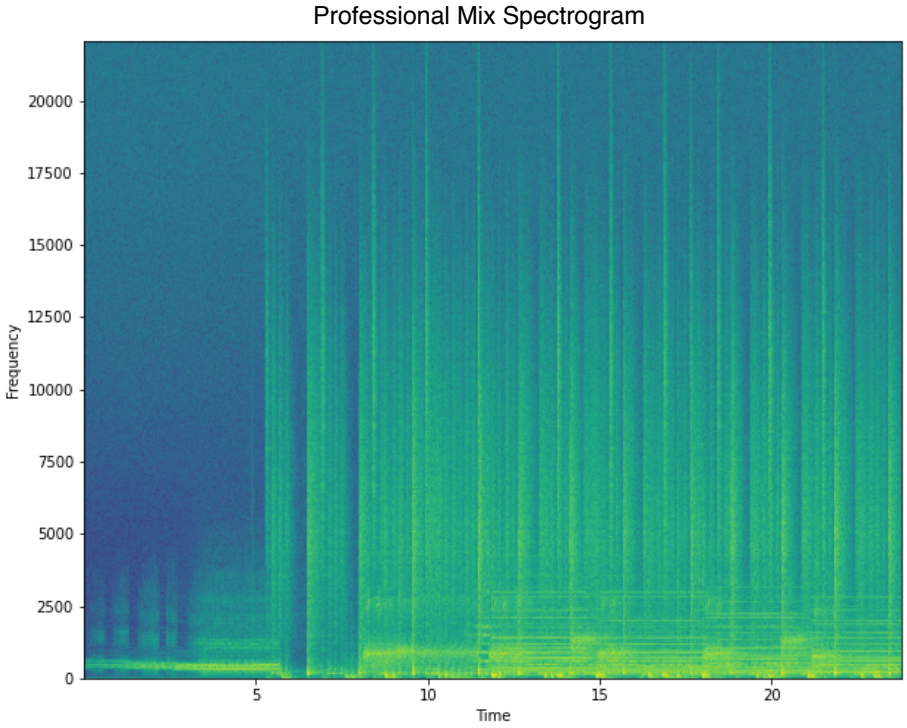


Figure 6.1: Output spectrogram of the professional mix

6.2.1 32 Band Multi-layered Densely Connected Architecture

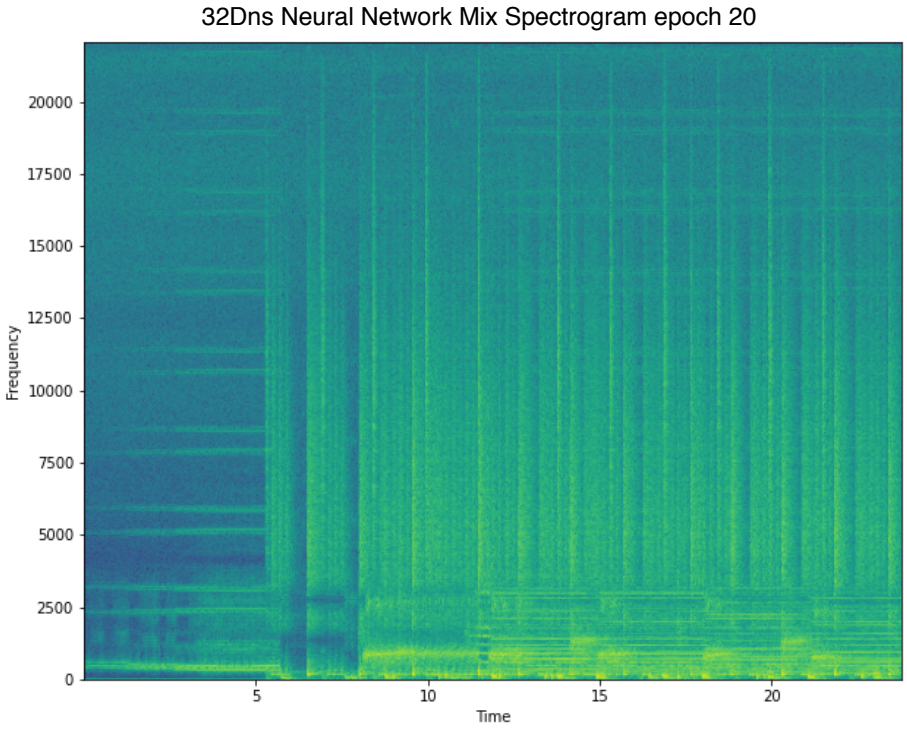


Figure 6.2: Output spectrogram of the 32Dns equaliser neural network on test input after 20 epochs of training

The spectrogram produced by the densely connected 32 band model is very interesting. There are two main points of interest that can be seen in Figure 6.2. The first is the duplicate frequency information that can be seen throughout the duration of the song. This is most obvious in the first 5 seconds of the song but seems to be present throughout to a certain extent. This is a clear example of the negative effects of having the higher frequencies' magnitudes being dependant on the content of the lower frequencies. This behaviour did not happen in every song but it was a fairly frequent occurrence when assessing

the spectrograms of the test tracks. This behaviour is obviously not ideal.

The next point of interest is the large reduction of the low to mid frequencies in the 5-10 second period. There is a clear reduction of two ranges of frequencies: The first from approximately 1200 Hz and the second from approximately 2800Hz. Due to the structure of the network the reason for the harsh cutting of these frequencies at this point could be related to the presence of any other instrument that gets introduced at that point. In the song the drum and bass instruments are added at around the six second mark. These would contain very strong low frequencies and it is possible that the network has produced EQs for the other instruments that cut the previously mentioned frequency ranges when the bass and kick instruments are present. This type of dynamic frequency band cutting is common in mix engineering but would hardly ever be done so harshly. This extreme cutting of frequencies at the introduction of the drum and bass instruments is quite clear when you listen to the mix. There are clearly defined spaces for the drum and bass instruments and it sounds like an overdone dynamic EQ that cuts the other instruments when the drums or bass are present.

The rest of the song for the most part looks fairly similar to the professional mix except for the general haziness in the 5000-12500 Hz range and the higher frequency duplicates as mentioned before.

6.2.2 32 Band Custom Musically Informed Architecture

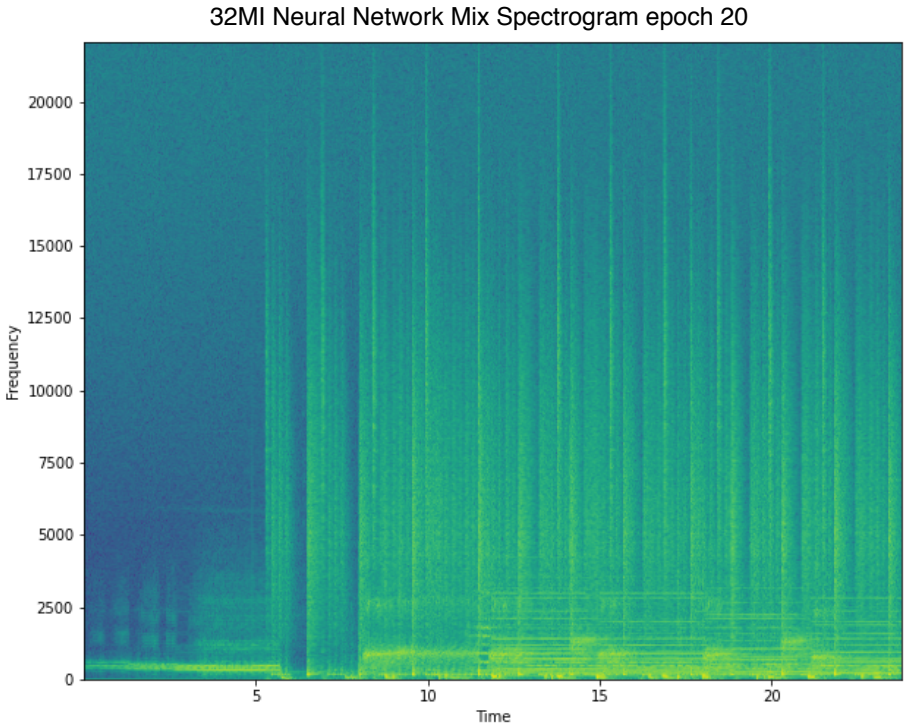


Figure 6.3: Output spectrogram of the 32MI equaliser neural network on test input after 20 epochs of training

The musically informed model produces a spectrogram much closer to that of the professional mix with none of the higher frequency duplicates present in the densely connected model. This architecture prevents that kind of duplicate because the presence of low frequency content in one instrument cannot affect the gains given to the higher frequencies of the other instruments. In general, the most important thing to note here is that there seems to be more intensity in the higher frequencies than in the professional mix. This is easiest to see in the very top left corner of both the professional spectrogram in Figure 6.1 and the neural network's mix in Figure 6.3. In Figure 6.3 that portion is more

yellow which means higher intensity. On the other hand, the lower frequencies in between the harmonics in the neural network's mix (0-2500 Hz) seem to have a lower intensity. This can be seen most easily in the first 2.5 seconds of the song.

The only other difference is the faint line just above 5000 Hz that is present in the neural network's mix but not in the professional mix. The mix engineer obviously had a strong low pass filter that was used on whatever instrument produced that frequency that this model did not cut. The differences are minuscule but interesting nonetheless.

6.2.3 32 Band Custom Musically Informed Architecture with Subsequent Dense Layers

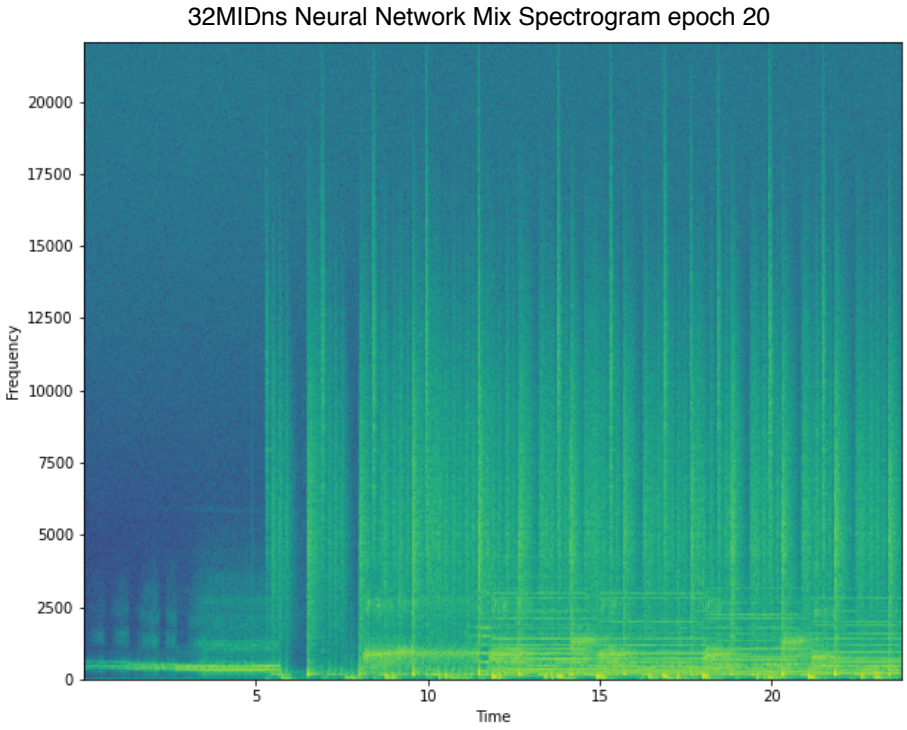


Figure 6.4: Output spectrogram of the 32MID equaliser neural network on test input after 20 epochs of training

This model performed the best, and has the spectrogram that most resembles the professional mix although the difference between this and the musically informed model in Figure 6.3 is very small. The only difference is similar to that mentioned above regarding the line just above the 500 Hz level. The other differences mentioned in 6.2.2 such as the general increase in intensity in the top left corner is no longer there in this mix, and the visible cutting in between harmonics is also no longer there. This model is extremely similar to the professional mix.

6.2.4 4096 Band Multi-layered Densely Connected Architecture

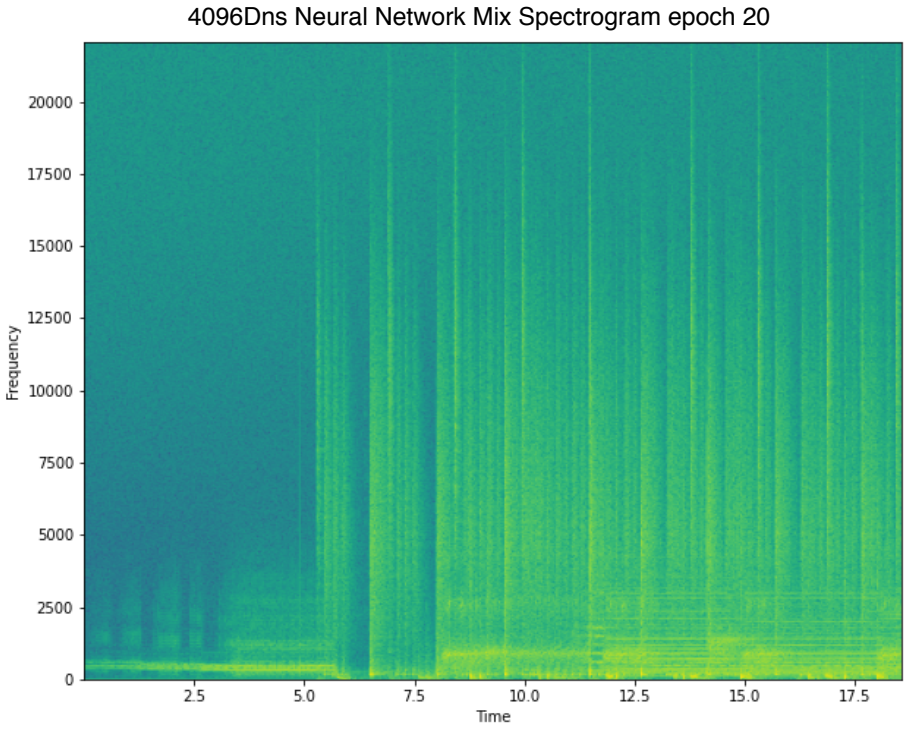


Figure 6.5: Output spectrogram of the 4096Dns equaliser neural network on test input after 20 epochs of training

This model performed the worst and by looking at the spectrogram it is clear why. The overall intensity is far greater than any of the others and there are far fewer clear spaces between harmonics. While the spectrogram still vaguely resembles the professional mix, it is clear that this model was not able to produce professional quality mixes. To listen this example and various others from the test set, and to see the code that created them visit this link: `https://drive.google.com/drive/folders/1pQ5rWE66H0MPUBq1j7XNrA09DJ8hBZNd?usp=sharing`.

6.3 EQ visualisation

The following figures use a sample song from the test set to show the output frequency bin gains produced by the best performing EQ model (32MIDns).

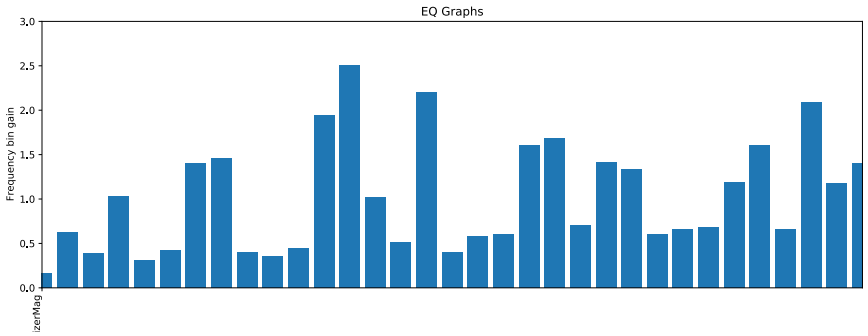


Figure 6.6: Synthesizer EQ

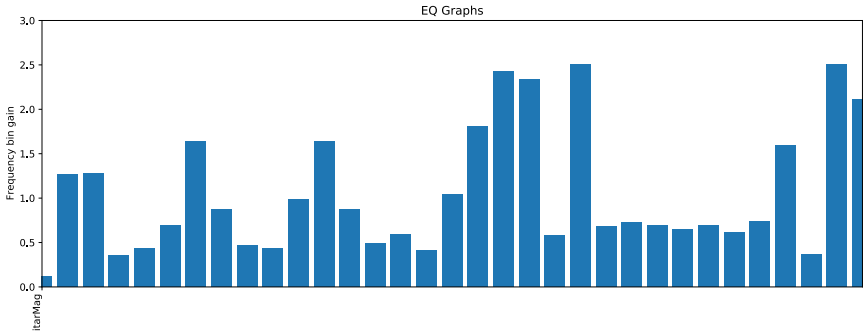


Figure 6.7: Acoustic Guitar EQ

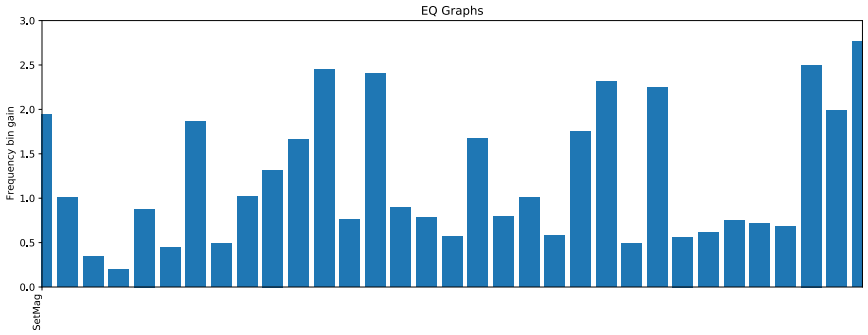


Figure 6.8: Drum Set EQ

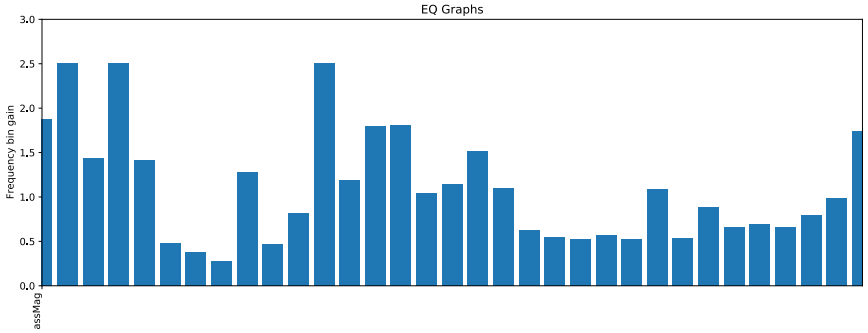


Figure 6.9: Electric Bass EQ

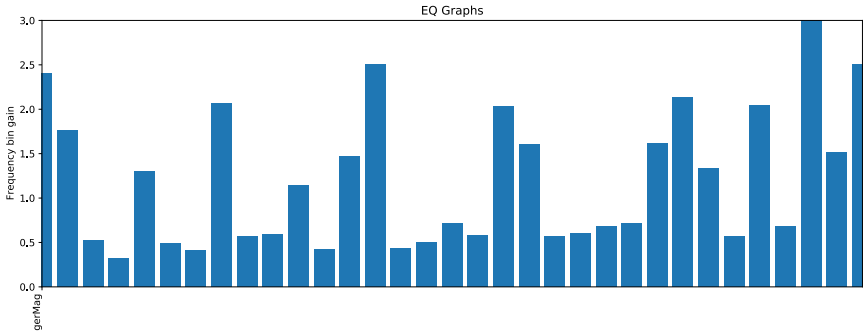


Figure 6.10: Female Singer EQ

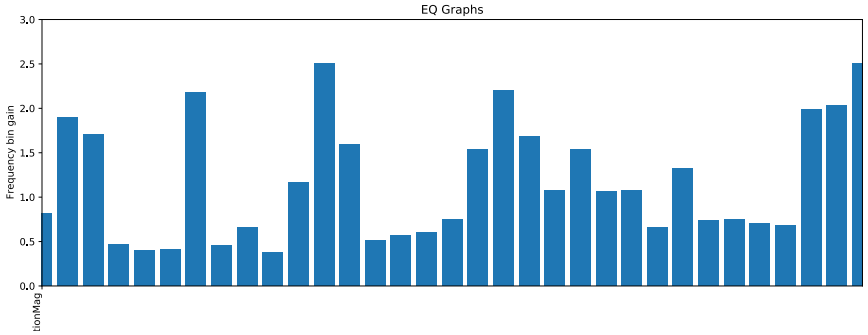


Figure 6.11: Horn Section EQ

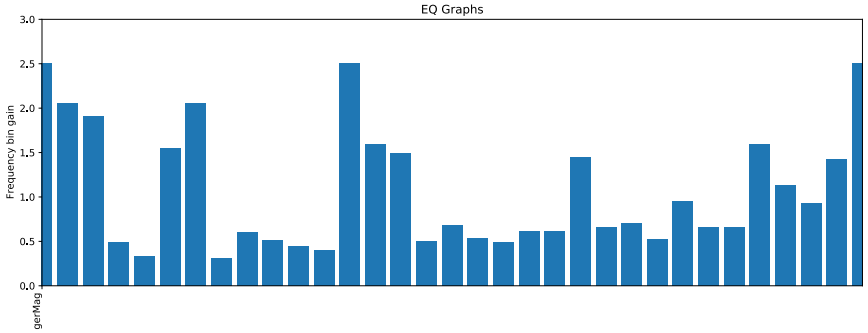


Figure 6.12: Male Singer EQ

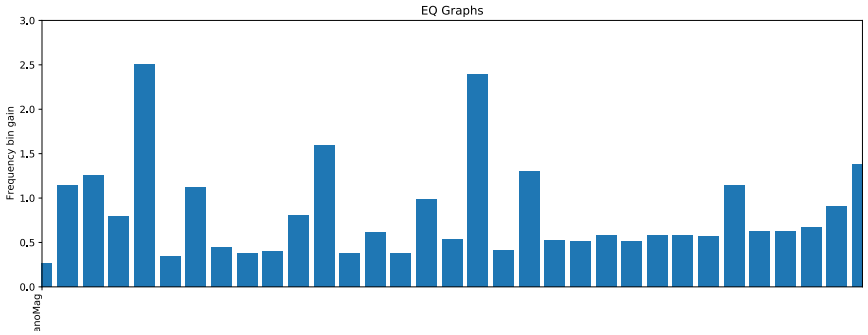


Figure 6.13: Piano EQ

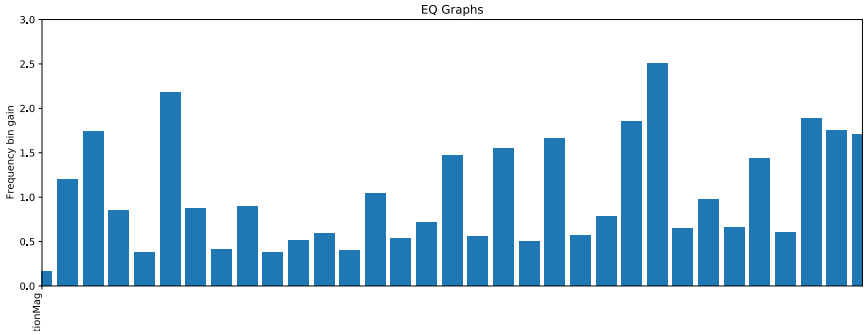


Figure 6.14: String Section EQ

There are a number of observations that can be made looking at these figures. The main purpose though is to demonstrate that the network learnt how to deal with frequency masking. The main instruments that will almost always fall victim to frequency masking are the drum and bass instruments. The kick

drum and any bass instruments both occupy the lower frequencies and therefore compete with each other. A mix engineer will usually experiment to find what sounds best, either cutting the bass when the kick is present or cutting the kick when the bass is present. In this example, looking at Figures 6.9 and 6.8, we can see that the network has chosen to cut the drums and boost the bass in the lower frequencies. Notice how after the first sub-bass frequency band, which are both amplified similarly, the next four frequency bands of the bass are all boosted above 1.0 while the next four bands for the drums are all cut below 1.0. This is an example of the network preventing frequency masking by boosting one signal and cutting another when they occupy the same frequency range.

It is also interesting to note that a number of these instruments have been given what resembles a high pass treatment. This is done by cutting the first band gain which is applied to the piano, string section, horn section, acoustic guitar and synthesizers. These are instruments one would expect in most cases to occupy the mid-range frequencies, except for possibly the synthesizer in some cases. It is not, however, done for instruments like the bass and the drums which one would expect to be present in the lower ranges, and so one would not normally apply a high pass in that case.

6.4 Volume Control

As is done for the equaliser networks above, the following table shows how the volume control networks performed on the test set, followed by a discussion relating to the key differences between the architectures, and how that affected

their performance. All values are the MSE values produced by equation 5.1. Again for the sake of spacing, the network architectures have been given aliases.

Network Architecture	**Alias**
64 Sample Multi-layered Densely Connected Neural Network	64Dns
8192 Sample Multi-layered Densely Connected Neural Network	8192Dns

Table 6.3: Volume Control network aliases

Song	**Unmixed**	**64Dns**	**8192Dns**
The Silver Sea	470	7	120
Keep Me Running	192	0.07	639
The Alchemist	1764	101	506
Chicken Fried Steak	275	81	129
Honey Brown	524	6	489
Return To Venezia	12	14	9
Mendelssohn Piano Trio 1 Movement 2	41	37	11
I'd Like To Know	2739	49	37
Improvisation 3	7	8	47
Reign Check	901	0.45	497

Prisoners Cinema	6232	89	4092
Miss Thing	1839	948	2994
Schubert Movement	91	8	21
I Promise	1279	116	783
Perfect Day	2002	561	2109

Table 6.4: Volume control MSE comparison results table

The results show that in general the 64 sample network performed better than the 8192 sample network. The main difference between the two models was that the 8192 model took in a single metric that represented the entire 8192 sample section of the song as explained in 5.2.2. This lack of information may have been the reason for the 8192Dns model's poor performance. The 64Dns model which took in the actual waveforms of the tracks has performed extremely well in some cases. The errors here are often lower than those in the EQ section. This means that adjusting the volumes of the different tracks was often able to produce mixes that were close to the professional mix without any EQ. There are some clear examples where that is not the case like with the songs **Miss Thing** and **Return to Venezia**.

In chapter 5.2.2, it was mentioned that the network's outputs were constrained in such a way that would allow for phase cancellation. The test set had a number of songs that contained bleed, however none of the output track gains produced by either the 8192 sample model or the 64 sample model produced negative gains. This means that no phase flipping and therefore no phase

cancellation took place, and it is safe to assume the networks did not learn phase flipping.

6.5 Combined System Results

After each of the networks had been trained and tested individually the best combination of models was selected to be combined. For the EQ the 32 Band musically informed model with additional dense layers was selected and the 64 sample model that performed best was the logical choice for the volume control model. These neural networks were then chained together so that the output gains of the EQ would be used to affect the input stems. These affected stems would then be the input stems which were passed to the volume control network. The volume control network would produce the track gains which would be applied to the affected stems to produce the final mix. The results of these final mixes are shown in the table below. All values are the MSE values produced by equation 5.1.

Song	**Unmixed MSE**	**Neural network mix MSE**
The Silver Sea	470	454
Keep Me Running	192	98
The Alchemist	1764	932

Chicken Fried Steak	275	51
Honey Brown	524	122
Return To Venezia	12	3
Mendelssohn Piano Trio 1 Movement 2	41	23
I'd Like To Know	2739	1353
Improvisation 3	7	3
Reign Check	901	271
Prisoners Cinema	6232	301
Miss Thing	1839	991
Schubert Movement	91	46
I Promise	1279	1046
Perfect Day	2002	579

Table 6.5: Combined system MSE comparison results table

Each example in the test set shows that the combined neural networks consistently produced mixes that were closer to the professional mix than the unmixed sum of the stems. This suggests that the neural networks have learnt to a certain extent how to mix songs in a way that is better than just adding the stems together. It is important to note that there is still error between the neural network outputs and the professional mix, so the mixes are never at the level of the professional mix. This is probably due to the fact that the mix engineer would have used a combination of many other effects like compression and reverb for example, compared to this project which was limited to equalisation and volume control.

The combined networks performed better than having the EQ alone, but worse than having the volume control alone for all except two songs. Volume control is an extremely important part of the mixing process, and is often able to fix numerous spectral problems in a mix, as well as allowing the listener to hear all the different stems as they were intended to be heard. While the loss is consistently lower for the volume network alone, the addition of the equaliser had a positive effect on a purely subjective level when listening to the final mixes. The spectral clarity and clear assignment of frequency space that is produced by the EQ creates pleasing mixes with losses that are still below that of the unmixed stems.

Chapter 7

Recommendations for Future Work

As mentioned in chapter 3.2, combining the two separate neural networks into one combined pipeline and training it as one network would likely produce better results. Currently each network is attempting to match the professionally mixed outputs on its own, without being aware of the effect that the other network would have. This is not ideal as the equaliser network might transform the tracks in such a way that increases the loss at the volume control stage. This is because the loss at the volume control network has no way to affect the equalisers weights. This is not ideal, and it is recommended that in any future work a combined pipeline is used.

It is also mentioned in chapter 5, that changing the order of the networks would be interesting and it is recommended that even in the proposed combined pipeline it would be interesting to observe the difference in loss when training the network with the equaliser being placed after the volume control network. It is therefore recommended that future work explores the effect that the order of the networks in the pipeline has on the results.

One more area that would be interesting to explore with these models would be investigating the effect of chaining multiple EQ networks one after the other. This is often done in a classic mixing chain where the mix engineer will do a first EQ that cuts out unwanted frequencies only, then later on in the chain have another EQ that boosts desired frequencies. The models in this paper have to do both at the same time. It would be interesting to use the activation functions to limit the first EQ to a maximum gain of 1.0 so that all it can do is cut unwanted frequencies, then have another EQ where the minimum threshold is 1.0 so that boosting is forced. The combinations and possibilities are endless, and it would be interesting to see the effects these changes would have on the final mixes.

Chapter 8

Conclusion

This project described the development of an automatic mixing system that uses machine learning to take individual musical stems and produce a near professional final mix. Multiple neural networks were designed and tested for equalisation and for volume control. Each network was trained on the MedleyDB dataset. The results of the system were evaluated using the spectrogram and waveform arrays of the final mixes produced by the system, compared with those of the professional mixes of a number of test songs from the MedleyDB datase. When the architectures were compared it was found that for the EQ network an architecture that consisted of a musically informed densely connected deep network performed best. For the volume control model, the 64 sample model which took the waveform as an input performed best. When evaluating the performance of the system on the test set, each of the best neural networks were chosen and combined. Each neural network mix had less error than the unmixed sum of the stems and achieved results close to those of the professionally produced mixes.

In [Ramırez and Reiss, 2017], a similar goal is presented and deep auto en-

coders with 3 layers were used. This project was able to improve on the results in [Ramırez and Reiss, 2017] by using a musically informed neural network structure. No metrics were provided in [Ramırez and Reiss, 2017] so the only comparison that can be made is the spectrogram images. When comparing the spectrograms in [Ramırez and Reiss, 2017] to those in chapter 6, the results of this project's neural networks are a clear improvement. This improvement is probably because the DAEs used in [Ramırez and Reiss, 2017] were not musically informed and had far fewer hidden layers.

In [Pestana et al., 2014] reducing frequency masking is mentioned as one of the most important steps when mixing a song. When assessing the output gains of the equaliser neural network in this project (as is done in chapter 6), it is clear that for certain stems there has been an effort to reduce frequency masking by adjusting the weights in such a way as to give different stems with similar frequency content their own space in the mix.

This project has applications in both live performances and in the mixing of songs recorded in a studio, however formal that studio may be. The project would aid a significant number of producers and musicians with varying levels of signal processing expertise by ensuring that the quality of the mixes of the songs experienced by audiences, whether at live shows or in recorded musical works, will be of a high standard. In so doing, the songs will be heard as they were intended by the musicians.

8.1 Applications

8.1.1 Open-Source Professional Mixing

There are an abundance of musicians and producers with the requisite talent to produce and publish musical works with ease, thanks to the proliferation of music production software. A large majority of these producers and musicians however, lack the experience in the mixing of songs and the signal processing knowledge needed to be able to properly mix the stems of the songs they produce into a high quality final mix. This project would aid these producers and musicians by providing a system that is able to take their individual tracks and produce a quality mix of their song. This would make it possible for their songs to be heard in a way that highlights the right parts of the song and allows each instrument and vocal to be heard clearly as intended.

8.1.2 Automatic Live Mixing

There are many live performances taking place every day where there are no sound engineers present, leaving the responsibility of mixing the songs to non-professionals. This often results in a poor experience for the audience. This project would create a system which is able to control the mix of a live performance by taking real time inputs from each sound signal present and, using the spectrograms and the neural networks developed, determine the equalisation and volume control to apply to each input in order to get as close to a professional mix as possible. This would find application in the many venues that host musical performances that feature musicians who do not have sound engineers available to them.

References

ABOUT BRUCE SWEDIEN, 2020. URL http://www.inthestudiowithbruceswedien.com/bruce-swedien-bio.html(accessedAugust10,2020).

Online Audio Mastering Software - Try Mastering Free, 2020. URL https://www.landr.com/en/online-audio-mastering?gclid=EAIaIQobChMI4MaL5qj44AIVYbXtCh1CKw3DEAAYASAAEgLaK_D_BwE&utm_campaign=AcquisitionBrandedENCoreSearch&utm_medium=cpc&utm_source=google&utm_term=LANDR_free_mastering(accessedAugust10,2020).

Susan Rogers, 2020. URL https://www.berklee.edu/people/susan-rogers(accessedAugust10,2020).

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

J. S. Abel and D. P. Berners. Filter design using second-order peaking and shelving sections. In *ICMC*, 2004.

I. Asimov. Runaround. *Astounding Science Fiction*, 29(1):94–103, 1942.

U. Berlin. Normal equal-loudness level contours-iso 226: 2003 acoustics international organization for standardization (iso). 2003.

R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello. Medleydb: A multitrack dataset for annotation-intensive mir research. In *ISMIR*, volume 14, pages 155–160, 2014.

W. Campbell, J. Paterson, and R. Toulson. *The effect of dynamic range compression on the psychoacoustic quality and loudness of commercial music.* 2010.

S. Chen, S. Billings, and P. Grant. Non-linear system identification using neural networks. *International journal of control*, 51(6):1191–1214, 1990.

K. Choi, G. Fazekas, M. Sandler, and K. Cho. Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2392–2396. IEEE, 2017.

F. Chollet et al. *Keras*, 2015. URL `https://github.com/fchollet/keras(accessedAugust10,2020)`.

B. De Man, J. Reiss, and R. Stables. Ten years of automatic mixing. 2017.

D. Dugan. Automatic microphone mixing. *Journal of the Audio Engineering Society*, 23(6):442–449, 1975.

S. Genc. Parametric system identification using deep convolutional neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2112–2119. IEEE, 2017.

J. George and L. Shamir. Unsupervised analysis of similarities between musicians and musical genres using spectrograms. *Artif. Intell. Research*, 4(2): 61–71, 2015.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016. ISBN 0262035618.

S. Hafezi and J. D. Reiss. Autonomous multitrack equalization based on masking reduction. *Journal of the Audio Engineering Society*, 63(5):312–323, 2015.

G. Heinzel, A. Rüdiger, and R. Schilling. Spectrum and spectral density estimation by the discrete fourier transform (dft), including a comprehensive list of window functions and some new at-top windows. 2002.

T. ISO. 43."iso 266: 1997, acoustics: Preferred frequencies for measurements". *International Organization for Standardization (ISO), Geneva, Switzerland*, 1997.

A. Kaplan and M. Haenlein. Siri, siri, in my hand: Who's the fairest in the land? on the interpretations, illustrations, and implications of artificial intelligence. *Business Horizons*, 62(1):15–25, 2019.

P. Kendrick, I. R. Jackson, F. F. Li, B. Fazenda, T. Cox, et al. Perceived audio quality of sounds degraded by non-linear distortions and single-ended assessment using hasqi. *Journal of the Audio Engineering Society*, 63(9): 698–712, 2015.

S. N. Kumpati, P. Kannan, et al. Identification and control of dynamical systems using neural networks. *IEEE Transactions on neural networks*, 1 (1):4–27, 1990.

S. I. Mimilakis, K. Drossos, T. Virtanen, and G. Schuller. Deep neural networks for dynamic range compression in mastering applications. In *Audio Engineering Society Convention 140*. Audio Engineering Society, 2016.

S. Nercessian. Neural parametric equalizer matching using differentiable biquads. In *23rd International Conference on Digital Audio Effects*, 2020.

S. Nisar, O. U. Khan, and M. Tariq. An efficient adaptive window size selection method for improving spectrogram visualization. *Computational intelligence and neuroscience*, 2016, 2016.

G. Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. *CUIDADO IST Project Report*, 54(0): 1–25, 2004.

E. Perez Gonzalez and J. Reiss. Determination and correction of individual channel time offsets for signals involved in an audio mixture. In *Audio Engineering Society Convention 125*. Audio Engineering Society, 2008.

E. Perez-Gonzalez and J. Reiss. Automatic gain and fader control for live mixing. In *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4. IEEE, 2009a.

E. Perez-Gonzalez and J. Reiss. Automatic equalization of multichannel audio using cross-adaptive methods. In *Audio Engineering Society Convention 127*. Audio Engineering Society, 2009b.

P. D. Pestana, J. D. Reiss, et al. Intelligent audio production strategies informed by best practices. 2014.

M. A. M. Ramırez and J. D. Reiss. Deep learning and intelligent audio mixing. *acoustic guitar*, 55:24, 2017.

D. Reed. A perceptual assistant to do sound equalization. In *Proceedings of the 5th international conference on Intelligent user interfaces*, pages 212–218, 2000.

E. Scerri. *Music Recognition, Analysis, and Automatic Tagging using Spectrogram Pattern Matching*. PhD thesis, University of Malta, 2016.

J. Scott, M. Prockup, E. M. Schmidt, and Y. E. Kim. Automatic multi-track mixing using linear dynamical systems. In *Proceedings of the 8th Sound and Music Computing Conference, Padova, Italy*, page 12. Citeseer, 2011.

J. Semmlow. *Signal Analysis in the Frequency Domain—Implications and Applications*, page 169–206. Elsevier, 2018. 10.1016/b978-0-12-809395-5.00004-7. URL http://dx.doi.org/10.1016/B978-0-12-809395-5.00004-7.

X. Serra and J. O. Smith. Stft 2. https://www.coursera.org/learn/audio-signal-processing/lecture/tjEQe/stft-2, 2019. Accessed August 14 2020.

T. Serre. Deep learning: the good, the bad, and the ugly. *Annual Review of Vision Science*, 5:399–426, 2019.

L. Shamir, N. Orlov, D. M. Eckley, T. Macura, J. Johnston, and I. G. Goldberg. Wndchrm–an open source utility for biological image analysis. *Source code for biology and medicine*, 3(1):13, 2008.

A. Shea. *Berklee Alum Behind Robin Thicke's Grammy Nominated 'Blurred Lines'*, Jan 2014. URL https://www.wbur.org/artery/2014/01/24/berklee-alum-robin-thicke(accessedAugust10,2020).

A. Simon, M. S. Deo, S. Venkatesan, and R. Babu. An overview of machine learning and its applications. *International Journal of Electrical Sciences & Engineering*, 1(1):22–24, 2016.

J. Stegemann and N. Buenfeld. A glossary of basic neural network terminology for regression problems. *Neural computing & applications*, 8(4):290–296, 1999.

J. Tot. *Multitrack Mixing: An Investigation into Music Mixing Practices*. PhD thesis, 03 2018.

V. Välimäki and J. D. Reiss. All about audio equalization: Solutions and frontiers. *Applied Sciences*, 6(5):129, 2016.

D. Velde. Aams home, 2020. URL http://curioza.com/.

E. Vickers. The loudness war: Background, speculation, and recommendations. In *Audio Engineering Society Convention 129*. Audio Engineering Society, 2010.

List of Figures

1.1 Trivial example of the effect of compression on dynamic range . 6
1.2 Example of compression on an actual vocal signal 6
1.3 Example of a graphic equaliser 9
1.4 Time domain representation of Blackman window 17
1.5 Frequency domain representation of Blackman window 17

4.1 Layout of files in dataset 42
4.2 Sine sweep spectrogram comparison 45
4.3 High frequency resolution and improved temporal resolution . . 46
4.4 Waveform comparison between different FFT and hop sizes . . . 47
4.5 Spectrogram comparison between different FFT and hop sizes . 47
4.6 Input-output pairs for the equaliser network example 50
4.7 Input-output pairs for the equaliser network example 51
4.8 Input-output pairs for volume control network example 53

5.1 System Model - EQ first 57
5.2 Equaliser Loss function explanation 60
5.3 Volume Control Loss function explanation 63
5.4 Simplified representation of the multi-layered densely connected equaliser model used in this paper 67

5.5 Simplified representation of the 32 band musically informed custom architecture equaliser model used in this paper 69
5.6 Simplified representation of the 32 band musically informed custom architecture equaliser model used in this paper with extra dense layers 71
5.7 Simplified representation of the 4096 band equaliser model used in this paper 72
5.8 Simplified representation of the densely connected 64 sample volume control model used in this paper 75
5.9 Simplified representation of the densely connected 8192 sample volume control model used in this paper 77
5.10 Phase cancellation example 79
5.11 Phase cancellation solution example 80

6.1 Output spectrogram of the professional mix 86
6.2 Output spectrogram of the 32Dns equaliser neural network on test input after 20 epochs of training 87
6.3 Output spectrogram of the 32MI equaliser neural network on test input after 20 epochs of training 89
6.4 Output spectrogram of the 32MID equaliser neural network on test input after 20 epochs of training 91
6.5 Output spectrogram of the 4096Dns equaliser neural network on test input after 20 epochs of training 92
6.6 Synthesizer EQ 93
6.7 Acoustic Guitar EQ 93
6.8 Drum Set EQ 93
6.9 Electric Bass EQ 94

6.10 Female Singer EQ 94
6.11 Horn Section EQ 94
6.12 Male Singer EQ 95
6.13 Piano EQ 95
6.14 String Section EQ 95

Appendices

Appendix A

Dataset Sounds

- piano
- drum set
- double bass
- bass clarinet
- trumpet
- trombone
- tenor saxophone
- distorted electric guitar
- electric bass
- male singer
- clean electric guitar
- trumpet section
- trombone section
- cello
- Main System
- violin
- vibraphone
- viola
- string section
- acoustic guitar
- female singer
- horn section
- synthesizer
- flute
- harp
- cello section
- violin section
- viola section
- french horn
- clarinet
- oboe
- bassoon
- soprano saxophone
- tambourine
- auxiliary percussion

- kick drum
- electric piano
- vocalists
- bass drum
- cymbal
- alto saxophone
- electronic organ
- fx/processed sound
- drum machine
- castanet
- glockenspiel
- shaker