AUTOMATION IN AUDIO ENHANCEMENT USING UNSUPERVISED LEARNING FOR UBIQUITOUS COMPUTATIONAL ENVIRONMENT

A PROJECT REPORT

Submitted by

HIMESH N. [Reg No: RA1611003010593]
VAIBHAV K. [Reg No: RA1611003010699]

Under the guidance of
Mr. M. Karthikeyan, M.E
(Assistant Professor (O.G), Department of Computer Science & Engineering)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

S.R.M. Nagar, Kattankulathur, Kancheepuram District

MAY 2020

SRM UNIVERSITY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled "**AUTOMATION IN AUDIO ENHANCEMENT USING UNSUPERVISED LEARNING FOR UBIQUITOUS COMPUTATIONAL ENVIRONMENT**" is the bonafide work of " **HIMESH N. [Reg No: RA1611003010593], VAIBHAV K. [Reg No: RA1611003010699], , ,** ", who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Mr. M. Karthikeyan, M.E **GUIDE** Assistant Professor (O.G) Dept. of Computer Science & Engineering	Dr. B. Amutha **HEAD OF THE DEPARTMENT** Dept. of Computer Science & Engineering
Signature of the Internal Examiner	Signature of the External Examiner

ABSTRACT

The incorporation of electronics into the music production process has only been shaping new boundaries in the field of production. Recent decade has seen the rise of a fairly new genre called Electronic Dance Music, where music is being recorded, arranged and processed using electronic media. Certain researches have also revealed how Neural Networks are being used for automating the process of composition and production of music. This calls for a system that automates the post-production audio enhancement and optimization techniques, called Mastering, in order to attain a balance in the sound quality and make this procedure more robust. To do the same, we make use of Fast Fourier Transform Techniques, Unsupervised Learning and Neural Networks. The chief principle behind the processing of any audio is to provide a sophisticated mechanism to enhance the extracted acoustic characteristics of the signal. The evaluation indices of an optimized or mastered audio, via human listening test, to showcase the power of Artificial Intelligence and how it can be used as a constraint optimization model to optimize playback of the stereo mix.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my guide,Mr. M. Karthikeyan his valuable guidance, consistent encouragement, personal caring, timely help and providing me with an excellent atmosphere for doing research. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to me for completing this research work.

Author

TABLE OF CONTENTS

ABSTRACT **iii**

ACKNOWLEDGEMENTS **iv**

LIST OF TABLES **vii**

LIST OF FIGURES **ix**

ABBREVIATIONS **x**

1 INTRODUCTION **1**

1.1 KEY CONCEPTS 1

1.1.1 Audio Mastering 1

1.1.2 Acoustic Characteristics 2

1.2 TOOLS 3

2 LITERATURE SURVEY **4**

2.1 Composition of music using Artificial Intelligence 4

2.2 Audio Signal Processing Using Python 5

2.3 Music Information Retrieval & Analysis 6

2.4 Audio Enhancement & Mastering 7

3 System Architecture **8**

4 System Realisation **10**

4.1 Feature Extraction 10

4.1.1 Spectrogram 12

4.1.2 Zero-Crossing Rate 13

4.1.3 Spectral Centroid 13

4.1.4 Mel-Frequency Cepstral Coefficients 15

4.2 Analysis and Genre Detection 15
4.2.1 Dataset 17
4.2.2 Classification 18
4.3 Gain Optimisation - Limiter 20
4.4 Compression Optimisation 22

5 System Testing **24**
5.1 Objective Tests 24
5.2 Subjective Testing 24
5.2.1 Phase I reviews: 25
5.2.2 Phase II reviews: 25
5.3 Accuracy Testing of Neural Networks 25

6 System Applications **27**
6.1 Post-production Enhancements 27
6.2 Surveillance audio amplification 27
6.3 Medical Applications 27
6.4 Flight data Amplifications 28

7 Conclusion **29**

8 Future Enhancement **30**

A Fast Fourier Transform **31**
A.1 Sampling Rate 31

LIST OF TABLES

5.1 Similarity Index 24
5.2 General Impressions in Phase I 25
5.3 General Impressions in Phase II 25

LIST OF FIGURES

3.1 Class Diagram For Architecture. 8

3.2 Block Diagram For Architecture. 8

3.3 Flow Chart For Architecture. 9

4.1 Flow Chart For Feature Extraction. 10

4.2 Constituent sinusoidal components of the original signal. 11

4.3 Music Information Retrieval (MIR) Process initialisation. 12

4.4 Script to Display Waveform. 12

4.5 Script for spectrogram. 13

4.6 Spectrogram Of An Audio. 13

4.7 Script for Zero-Crossing Rate (ZCR). 13

4.8 ZCR Of An Audio. 14

4.9 Script for Spectral Centroid (a). 14

4.10 Script for Spectral Centroid (b). 14

4.11 Script for Spectral Centroid (c). 14

4.12 Script for Spectral Centroid (d). 14

4.13 Spectral Centroid Of An Audio. 15

4.14 Script for Mel Frequency Cepstral Coefficient (MFCC). 15

4.15 MFCC Of An Audio. 16

4.16 Neural Network Structure. 16

4.17 Code Snippet for ANN Packages. 17

4.18 Code Snippet for ANN. 17

4.19 Artificial Neural Networks (ANN) Running Epoch 18

4.20 Dataset Representation. 18

4.21 Hip Hop Song Envelope. 19

4.22 Hip Hop Song With Vocals Envelope. 19

4.23 Disco Song Envelope. 19

4.24 Classical Song Envelope. 20
4.25 An analog limiter (a) 21
4.26 An analog limiter (b) 21
4.27 Code Snippet For Limiter. 22
4.28 Code Snippet For Compressor. 23

5.1 Training Curve for Neural Network. 26
5.2 Accuracy of the Neural Network. 26

8.1 Equalizer. 30

A.1 Effect of Sampling Rate (sr). 31

ABBREVIATIONS

FFT	Fast Fourier Transform
DAW	Digital Audio Workstation
ANN	Artificial Neural Networks
SSC	Spectral Subband Centroid
MIR	Music Information Retrieval
MFCC	Mel Frequency Cepstral Coefficient
LSTM	Long Short Term Memory
ZCR	Zero-Crossing Rate
CGS	"Computer Generated Sound"
RMSE	Root Mean Square Error
CSV	Comma Separated Value
ReLU	Rectified Linear Unit
SC	Standard Scaler
Hz	Hertz
kHz	kilo-Hertz
sr	Sampling Rate

LIST OF SYMBOLS

$f(t)$	Overall Time
ω	Frequency
θ	Angle of Dip, deg

CHAPTER 1

INTRODUCTION

The music production procedure comprises two main steps, the first is to record instruments and arrange them to make a track, and the second being the post-production mixing and mastering of the track. Audio enhancements and optimization, mastering, is done to balance and optimize the audio. Traditionally, each step of this mastering procedure has been done by a sound engineer in a recording studio, manually. This is done using the help of specialized software called the Digital Audio Workstation (DAW).

However, recent years have seen a consistent increase in the popularity and growth of the services that provide automation to this procedure. Applications like "Snapchat", allows users to perform automated post-processing of both audio signals or images without requiring any knowledge of signal processing. Moreover, there are some online platforms that use automation for mastering as a service without much details from the user. The input to such platforms are the user mix or the initial audio and the output is the processed audio.

1.1 KEY CONCEPTS

The idea behind this piece of work is to create an application that can make the process of enhancement of audio automated using calculated parameters. The output of this model generates a 16-bit processed audio, in the *wav* file format.

1.1.1 Audio Mastering

Audio mastering is a technique responsible for dynamic and frequency response and the amplification characteristics.

1.1.2 Acoustic Characteristics

Spectrum

This is the fourier transform of a signal, and hence is instrumental in converting a time-domain signal to a frequency domain. Hence, it can be said that a spectrum is the frequency domain representation of the input audio's time signature.

Spectral Centroid

A spectral centroid is defined as the center of mass of the audio. It is obtained by using signal processing algorithms over audio, by calculating the mean of the amount of frequencies in a signal at a time period.

Mel Frequency Cepstral Coefficient

MFCC are the set of features that are instrumental in deciding the shape of the audio envelope.

Frequency - Time Graph and Spectrogram

The python modules like Matplotlib.pyplot provide certain features like specgram() method that uses Fast Fourier Transform (FFT) to extract all the frequencies present in the audio file and plot it against time period to provide analysis per unit time.

Roll-Off Factor

The roll-off factor allows the estimation of highly energetic frequencies in the signal. The proposed system requires the Roll-off frequency to understand the different energies associated with different genres, which will play a crucial role in the genre prediction process.

Spectrum

The rate at which a signal changes from negative to positive and vice-a-versa is defined as the Zero-Crossing Rate. It is a highly recommended feature for the use of genre prediction, simply because it gives a higher value for higher percussive audio, like that of rock and metal genres.

1.2 TOOLS

A Python environment, inducing the techniques of Fast Fourier transform (FFT) as an application of unsupervised learning algorithm of machine learning and Artificial Neural Network, for genre predictions, was used in order to automate this process of post-production enhancements, i.e. mastering. The use of libraries of algorithms for audio analysis, dynamic range control and limiter control, and other audio effects was made in order to extract musical features and process the audio in order to optimize it. A fourier analysis of an audio spectrum is used to convert the signal from its original time domain to a frequency domain or vice-a-versa. Fast Fourier Transform is a mathematical operation that changes the domain (x-axis) of a signal from time to frequency. The latter is particularly useful for decomposing a signal consisting of multiple pure frequencies.

CHAPTER 2

LITERATURE SURVEY

2.1 Composition of music using Artificial Intelligence

Music is an art form that consists of organised sounds with respect to a time domain. There are various elements in a song that we hear every day. These elements can be instruments, computer-generated sounds or other sound samples, all arranged in racks and layered on top of each other. Any song is majorly constructed using a melody, a harmony, a beat and an ambience. In their research, Zhao, K., Li, S., Cai, J., Wang, H., & Wang, J. (2019) explains the use of Biaxial Long Short Term Memory (LSTM) networks to generate polyphonic music. A separate "lookback" factor is also introduced for the purpose of improving the architecture of the system. The design of such an architecture for emotion-based composition indices of the evaluation of generated music by comparing it to real music.

Venugopal, K., & Madhusudan, P. (2017), as they quote it, “musico-mathematically” divides audio into three dimensions namely volume, rhythm and pitch. Physics, however, defines these cardinals as amplitude, periodicity and frequency according to the theories based in classical sonic wave mechanics. They then go on to explain the use of artificial neural networks for the synthesis of rhythms and composing music. ANN, as we know it, only accepts numerical inputs. Therefore, according to them, music must be assimilated into these three cardinals, each of which is compatible with a learning network. ANN can hence be used to recognize a pattern and maybe then exploited to learn to recognize the patterns that comprise music. A trained network, therefore, can be used to generate patterns that are mathematically similar to pre-existing music.

J.-P. Briot, G. Hadjeres and F.-D. Pachet (2019) has been successful in providing a critical analysis of different ways of using ANN to compose musical content. They divided their concerns into 5 major dimensions. These are

What content is to be generated? What concepts are to be manipulated? What are the limitations and open challenges? What type of neural network is to be used? How to model and control this mechanism of generation?

For each extent, they conducted a relative investigation of various models and techniques and proposed certain tentative multi-dimensional categorization. This categorization was completely based on the analysis of many pre-existing neural network models for music composition aesthetically selected from various relevant literature and manuscripts.

2.2 Audio Signal Processing Using Python

The use of python for the very purpose of audio processing is a fairly new notion. Since python provides a free to use environment, the idea of Using python for visualization and processing is inexpensive to construct. Viarbitskaya, T., & Dobrucki, A. (2018) define ways for the recognition of instruments and the techniques used by different artists for playing that instrument for the perception and correction of errors in a given audio sample. This system demonstrates the attribute of recorded sound and also defines the methods to compare and contrast amplitudes and frequencies of the same sample, but performed by different individuals and also with using various distinct instruments. For this purpose, they made use of certain signal processing algorithms, which are conveniently available in conventional python libraries such as “numpy” and “scipy”. The indispensable idea behind such processing is the critical recognition and analysis of errors, based on the individual style of the player.

The python programming environment provides the features for both visualization and computational tasks. Python is advantageous to developers to make the use of various packages that provide certain potentials, such as a matrix or an array manipulation, visualization, digital signal processing, and image processing. or the implementations of many processing techniques. Therefore, it would be highly superior to combine the flexibility and power of a more general purpose programming language such as Python, which allows a faster process for the development of prototypes using packages that are focused on audio processing techniques.

2.3 Music Information Retrieval & Analysis

The feature-extraction plays a crucial role in analyzing the signal and finding certain relationships. The data provided of audio can't be directly understood by the models, and hence cannot be directly converted to a machine understandable format. Therefore, for this very purpose the techniques of MIR are used. This process contemplates most of the data in a "machine-understood" method. Hence, it is used for prediction, classification and recommendation algorithms. Using librosa one can easily analyze the audio in its waveform. For playing the sound file pyAudio package is used. Wickert, Mark (2018) shows how easy it is to develop and test using real-time signal processing algorithms for processing fed input and returning processed outputs using the package scikitdsp-comm. A proper control of code is methodically obtained by using the "ipywidgets" package. The bandwidth and the sampling rate of the audio signal are limited by the OS' audio subsystem potential, but it can at least be about 48 kiloHertz and mostly is 96 kiloHertz. IPython.display provides a simple interface allowing the capabilities of playing the audio file in the module.

McFee, Brian & Raffel, Colin & Liang, Dawen & Ellis, Daniel & Mcvicar, Matt & Battenberg, Eric & Nieto, Oriol (2015) defines MIR is an intersection of library science, feature extraction, DSP, machine learning, and musicology. Since this was an upcoming science, it was quite clear that the exact applications given by librosa may not be much useful in representing the state-of-the-art for any given problem. Therefore, instead of using the "abstract-class" hierarchies, they allowed the users to create their own functions when in need. To provide such a modular environment, they made the use of packages such as "numpy" and "scipy".

2.4 Audio Enhancement & Mastering

D. S. Eley (2013), in his book, reveals that the secret to giving any mix a professional and a commercial sounding finish is to master the audio thoroughly. Bob Katz and Robert A. Katz also define mastering. They define it as the technique used for balancing and optimizing the sonic elements of the audio, which is done using tools like limiter, compressor and an equaliser.

Jijun Deng, Wanggen Wan, XiaoQing Yu, Xueqian Pan, & Wei Yang (2011) introduces an audio attributing algorithm for the optimisation of sonic element. This algorithm is completely based on the Spectral Subband Centroid (SSC). Initially, the predominant pitches of the signal is extracted using enhancement techniques. Then it computes the SSC features of the audio. The preliminary test reveals the accuracy of the model. However, after reviewing these results, it does appear that there is plenty of room for improvement. Naiduchowski, E., Lewandowski, M., & Bobinski, P. (2018) also presents a system which provides the feature of automatically optimizing the unprocessed audio signal based on the reference parameters, which are extracted through analysis of acoustic characteristics of the audio spectrum.

CHAPTER 3

SYSTEM ARCHITECTURE

The following diagrams can be used to understand the design of the system.

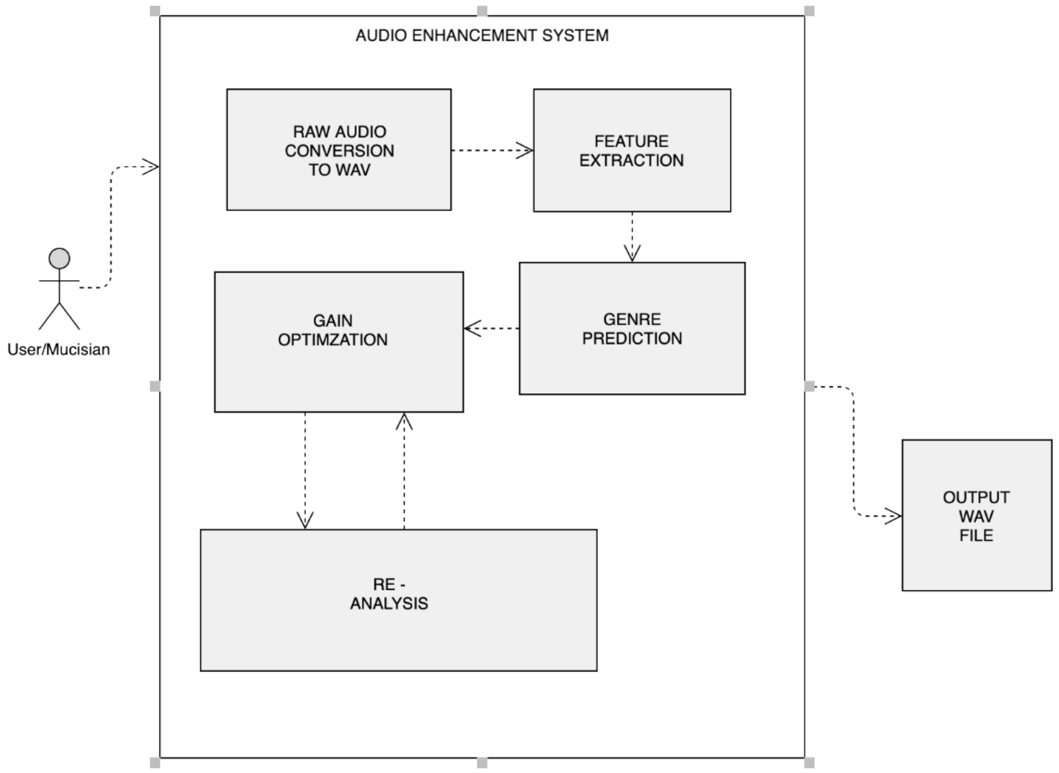


Figure 3.1: Class Diagram For Architecture.

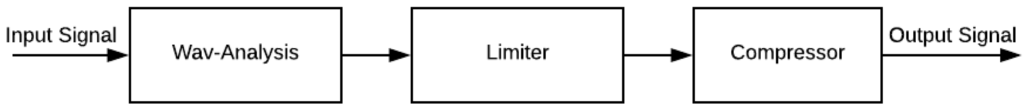


Figure 3.2: Block Diagram For Architecture.

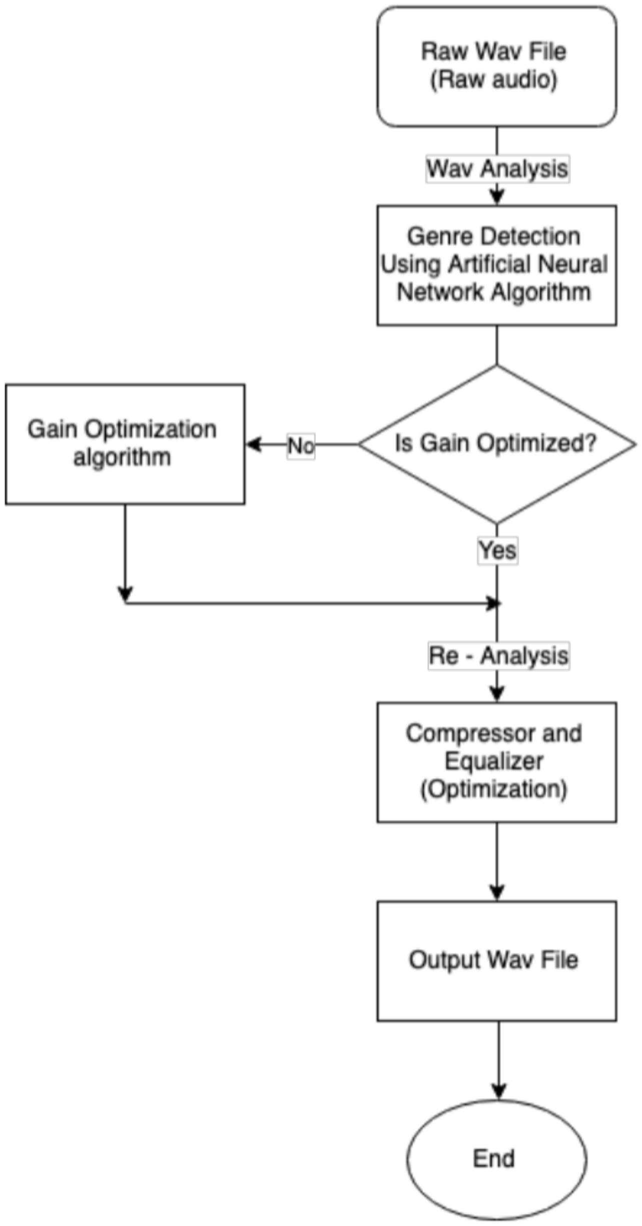


Figure 3.3: Flow Chart For Architecture.

CHAPTER 4

SYSTEM REALISATION

4.1 Feature Extraction

One of the most important steps in any novel system is to extract defining characteristics to find an indicative relationship for the synthesis of the application. In a model that deals with audio, such as ours, these attributes are indeed the acoustic characteristics that basically define the construct of any audio. As we already know that the goal of this project is to make Artificial Intelligence to enhance the raw audio so as to make the listening experience balanced and cohesive. Therefore, the first step in this process is to make sure that these features are retrieved with the utmost efficiency. A separate python script is written to do the same. The figure below explains the process of Musical Information Retrieval.

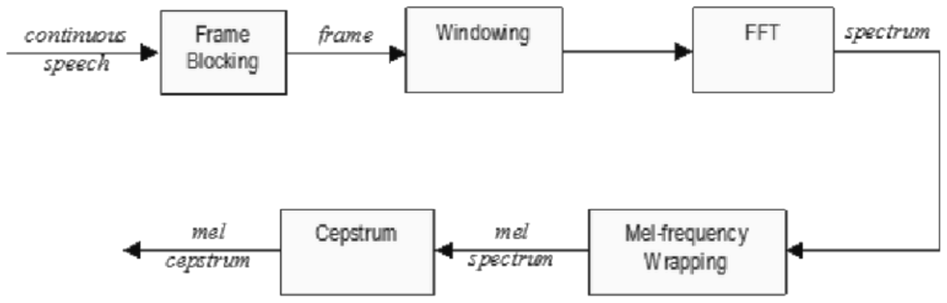


Figure 4.1: Flow Chart For Feature Extraction.

It plays a crucial role in analyzing the signal and finding certain relationships. The data provided of audio can't be directly understood by the models, and hence cannot be directly converted to a machine understandable format. Therefore, for this very purpose the techniques of music information retrieval are used. This process contemplates most of the data in a "machine-understood" method. Hence, it is used for prediction, classification and recommendation algorithms.

Using FFT equations in python, we were able to define a script to extract the features using fourier analysis.

Technically, such a process can be represented easily by the following FFT equation:

$$\boxed{F(\omega)\int_{-\infty}^{\infty} f(t)e^{-j\omega t}dx} \tag{4.1}$$

where

$$\boxed{e^{-j\omega t} = \cos(\omega t) + j\sin(\omega t)} \tag{4.2}$$

The equation represents the product of the overall time of the waveform f(t) with a complex exponential, and the resultant

$$F(\omega)$$

is the Fourier coefficient. This is an important equation for converting the original audio signal into its constituent sinusoidal components.

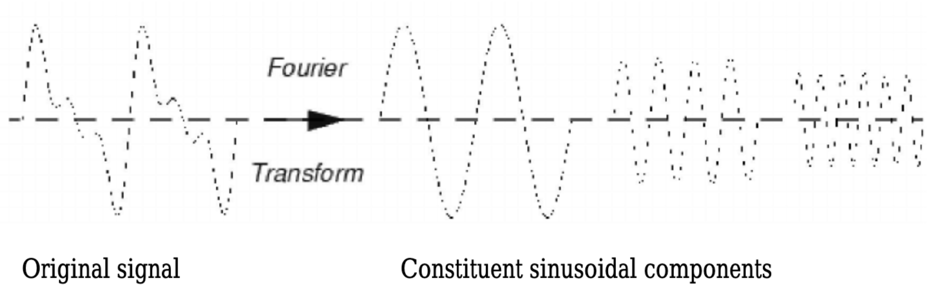
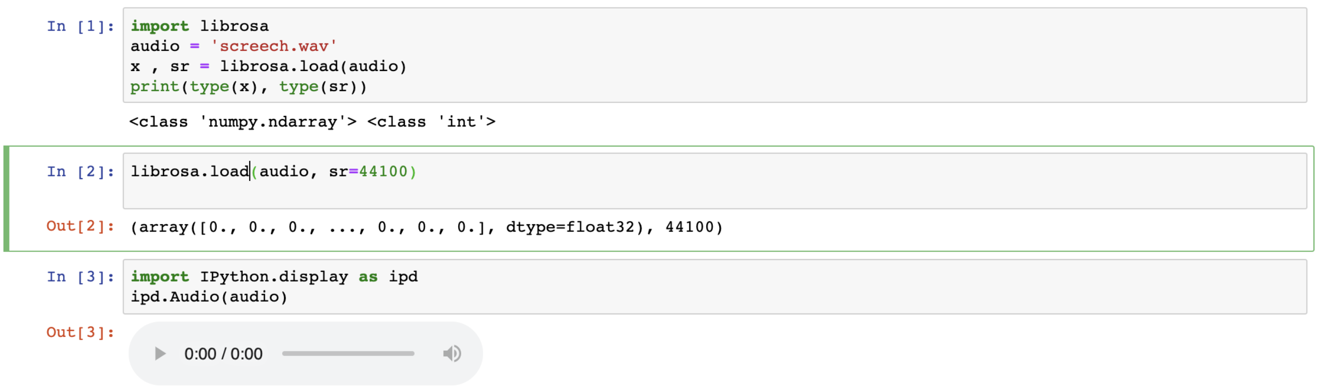


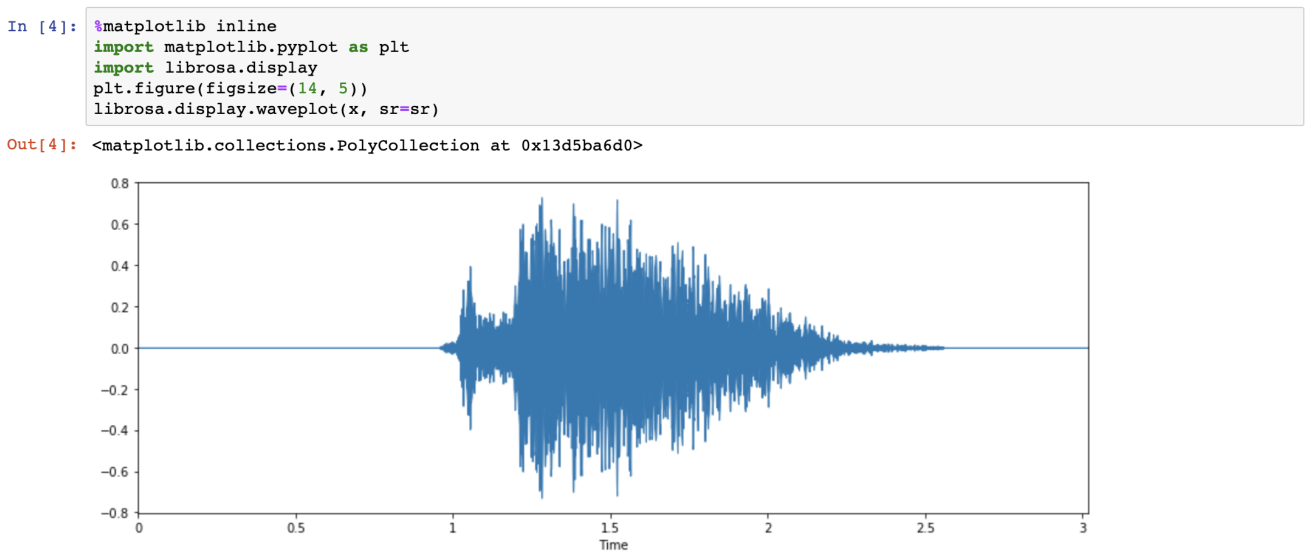
Figure 4.2: Constituent sinusoidal components of the original signal.

The first step in this process is to import all the important libraries and the audio for the extraction for the process of extracting empirical acoustic audio features. Using librosa one can easily analyze the audio in its waveform. For playing the sound file pyAudio package is used.



```
In [1]: import librosa
        audio = 'screech.wav'
        x , sr = librosa.load(audio)
        print(type(x), type(sr))

        <class 'numpy.ndarray'> <class 'int'>

In [2]: librosa.load(audio, sr=44100)

Out[2]: (array([0., 0., 0., ..., 0., 0., 0.], dtype=float32), 44100)

In [3]: import IPython.display as ipd
        ipd.Audio(audio)

Out[3]:
```

Figure 4.3: MIR Process initialisation.

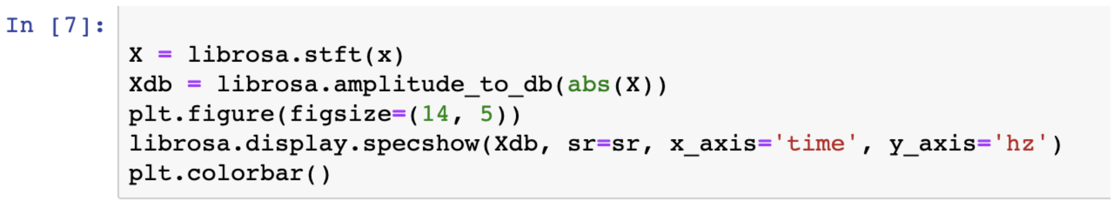


```
In [4]: %matplotlib inline
        import matplotlib.pyplot as plt
        import librosa.display
        plt.figure(figsize=(14, 5))
        librosa.display.waveplot(x, sr=sr)

Out[4]: <matplotlib.collections.PolyCollection at 0x13d5ba6d0>
```

Figure 4.4: Script to Display Waveform.

4.1.1 Spectrogram

A spectrogram is a visual representation of the spectrum of frequencies of sounds or other signals as they vary with time. It's a representation of frequencies changing with respect to time for given music signals.



```
In [7]:
X = librosa.stft(x)
Xdb = librosa.amplitude_to_db(abs(X))
plt.figure(figsize=(14, 5))
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar()
```

Figure 4.5: Script for spectrogram.

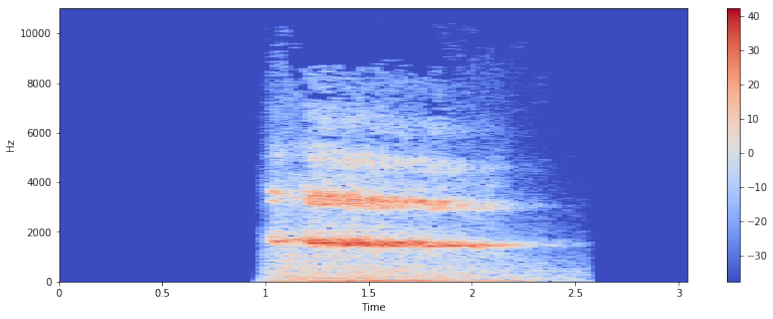
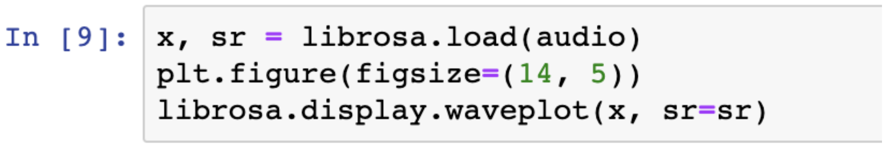


Figure 4.6: Spectrogram Of An Audio.

4.1.2 Zero-Crossing Rate

The ZCR is the rate of sign-changes along with a signal, i.e., the rate at which the signal changes from positive to negative or back. This feature has been used heavily in both speech recognition and music information retrieval. It usually has higher values for highly percussive sounds like those in metal and rock.



```
In [9]: x, sr = librosa.load(audio)
        plt.figure(figsize=(14, 5))
        librosa.display.waveplot(x, sr=sr)
```

Figure 4.7: Script for ZCR.

4.1.3 Spectral Centroid

It indicates where the ”centre of mass” for a sound is located and is calculated as the weighted mean of the frequencies present in the sound. If the frequencies in music are the same throughout then the spectral centroid would be around a centre and if there are high frequencies at the end of sound then the centroid would be towards its end.

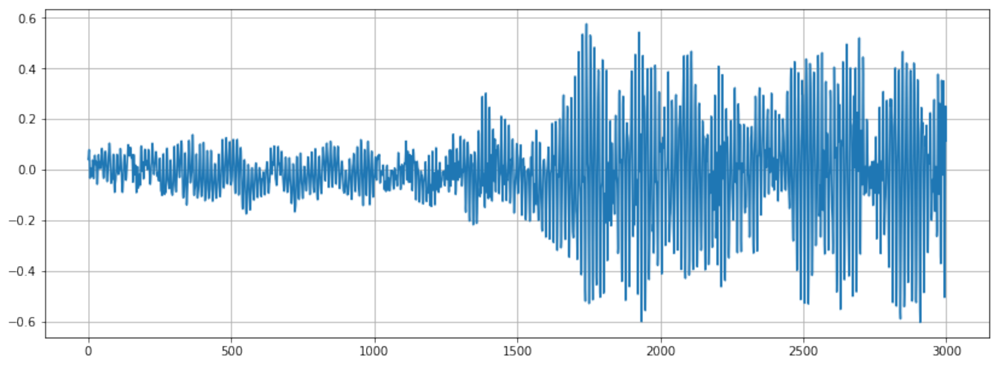
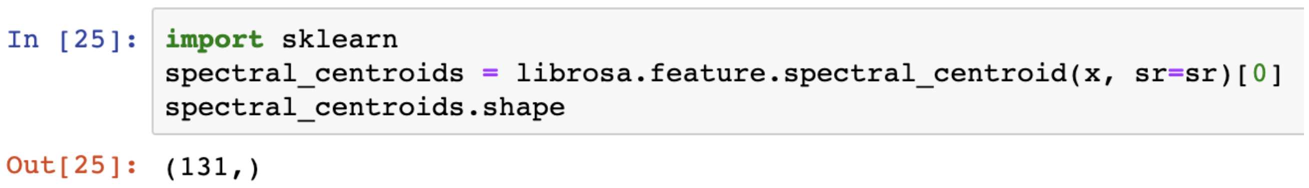
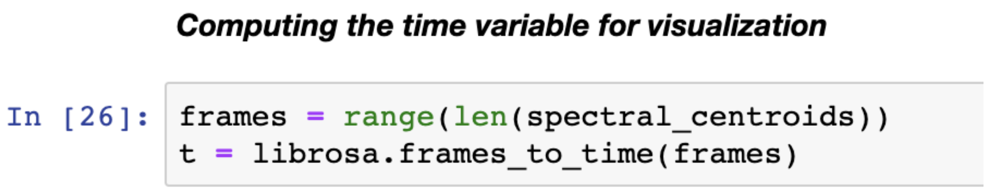


Figure 4.8: ZCR Of An Audio.



```
In [25]: import sklearn
         spectral_centroids = librosa.feature.spectral_centroid(x, sr=sr)[0]
         spectral_centroids.shape

Out[25]: (131,)
```

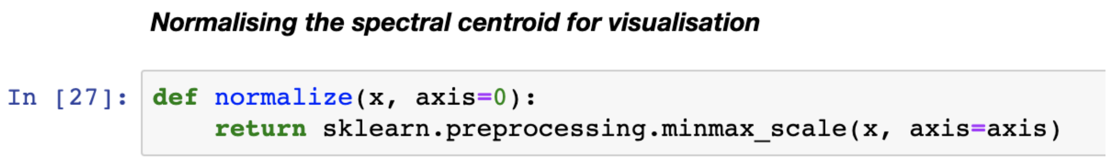
Figure 4.9: Script for Spectral Centroid (a).



Computing the time variable for visualization

```
In [26]: frames = range(len(spectral_centroids))
         t = librosa.frames_to_time(frames)
```

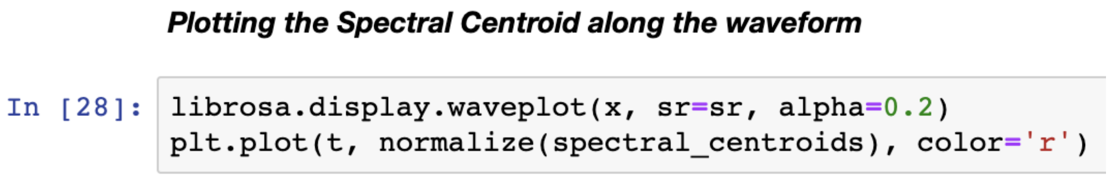
Figure 4.10: Script for Spectral Centroid (b).



Normalising the spectral centroid for visualisation

```
In [27]: def normalize(x, axis=0):
             return sklearn.preprocessing.minmax_scale(x, axis=axis)
```

Figure 4.11: Script for Spectral Centroid (c).



Plotting the Spectral Centroid along the waveform

```
In [28]: librosa.display.waveplot(x, sr=sr, alpha=0.2)
         plt.plot(t, normalize(spectral_centroids), color='r')
```

Figure 4.12: Script for Spectral Centroid (d).

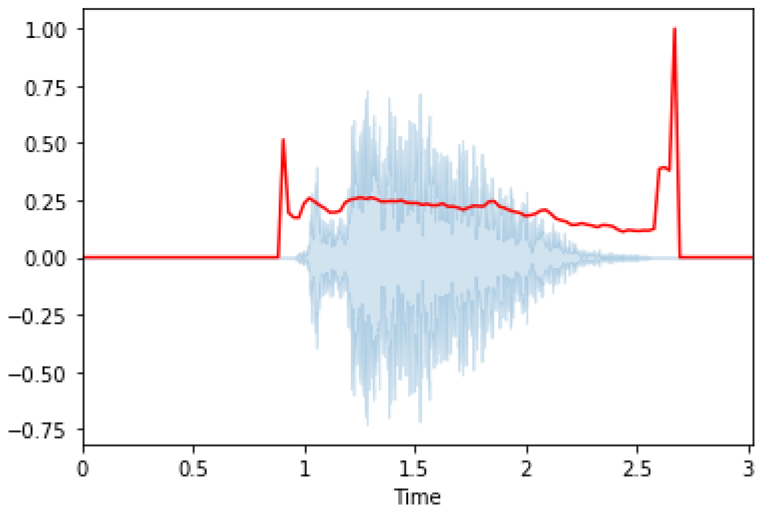
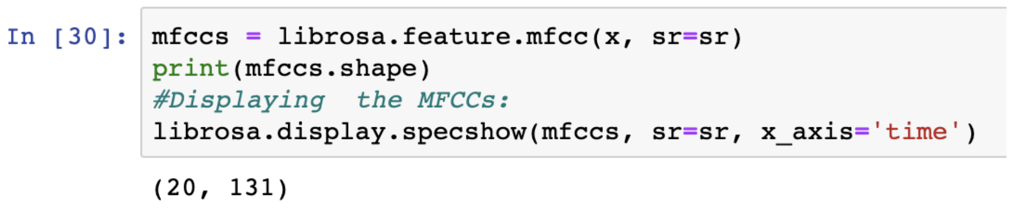


Figure 4.13: Spectral Centroid Of An Audio.

4.1.4 Mel-Frequency Cepstral Coefficients

This feature is one of the most important methods to extract a feature of an audio signal and is used majorly whenever working on audio signals. The MFCC of a signal are a small set of features (usually about 10–20) which concisely describe the overall shape of a spectral envelope



```
In [30]: mfccs = librosa.feature.mfcc(x, sr=sr)
         print(mfccs.shape)
         #Displaying  the MFCCs:
         librosa.display.specshow(mfccs, sr=sr, x_axis='time')

         (20, 131)
```

Figure 4.14: Script for MFCC.

4.2 Analysis and Genre Detection

The research begins with the analysis of a wav file by using MatPlotLib and Librosa libraries and plotting the wav file in the frequency versus time graph so as to analyze the frequency band graph and the necessary elements of the raw sound files like amplitude, pitch, peak volume etc. .

Due to the requirement of the whole model of dealing with the wav file, another script was written converting the most commonly used audio file format, mp3, to the

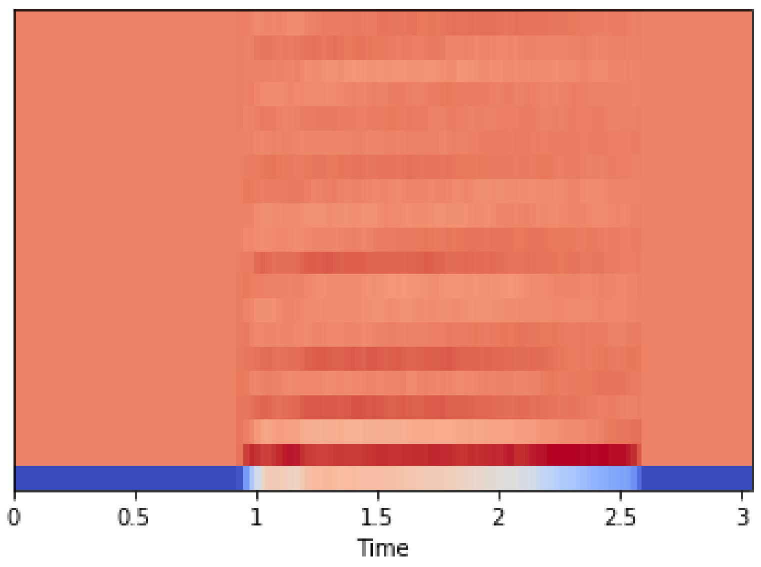


Figure 4.15: MFCC Of An Audio.

less compressed wav file.

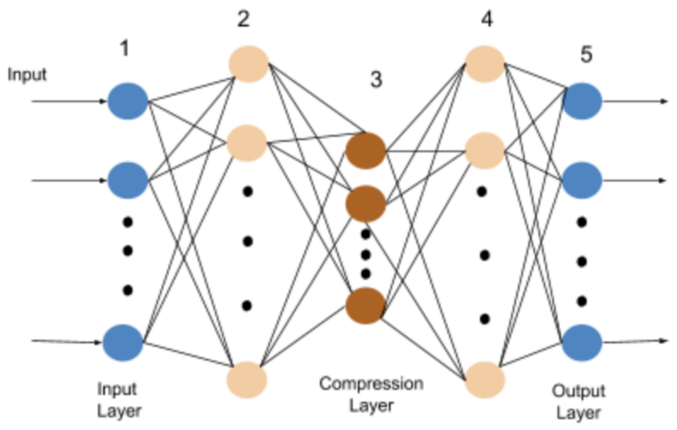
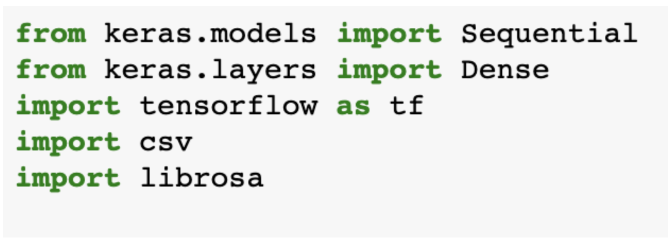


Figure 4.16: Neural Network Structure.

Neural networks (connectionist systems) are computing systems that are basically trained to "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any prior knowledge about cats, for example, that they have fur, tails, whiskers and cat-like faces. Instead, they automatically generate identifying characteristics from the learning material that they process.

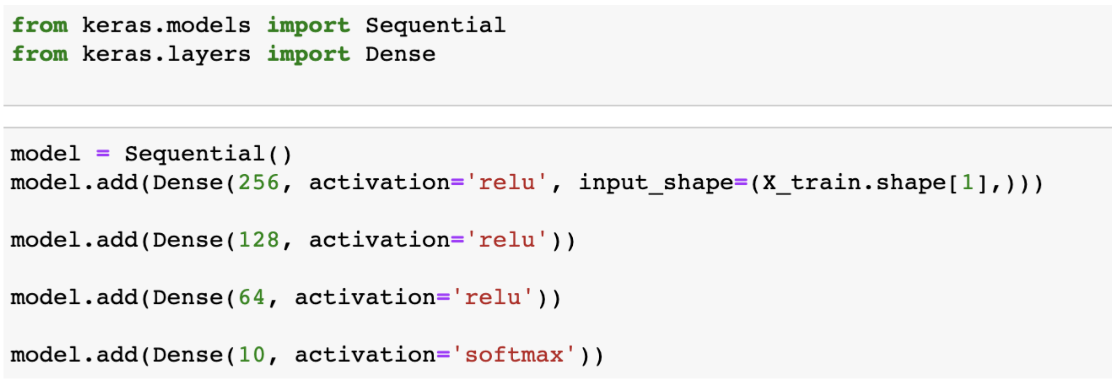
Similarly in our model we defined a 5 layer ANN for genre classification tasks where



```
from keras.models import Sequential
from keras.layers import Dense
import tensorflow as tf
import csv
import librosa
```

Figure 4.17: Code Snippet for ANN Packages.

given an input features X , We tend to predict the class Y. Based on the parameters for the training set of data , ANN analyzes the data variation and tends to minimize the cost function output hence re-adjust the weights of each node by calculation of loss.



```
from keras.models import Sequential
from keras.layers import Dense
```

```
model = Sequential()
model.add(Dense(256, activation='relu', input_shape=(X_train.shape[1],)))

model.add(Dense(128, activation='relu'))

model.add(Dense(64, activation='relu'))

model.add(Dense(10, activation='softmax'))
```

Figure 4.18: Code Snippet for ANN.

First layer is responsible for the calculation of the weighted sum of bias term , its weights, and its inputs, and then execute the Rectified Linear Unit (ReLU) Activation Function. The ReLU activation function basically states that anything with a magnitude of zero will remain zero, while anything with a magnitude greater than zero will be the value itself. The first layer contains 100 neurons. As we progress in the number of layers, the layers keep becoming denser. Further, we just fit the training data and labels, after standardizing the data using Standard Scaler (SC), and then make the system to repeat for 20 epochs (later to even 1000 epochs)

4.2.1 Dataset

The dataset used is a compilation of released and popular songs of various genres. We made the use of 'GTZEN' dataset to prepare a comma separated value file with the data of almost 1000 songs. Each genre is classified as a label, and then the properties such as MFCC, Root Mean Square Error (RMSE), chroma shift, zero-crossing rate, etc., linking

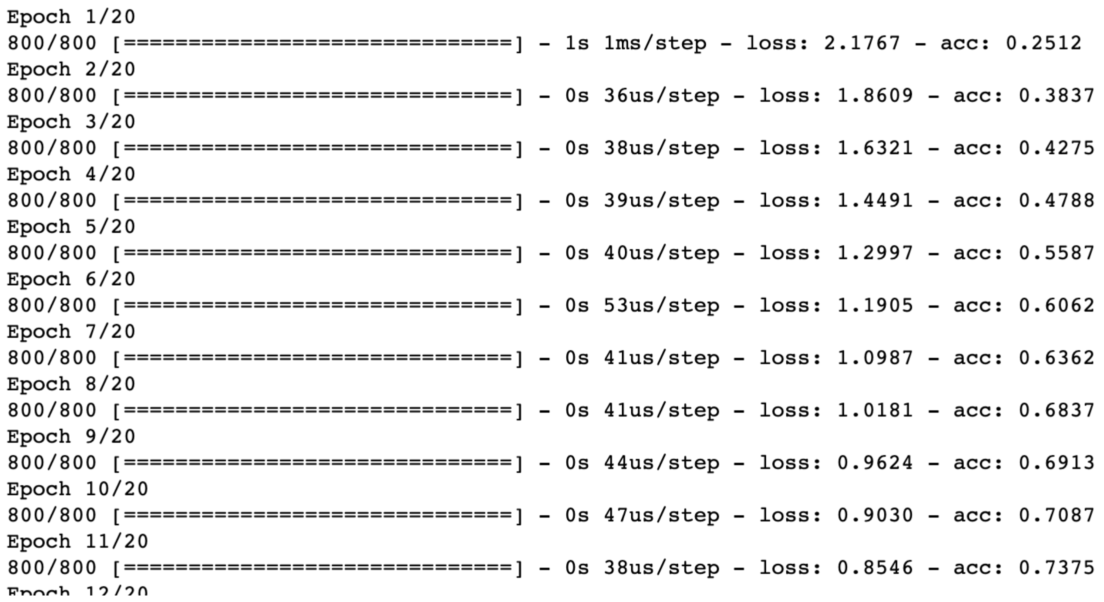
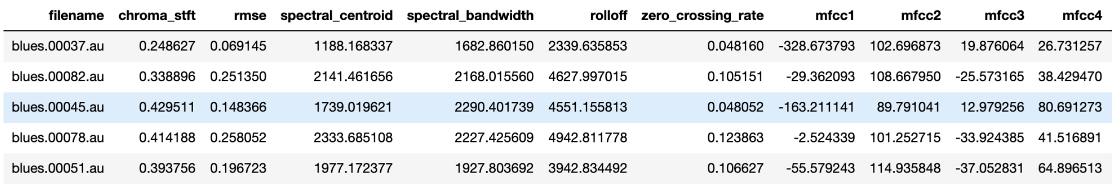

```
Epoch 1/20
800/800 [==============================] - 1s 1ms/step - loss: 2.1767 - acc: 0.2512
Epoch 2/20
800/800 [==============================] - 0s 36us/step - loss: 1.8609 - acc: 0.3837
Epoch 3/20
800/800 [==============================] - 0s 38us/step - loss: 1.6321 - acc: 0.4275
Epoch 4/20
800/800 [==============================] - 0s 39us/step - loss: 1.4491 - acc: 0.4788
Epoch 5/20
800/800 [==============================] - 0s 40us/step - loss: 1.2997 - acc: 0.5587
Epoch 6/20
800/800 [==============================] - 0s 53us/step - loss: 1.1905 - acc: 0.6062
Epoch 7/20
800/800 [==============================] - 0s 41us/step - loss: 1.0987 - acc: 0.6362
Epoch 8/20
800/800 [==============================] - 0s 41us/step - loss: 1.0181 - acc: 0.6837
Epoch 9/20
800/800 [==============================] - 0s 44us/step - loss: 0.9624 - acc: 0.6913
Epoch 10/20
800/800 [==============================] - 0s 47us/step - loss: 0.9030 - acc: 0.7087
Epoch 11/20
800/800 [==============================] - 0s 38us/step - loss: 0.8546 - acc: 0.7375
Epoch 12/20
```

Figure 4.19: ANN Running Epoch

to each genre, was calculated. Finally this Comma Separated Value (CSV) file was used to train the neural network model.



filename	chroma_stft	rmse	spectral_centroid	spectral_bandwidth	rolloff	zero_crossing_rate	mfcc1	mfcc2	mfcc3	mfcc4
blues.00037.au	0.248627	0.069145	1188.168337	1682.860150	2339.635853	0.048160	-328.673793	102.696873	19.876064	26.731257
blues.00082.au	0.338896	0.251350	2141.461656	2168.015560	4627.997015	0.105151	-29.362093	108.667950	-25.573165	38.429470
blues.00045.au	0.429511	0.148366	1739.019621	2290.401739	4551.155813	0.048052	-163.211141	89.791041	12.979256	80.691273
blues.00078.au	0.414188	0.258052	2333.685108	2227.425609	4942.811778	0.123863	-2.524339	101.252715	-33.924385	41.516891
blues.00051.au	0.393756	0.196723	1977.172377	1927.803692	3942.834492	0.106627	-55.579243	114.935848	-37.052831	64.896513

Figure 4.20: Dataset Representation.

4.2.2 Classification

As we already know that the network is trained for 1000 epochs. The confidence score is calculated from the normalized squared error value and the category is decided based on the highest 'confidence score'. Such a performance of music genre classification for different duration of music clips, relating to different genres of music, is obtained by purely trial and error method.

A Class-wise (genre) representations of the MFCC envolope are represented below, for a better understanding:

```
In [9]: display_mfcc('genres/hiphop/hiphop.00098.au')
```

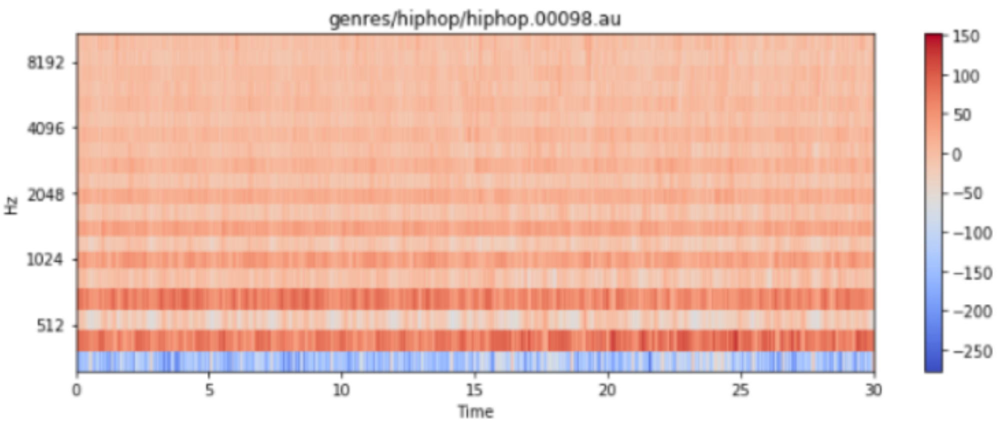


Figure 4.21: Hip Hop Song Envelope.

```
In [10]: display_mfcc('genres/hiphop/hiphop.00028.au')
```

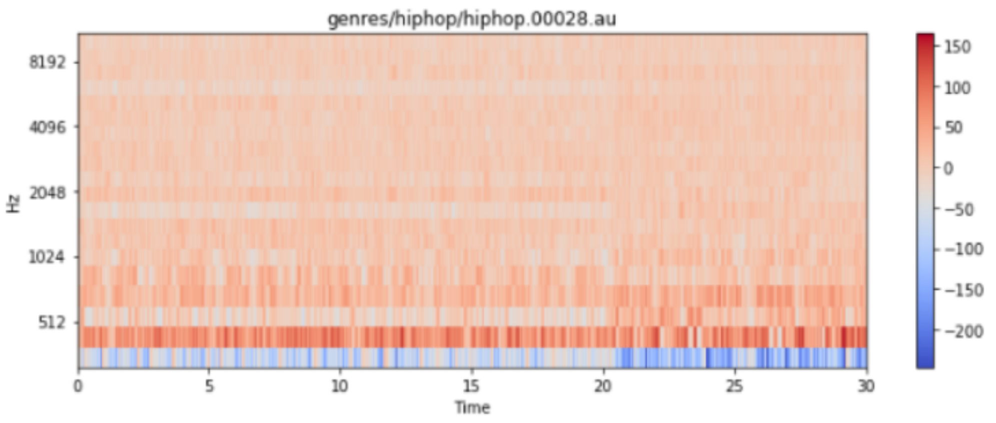


Figure 4.22: Hip Hop Song With Vocals Envelope.

```
In [6]: display_mfcc('genres/disco/disco.00070.au')
```

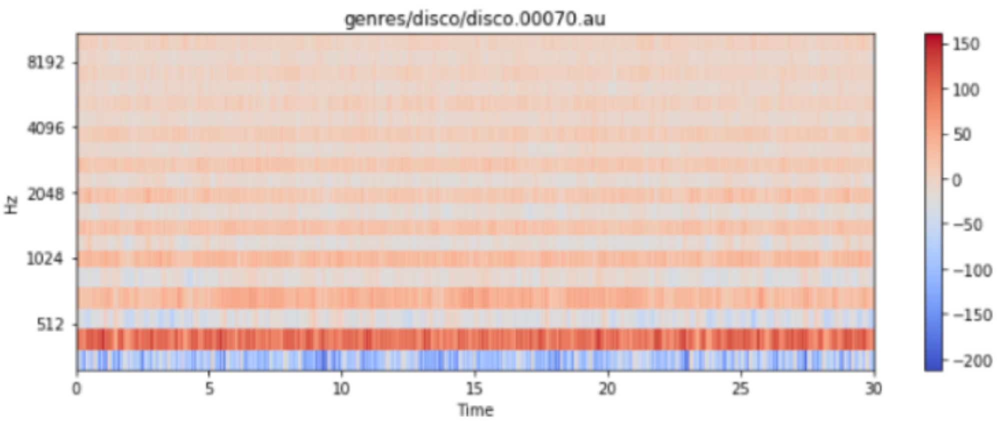


Figure 4.23: Disco Song Envelope.

```
In [7]: display_mfcc('genres/classical/classical.00067.au')
```

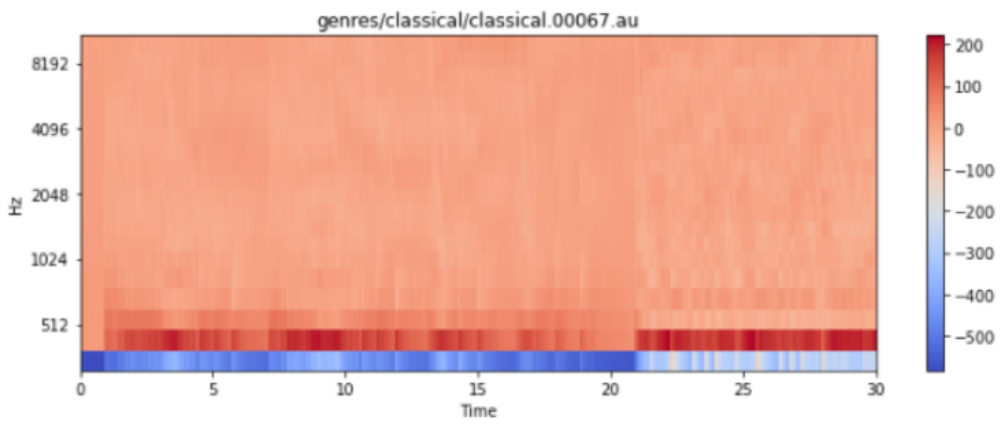


Figure 4.24: Classical Song Envelope.

4.3 Gain Optimisation - Limiter

From the recording of the sound from the source, with each superseding generation, the amplitude of the signal is defined by the gain settings. It is nothing but the volume of the recorded audio or a "Computer Generated Sound" (CGS). A gain too low will require the listener to increase the volume to a limit that the audio will start to distort and after normalisation, will start clipping, leading to a bad audio. Similarly a gain too high will lead the listeners to manually reduce the volume, which might lead to inconsistent audio. Hence, an optimized gain can not only prevent inconsistencies or ear damage, but also makes the audio clean and distortion free, thereby ensuring a pleasant listening experience for the listener, in a nutshell.

What this means is that a limiter controls the system volume levels of the audio signal. It does so by cutting the signal off at a certain peak value; by manipulating the amplitude property of the signal. Therefore, the gain of the entire audio is smoothly decreased when it gets out of proportion, or excessively louder, and is amplified back to an appropriate gain level ensuring that it does not exceed the threshold any more.

When we start to dissect an analog limiter, it turns out to be a combination of multiple resistances and capacitors as shown in the figures 4.24 & 4.25.

The limiter's task, in this system, is to amplify the gain of the system to an appropriate level by making the use of best possible fourier transform equations. The limiter's

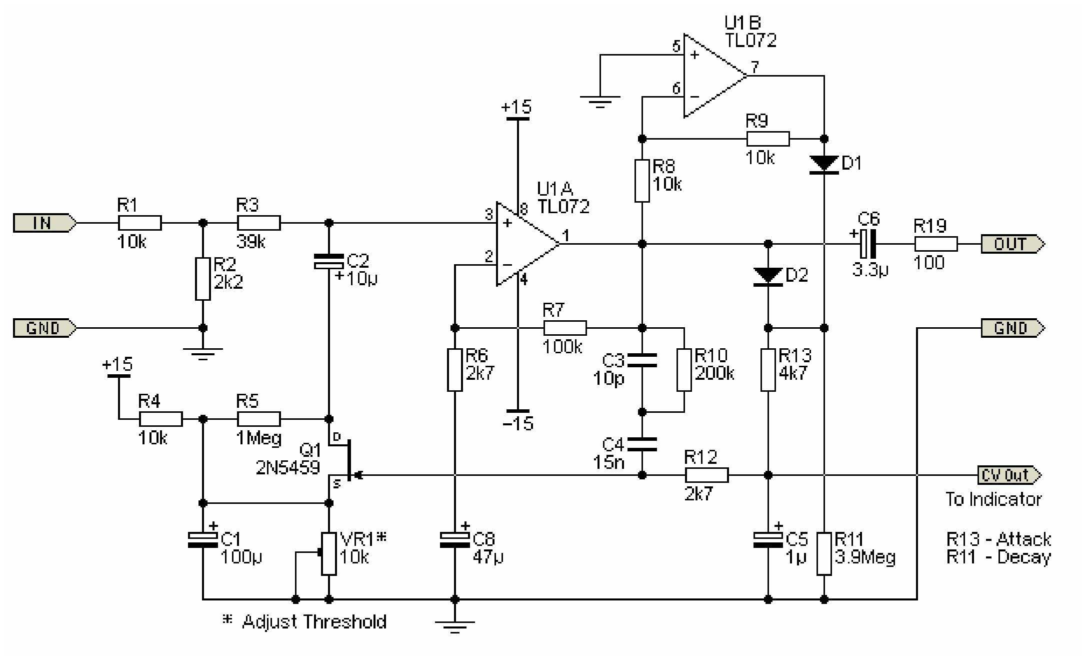


Figure 4.25: An analog limiter (a)

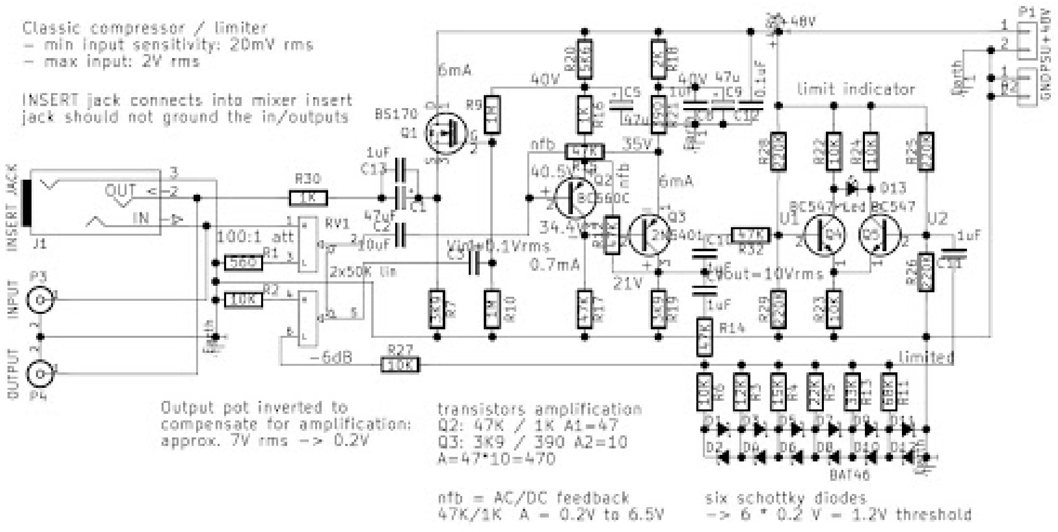


Figure 4.26: An analog limiter (b)

threshold is calculated only in the cases where there is a definite need for amplification, else it bypasses this process and jumps to the compression process.

The system uses unsupervised learning algorithms to automate the process of gain optimization. The reason behind using unsupervised learning algorithms is because each audio is a different entity and can be treated as a different constraint. Therefore, the output to each new audio will be an optimized audio of that audio, and there is a very slim possibility that the audios will have the same properties. Hence, there is no fixed output to this procedure.

```
def limiter(x, treshold=0.7):
    transfer_len = 1006
    transfer = np.concatenate([ np.repeat(-1, int(((1-treshold)/2)*transfer_len)),
                                np.linspace(-1, 1, int(treshold*transfer_len)),
                                np.repeat(1, int(((1-treshold)/2)*transfer_len)) ])
    return apply_transfer(x, transfer)
```

Figure 4.27: Code Snippet For Limiter.

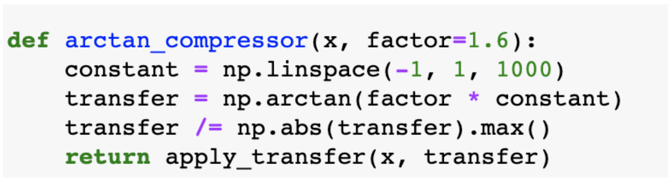
4.4 Compression Optimisation

Compression (standard lossless audio compression) is a gentle massaging or sculpting of sound to smooth out the difference between loud and soft sections. Compression is useful for preparing audio files for encoding. Streaming formats, after being piped through online networks, tend to sound better if the pre-encoded source audio is all about the same volume. A waveform editor is the tool to use here. Audio gurus spend lifetimes understanding the nuances of perfect compression. But this is only streaming audio, so don't stress it. Most waveform editors provide a few basic settings. Try a little bit of compression on your audio file, prior to the encoding process.

The factor which is customary to all the compressors is a Threshold, which is instrumental to set a tapered end, a point, above which all the dynamic changes will occur. By stationing the *Threshold point* in the center of this linear amplification inclination, the silent sections of the signal will remain uncompressed, while the wider section will have its dynamics compressed.

The expanse of the level of the audio above the Threshold Point, which is minimized in amplification, is set by a ratio. A 2:1 ratio will reduce the volume of the signal to half of the original signal, whereas a 4:1 ratio reduces the volume of the signal to the quarter of the original. This implies that the higher the ration, the more there is a reduction in the volume level. This means that the comprehensive dynamic range of the signal changes. Despite the fact that the initial dynamic range is the summation, the new dynamic ranges are significantly lesser.

To do the same we make the use of an ArcTan-Compressor, which is a compressor with a wider dynamic range as it uses the coordinates of the wave (which is plotted on a 2D. ArcTan is the inverse tangent function. For any real number a, ArcTan[a] represents the radian angle measure such that. The two-argument form ArcTan[a,b] represents the arc tangent of b/a, taking into account the quadrant in which the point lies. The script makes use of the linspace property of Numpy, which is a tool in Python for creating numeric sequences. This helps in dividing the time series into a fixed number of equidistant and evenly spaced sequences. We rooted this ’fixed number’ to be 1000, in order to account for the shortest of the time periods.



```python
def arctan_compressor(x, factor=1.6):
    constant = np.linspace(-1, 1, 1000)
    transfer = np.arctan(factor * constant)
    transfer /= np.abs(transfer).max()
    return apply_transfer(x, transfer)
```

Figure 4.28: Code Snippet For Compressor.

Then the system calculates the maximum absolute transfer, which is the ArcTan value of the ’linspace’ multiplied with a ’Factor’ which is the constant value which is defined by and changes based on the genre of the audio. This transfer is then used to maximise the effect of compression.

CHAPTER 5

SYSTEM TESTING

5.1 Objective Tests

To validate the algorithm of the system, a 16-bit wav-file audio was taken and tested. It is important to note that the script too produced a 16-bit wav-file audio. This output was compared to a digitally mastered audio in a DAW, called Ableton Live 10. The resultant output audios from both the procedures were carefully analyzed using a python script and represented graphically. It is also important to know that the process shows that the 2 outputs were 98.2% identical, when sent to a professional sound engineer.

Table 5.1: Similarity Index

Sr. no.	*Genre*	*similarity %*
1	Rock	98.2
2	Blues	94.66
3	Electronic	90.8
4	Jazz	88.2
5	Hip-Hop	87.66

This data only shows that the system performs extremely well for indie genres like rock and blues, and moderately well for electronically produced and mixed genres like Hip-Hop and Jazz.

5.2 Subjective Testing

The auditory tests were conducted in two phases. The first phase included the system being carefully critiqued by producers, musicians and sound engineers. The next phase indulged in the testing done by music enthusiasts.

5.2.1 Phase I reviews:

The first phase concerned the examination of the general impression of the 16-bit output WAV-file, which was previously used in objective tests, by engineers, producers and musicians and the table II shows the data of the general impressions of these engineers, producers and musicians.

Table 5.2: General Impressions in Phase I

Sr. no.	***Category***	***No***	***Yes***
1	Producers	3	17
2	Musicians	5	35
3	Engineers	2	22

[a]The yes and no are the votes, if the people found this a useful tool

5.2.2 Phase II reviews:

The first phase concerned the examination of the general impression of the 16-bit output WAV-file, which was previously used in objective tests, by music enthusiasts and the table III shows the data of the general impressions of these enthusiasts.

Table 5.3: General Impressions in Phase II

Sr. no.	***Category***	***No***	***Yes***
1	Enthusiasts	2	43

[a]The yes and no are the votes, if the people found this a useful tool

5.3 Accuracy Testing of Neural Networks

Neural networks are those learning algorithms which render the state of the precision and validity in many use cases. However, initially the accuracy of the network built was simply not satisfactory. This was due to the fact that the model was memorizing the training values rather than learning from them. Thus, giving rise to higher loss and failure percentage. Also, at this point, choosing an optimal learning curve became an important factor to determine whether the network converged to its global minima or not. A higher learning rate almost never got the global minima. Therefore, to include this global minima, more iterations were done, which was a lengthy procedure but increased modularity of the model. Using ANN, the system was trained to identify and quantitatively analyze the hyper parameters (such as loss function, accuracy function, etc.). Each time the system passed all the test cases, the number epochs increased, and so did the accuracy of the system. Initially, the system had incurred higher losses and low accuracy, but as the number of epochs were increased, the weight was repetitively updated; and hence the curve went from under-fitting to optimal.

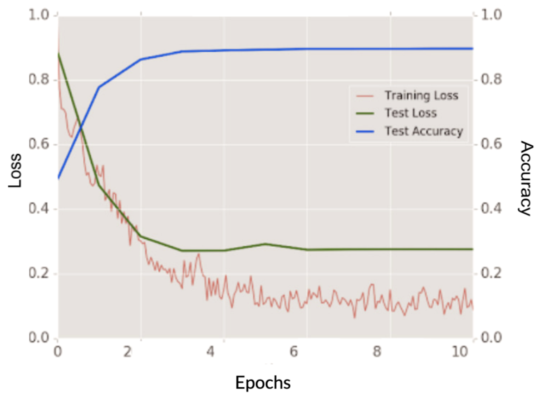


Figure 5.1: Training Curve for Neural Network.

Hence, as shown in graphs, as the training epochs increased, the accuracy went from 50% to approximately 87%. After which there was a negligible change in the test accuracy of the system.

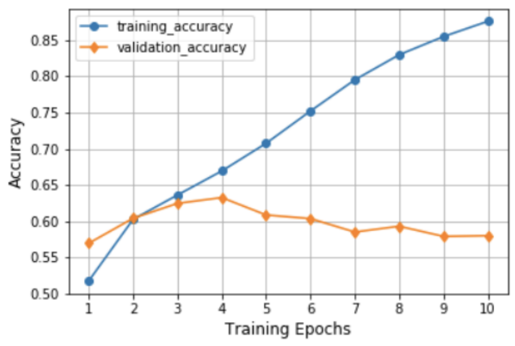


Figure 5.2: Accuracy of the Neural Network.

CHAPTER 6

SYSTEM APPLICATIONS

6.1 Post-production Enhancements

The system shows a huge potential to be used in modules linking to post production mastering practices by producers for amplitude and dynamics correction. The traditional method of doing so requires a huge amount of finance, training, effort and most importantly time. A system, such as ours, has the potential to automate a part of this process reducing time, effort and cost.

6.2 Surveillance audio amplification

In crime investigation units, while investigating using surveillance, audio plays an important role in every aspect of the investigations. A feeble audio might not be able to provide for the details that were required for progressing the case, thus, obstructing the flow of the investigations. Often these units have to reside to studio engineers to amplify the sound and thus the government has to pay an unusually high price from the treasury. This system can cut down that cost to zero, since we reside to an absolute non profitable means for government agencies and researchers.

6.3 Medical Applications

This system has a potential to be connected to medical instruments as an enhancement interface so as to give boost to feeble audios such as pulse. Applications such as ECG can be easily converted to its sinusoidal form and can be analysed and amplified using FFT. This can be a revolutionary technology in terms of medical applications, and can also reduce the hefty processing of ECG to half. It can also be used to make the hearing aids more digitised and robust.

6.4 Flight data Amplifications

The flight data used by Air Agencies help them in pilot training, crash investigations and other amenities, residing to the audio for revealing important details. When the flight is out of communicable range, then the audio starts to fade and hence in certain emergencies this communication gap turns out to be disastrous.

CHAPTER 7

CONCLUSION

The work presents the implementation of automation in audio enhancement and optimisation using fast fourier transform, as an application of unsupervised learning and neural networks, in a ubiquitous computational environment which accepts an audio as an input: an audio to be processed. Certain characteristics of the audio are identified: brightness, roll-off factor, Mel Frequency Cepstral Coefficient (MFCC), amplitude-time graph, etc.. The system was implemented as a separate software running specific scripts written in the python environment.

At this point, it is important to mention the demerits of the traditional mastering techniques, which are highly subjective to the audio content, This is basically done in studios with a combination of software and hardware. This hardware and software requires highly trained engineers to be operated because any slight mistake can completely alter the correctness of the previously performed recording and mixing phases.

The sound engineers, by carrying out mixing and mastering processes, often rely on their own experience or might follow the trial and error method. The trial and error method is carried out by adjusting the parameters of the equalization and compression algorithms and checking the effect by listening to the processed signal. The paper, however, focuses on automating the lengthy process of optimizations and enhancements. The result of the system that performs optimizations and enhancements automatically is, additionally, dependent on the appropriate selection of the reference material and a proper analysis on the initial audio. The classification of music serves as the fundamental step towards rapid growth and is useful in music indexing. For the purpose of genre classification, the system makes use of neural networks. For the aim of enhancing audio, signal processing libraries such as “numpy” and “scipy”, which are available in python programming environments, are used. The paper also showcases the power of machine learning algorithms and certain fast fourier transform equations, that are instrumental in limiting and compression, which makes it the backbone of the whole system.

CHAPTER 8

FUTURE ENHANCEMENT

The authors of this research are further trying to enhance the research by trying to make the genre prediction recognise more genres, including experimental genres. Also, the python modules for equalizer, depth, pitch correction and multi-band dynamics are under research.

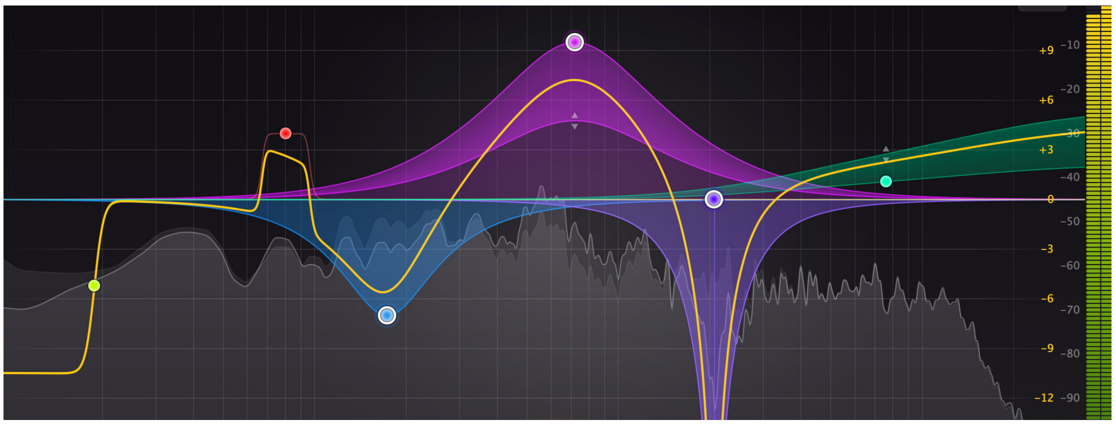


Figure 8.1: Equalizer.

APPENDIX A

FAST FOURIER TRANSFORM

A.1 Sampling Rate

sr is technically defined as the number of audio samples being carried per second. It is measured in units such as Hertz (Hz) or kilo-Hertz (kHz). In this research we set the sr = 44.8 kHz, in order to attain a high quality output.

We can also determine the sampling rate by the following equation

$$\boxed{fs/2} \tag{A.1}$$

where sampling rate fs= number of samples/ sampling time.

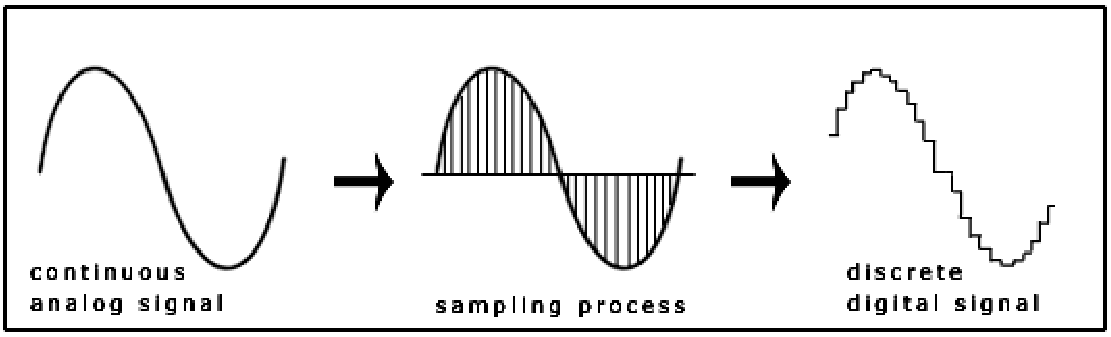


Figure A.1: Effect of sr.

REFERENCES

[1] D. S. Eley, The Audio Mastering Blueprint. TGMaudio, 2013.

[2] M. Piotrowska, S. Piotrowski, and B. Kostek, “A Study on Audio Signal Processed by ‘Instant Mastering’ Services,” in Proc. AES Convention, 2017.

[3] R. Izhaki, Mixing Audio: Concepts, Practices and Tools. Elsevier, 2008.

[4] E. Najduchowski, M. Lewandowski and P. Bobinski, "Automatic Audio Mastering System," 2018 Joint Conference - Acoustics, Ustka, 2018, pp. 1-6.

[5] Audio post-processing and identification based on audio features by Yunzhen Khan and Xiochan Yuan. Proceedings of 2017.

[6] Thiruvengatanadhan, R. (2018). Music Genre Classification using MFCC and AANN. International Research Journal of Engineering and Technology (IRJET).

[7] T. Viarbitskaya and A. Dobrucki, "Audio processing with using Python language science libraries," 2018 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Poznan, 2018, pp. 350-354.

[8] Venugopal, K., & Madhusudan, P. (2017). Feasibility of music composition using artificial neural networks. 2017 International Conference on Computing Methodologies and Communication (ICCMC).

[9] GiANNoulis, Dimitrios, Michael Massberg and Joshua D. Reiss. “Parameter Automation in a Dynamic Range Compressor.” (2013).

[10] Chun-Chi J. Chen and Risto Miikkulainen (2001). Creating Melodies with Evolving Recurrent Neural Networks. International Joint Conference on Neural Networks (IEEE).

[11] Viarbitskaya, T., & Dobrucki, A. (2018). Audio processing with using Python language science libraries. 2018 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA).

[12] Scheibler, R., Bezzam, E., & Dokmanic, I. (2018). Pyroomacoustics- A Python Package for Audio Room Simulation and Array Processing Algorithms. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[13] J.-P. Briot, G. Hadjeres and F.-D. Pachet, Deep Learning Techniques for Music Generation, Computational Synthesis and Creative Systems, Springer, 2019.

[14] Jijun Deng, Wanggen Wan, XiaoQing Yu, Xueqian Pan, & Wei Yang. (2011). Audio fingerprinting based on harmonic enhancement and spectral subband centroid. IET International Communication Conference on Wireless Mobile and Computing (CCWMC 2011).

[15] Glover, John C., Victor Lazzarini and Joseph Timoney. "Python for audio signal processing." (2011).

[16] Wickert, Mark. (2018). Real-Time Digital Signal Processing Using pyaudio_helper and the ipywidgets. 91-98.

[17] McFee, Brian & Raffel, Colin & Liang, Dawen & Ellis, Daniel & Mcvicar, Matt & Battenberg, Eric & Nieto, Oriol. (2015). librosa: Audio and Music Signal Analysis in Python.

[18] Bob Katz and Robert A. Katz. 2003. Mastering Audio: The Art and the Science. Butterworth-HeinemANN, USA.

[19] Bahuleyan, Hareesh. (2018). Music Genre Classification using Machine Learning Techniques.

[20] Hilsamer, Marcel and Stephan Herzog. "A Statistical Approach to Automated Offline Dynamic Processing in the Audio Mastering Process." DAFx (2014).

[21] Ronan, Malachy & Ward, Nicholas & Sazdov, Robert. (2016). The Perception of Hyper-Compression by Untrained Listeners.