# MUSIC GENRE CLASSIFICATION USING SPECTRAL ANALYSIS TECHNIQUES WITH HYBRID CONVOLUTION-RECURRENT NEURAL NETWORK

***Dissertation***

***Submitted***

***In partial fulfilment for the Degree of***

**Master of Technology**

**In**

**Computer Engineering**

***Submitted By***

**SAHIL (17MCS016)**

UNDER THE SUPERVISION OF

**MR. FAIYAZ AHMAD**

(Assistant Professor)

**Department of Computer Engineering**

**F/O Engineering & Technology**

**Jamia Millia Islamia, New Delhi-110025**

**2019**

# CERTIFICATE

This is to certify that the project report entitled **Music Genre Classification Using Spectral Analysis Techniques With Hybrid Convolution-Recurrent Neural Network** submitted by **Sahil** to the Department of Computer Engineering, F/O Engineering & Technology, Jamia Millia Islamia New Delhi-110025 in partial fulfilment for the award of the degree of **M. Tech (Computer Engineering)** is a *bona fide* record of project work carried out by my supervision. The content of this report, in full or in parts, have not been submitted to any other Institution or University for the award of any degree.

| | |
|---|---|
| **Mr. Faiyas Ahmad** | **Prof. Tanvir Ahmad** |
| Assistant Professor | Head of Department |
| Dept. of Computer Engineering | Dept. of Computer Engineering |
| Faculty Of Engg. and Technology | Faculty of Engg and Technology |
| Jamia Millia Islamia | Jamia Millia Islamia |

# DECLARATION

I declare that this project report title **Music Genre Classification Using Spectral Analysis Techniques With Hybrid Convolution-Recurrent Neural Network** submitted in partial fulfilment of the degree of **M. Tech (Computer Engineering)** is a record of original work carried out by me under the supervision of **Mr. Faiyaz Ahmad** and has not formed the basis for the award of any other degree, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgments have been made wherever the findings of others have been cited.

SAHIL

17MCS016

Dept. of Computer Engineering

Faculty of Engineering and Technology

Jamia Millia Islamia

New Delhi-110025

Email: sahiltinky21@hotmail.com

# ACKNOWLEDGMENTS

I would like to thank our mentor, Mr. Faiyaz Ahmad (Assistant Professor, Dept. Of Computer Engineering) for providing us with the opportunity to work on this project. His suggestion is visualizing the project and sustained interests to attain the objectives envisaged are gratefully acknowledge. His invaluable guidance and innovative ideas immensely helped me to surmount difficult situations to bring successful completion of the project.

I also appreciate our Faculty of Engineering and Technology and Jamia Millia Islamia for providing me the work culture and environment that has given me the feel of a professional corporate life. Special thanks towards all our teachers, friends and family for their invaluable support.

SAHIL

17MCS016

Dept. of Computer Engineering

Faculty of Engineering and Technology

Jamia Millia Islamia

New Delhi-110025

Email: sahiltinky21@hotmail.com

# ABSTRACT

Today we know, data are increasing exponentially over the years. So, result in analysis of the data are getting difficult and taking too long time to get task done. Specifically, for audio data, which contains large amount of sampling data (around 660,000) for duration of 30 seconds. In the real-time, like Saavn, which have over 5 crore audio data or music and all audios have also been categorized into different genre. Here, Data play vital role as a computer science or data science. In this work, the objective is to classify the audio data into different genres from GTZAN dataset which contain about 10 genres. We, first, perform the audio segmentation to make it signal into homogeneous content. Short-term Fourier Transform (STFT), Mel-spectogram and Mel-frequency cepstrum coefficient (MFCC) are the most common feature extraction technique and each feature extraction technique has been successful in their own various audio applications. Then, these feature extractions of the audio fed to the Convolution Neural Network (CNN) model and VGG16 Neural Network model, which consist of 16 convolution neural network. We perform different feature extraction with different CNN and VGG16 model with or without Recurrent Neural Network and evaluated performance measure. In this model, we have achieved overall accuracy 95.5% for this task.

Keywords– GTZAN, Short-term Fourier Transform (STFT), Mel-spectrogram, Mel-frequency cepstrum coefficient (MFCC), Convolution Neural Network, VGG16, Recurrent Neural Network (RNN)

# TABLE OF CONTENTS

# LISTS OF FIGURES

# LIST OF TABLES

# CHAPTER 1

# INTRODUCTION

## 1.1 Introduction

“Music” is also considered as a part of life. Whenever people work, or coming from office, or any other activity, the music gives us to relax our mind and make our soul to high spirit. Music also make us creativity and motivation when people are in sad mood or depression. Once you start listening to music, all the pain or stressing on our brain make you forget them and bring you into high spirit like confidence, no feeling stress, peaceful while mediation, etc.

Today, you will find music websites or app from smartphone like Saavn, Gaana, JioSavan, etc contains 50 million and with 15 different languages present in it. Most of the people are also remaking the combination of two songs, which called as MASHUP, to make to mix of these songs. So, more music or audios data are creating for the entertainment to the people. So, if we make a different genre for every music as which type of nature of songs it is. It would help people to find that song or genre they want to listen. Without that, it’s would never know what type of song or music it is, like is it mediation or pop songs? To do that, we listen to every music and make a list of genre they belong, which is very time consuming. So, Music Genre also play important of the life and entertainment to the people which can enjoy with the fullest.

Music Information Retrieval (MIR) comes to play to analyse and process with the audio data. Many applications are required in, like, music recommendation, automatic music genre classification, searching of music. But as a core part of MIR, genre classification are the most one needed to performed as it focus on assigning the unknow audio into specific genre.

“Genre”, as per dictionary, means categories of music. It also important to categorize the music into different part of music that they belong. For example, if you are in party, the music will be played related to rock or jazz genre. If you are doing aerobics or exercise part, then music will be played related to theme genre. So, music genre brings us which nature of music we like to listen.

## 1.2 Motivation

With the advanced artificial intelligence, deep learning has shown better performance in music or voice classification. Automatic Speech Recognition (ASR), for example, OK Google in Android phone or Siri in IOS phone can recognize and understand what you are speaking.

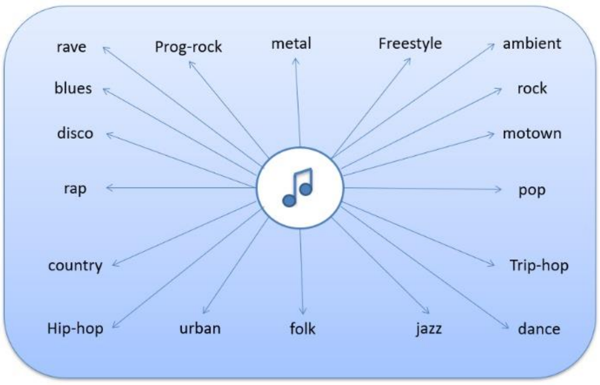


Fig 1.1 Various Music Genres

Researchers had applied different proposal algorithm and techniques to solve this kind of problem which can put it in deployment production. Spectrogram of audio data has proved and has considered to be the one very effective feature extraction techniques than others. For the classifier algorithm, several algorithms are common to work in this field like Gaussian Mixture Model (GMM), Support Vector Machine (SVM), etc. Convolution Neural Network (CNN) had shown the prominent result with this spectrogram features as it feature taken into consider as an image.

## 1.3 Objective

1. To design a model on given music data which can classify to specific music genre.

2. To analyse the characterize on different music signals using Spectral Analysis.

3. To improve the performance measure.

## 1.4 Framework Used

Python programming language was chosen to develop for this work. Librosa is a python package tool used for analysing and processing in audio. Keras is a framework used in deep learning as it provides an abstraction layer on TensorFlow or Theano. Keras can be used in both Python 2.7 and 3.5. The Keras provides modularity, minimalism, and extensibility.
The networks were trained and evaluated using Python Script and run using PowerShell. My system trained with the support of Nvidia GeForce GTX 1050 Ti GPU which will perform faster than CPU to reduce training time.

## 1.5 Organization of Thesis

**Chapter 2:** We describe the relevant work which that had been done before with different dataset, techniques and learning algorithms.

**Chapter 3**: We describe the most powerful neural network which had been achieved in recent years in image classification and image recognition is Convolution Neural Network (CNN) and Also explain the Recurrent Neural Network (RNN) which to take the idea of how human tried to remember the memory and connected with the next ones. We explained three types of RNN are Long Term Short Memory (LSTM), Gated Recurrent Unit (GRU) and Bi-Directional RNN.

**Chapter 4:** It shows the experimental result of this work with explained in every performed stage.

# CHAPTER 2

# LITERATURE SURVEY

Various papers have been published regarding the music genre classification. These techniques primarily used various feature extraction methods and then apply classifiers to classify the audio into specific genre. Some of them used Machine Learning and Deep Learning-Based approaches.

**Caifeng Liu  et al.** [1], they proposed new develop convolution neural networks architecture which takes the long related to context of information into considerations and transfers further suitable information to decision-making layer. To develop the new neural network, they used the idea of Inception model by combine convolutions with different kernel size and layers. So, the dataset they used were GTZAN, Ballroom and Extended Ballroom. The pre-processing steps done to extract feature called mel-spectogram with 128 Mel filters from audio signal. Then this feature input of size 647x128 fed to BBNN. They achieved overall accuracy 93.9%, 96.7%, 97.2%.

**Prasenjeet Fulzele et al.** [2], they implemented this proposed model into two parts. First, features were extracted from the audio files and arranged them such a manner that they fed to two models. Secondly, they involved fusion of two models by sum rule. To predict the final prediction, the separate posterior probabilities have combined them to get result. In this work, the dataset they used GTZAN dataset. They were extracted 9 features from audio file in order to see the audio pattern by machine learning algorithm. Now these inputs are trained into two model individually and then merge them to get final prediction result. Hyperparameter was used by Random Grid Search. They achieved with the overall accuracy of 89% by combining SVM and LSTM model.

**Nilesh M. Patil et al.** [3], they had done with the idea of applying Machine Learning Techniques. SVM and k-NN are the most frequent applied machine learning techniques in many fields like image processing, twitter sentimental analysis, etc. In this work, the dataset they used GTZAN. To obtain the feature extraction from the audio file, it’s done by Mel-Frequency Cepstral Coefficients (MFCC), then feature vectors are obtained. Now, these feature vectors are classified with supervised learning approaches i.e. k-NN, Linear SVM and Poly SVM. Each classifier have achieved overall accuracy with 64.4%, 60% and 77.78%.

**Ahmed Elbie et al.** [4], the GTZAN dataset is used and perform feature extraction is statistical descriptors. They extracted about 8 features i.e. zero crossing rate, spectral centroid, spectral contrast, spectral bandwidth, spectral roll off and MFCC. Each feature is evaluated with different machine learning techniques and then also same techniques was done with combined features one. It found that SVM had perform better with combined features which achieved 72%. Further, it extended to work with applying deep learning algorithm in audio which was Convolution Neural Network (CNN). Before fed to the network, they perform with 3 parts: raw data, STFT with hop length 1024 and window size 2048 and MFCC with 13 coefficients. It found that it achieved 66% accuracy.

**Pradeep Kumar D et al.** [5], they study to detect and classify GTZAN audio files and then compare them by using various classification algorithms. For feature extraction, they used Fast Fourier Transform (FFT) and Mel Frequency Cepstral Coefficients (MFCC). Logistic Regression, Support Vector Machines, Decision trees, K-NN, Recurrent Neural Network were used as the classifier algorithms. When apply Recurrent Neural Network, they achieved highest accuracy among the others with 86%.

**Anshuman Goel et al.** [6], they used 8 feature extraction from the audio file, that is, beat periodicity, loudness, energy, speechness, acousticness, valence, danceability, discrete wavelet transform (DWT). Then these features are fed to the neural network. They done with the 2 genres only, Classical and Sufi songs. They achieved with the accuracy of 85% testing. Classical songs correctly predicted with overall 87% of data and Sufi songs correctly predicted with overall 82% of data.

**Jan Jakubik** [7], they performed with two Recurrent Neural Network (RNN): LSTM and GRU. They experiment on 4 datasets: GTZAN, Emotify, Ballroom and LastFM. The spectrogram were used as the feature extraction. It compares all datasets with LSTM and GRU train and test. The GRU has been performed better than LSTM with the overall 92% accuracy and for LSTM, the overall accuracy 89% accuracy.

**George Tzanetakis** [8], three feature set are applied, i.e., timbral texture, rhythmic content and pitch content and the statistical pattern recognition classifier is used , i.e.. Gaussian Mixture Model (GMM), k-Nearest neighbor (kNN) and simple Gaussian (GS). They used GTZAN dataset and real time-based music. It has achieved with overall 61% accuracy to real time music.

**Lin Feng et al.** [9], they proposed hybrid model which consist of Convolution Neural Network and Bi-RNN parallel to other. GTZAN dataset used in this paper. They used short term fourier transform feature extraction technique with frame number 1024 with 50% overlapping. It proved that CNN working RNN can improved with achievement of 92% accuracy.

Table 2.1 Summary of Literature Survey

| **S.No** | **Paper Title with Author Name** | **Year** | **Dataset** | **Feature extraction and Learning Algorithm** | **Accuracy** |
|---|---|---|---|---|---|
| 1 | Bottom-Up Broadcast Neural Network For Music Genre Classification by *Caifeng Lui, Lin Feng, Guochao Liu, Huibing Wang* [1] | 2019 | GTZAN, Ballroom, Extended Ballroom | Feature Extraction: Melspectrogram, Classifier: BBNN | 93.9% |
| 2 | A Hybrid Model For Music Genre Classification by *Prasenjeet Fulzele, Rajat Singh, Naman Kaushik, Kavita Pandey* [2] | 2018 | GTZAN | Feature Extraction: SSD, MFCC, Spectral rolloff, zero crossing rate, chroma frequency, rhythm histogram, | 89% |

| | | | | | |
|---|---|---|---|---|---|
| | | | | spectral centroid, Classifier: SVM, LSTM | |
| 3 | Music Genre Classification using MFCC, kNN and SVM classifier by *Nilesh M. Patil, Dr. Milind U. Nemade* [3] | 2017 | GTZAN | Feature Extraction: MFCC, Classifier: SVM, kNN | 77% |
| 4 | Music Genre Classification and Recommendation by using Machine Learning Techniques by *Ahmed Elbir, Hilmi Bilal Cam, Mehmet Emre Iyican, Berkay Ozturk, Nizamettin Aydin* [4] | 2018 | GTZAN | Feature Extraction: Zero crossing rate, spectral centroid, spectral contrast, spectral bandwidth, spectral rolloff, MFCC (13 coeff), MFCC derivation, kNN, RF, NB, DT, SVM, CNN | 72% |
| 5 | A comparative study of classifiers for Music Genre Classification based on Feature | 2016 | GTZAN | Feature extraction: FFT and MFCC Classifier: Logistic Regression, kth | 86% |

| | | | | | |
|---|---|---|---|---|---|
| | Extractors by *Pradeep Kumar D, Sowmya B J, Chetan, K G Srinivasa* [5] | | | Nearest Neigbor, SVM, Decision Tree | |
| 6 | Genre Classification of Songs Using Neural Network by *Anshuman Goel, Mohd. Sheezan, Sarfaraz Masood, Aadam Saleem* [6] | 2014 | Classical and Sufi | Feature extractions: beats periodicity, loudness, energy, Speechiness, Acousticness, valence, danceability, DWT Classifier: NN | 85% |
| 7 | Evaluation of Gate Recurrent Neural Network in Music Classification Tasks by *Jan Jakubik* [7] | 2017 | GTZAN, Emotify, Ballroom, LastFM | Feature Extraction: Spectrogram Classifier: LSTM, GRU | 92% |
| 8 | Musical Genre Classification of Audio Signals by *George Tzanetakis* [8] | 2002 | GTZAN | Feature extraction: timbral texture features, timbral texture feature vector, rhythmic content features | 61% |

| | | | | | |
|---|---|---|---|---|---|
| | | | | Classifier: GS, GMM, kNN | |
| 9 | Music Genre Classification with Parallelling Recurrent Convolution Neural Network by *Lin Feng, Shenlan, Liu, Jaining Yao* [9] | 2017 | GTZAN | Feature extraction: Short-term fourier transform, Classifier: PRCNN | 92% |

# CHAPTER 3

# PRELIMINARIES

## 3.1 Spectral Analysis Techniques

Spectral Analysis is to analyse the spectrum to find the pattern or properties from the source. A spectrum means to differentiate between amplitude and frequency domain. When we say spectral analysis or techniques, there is always involve with Fourier Transform or any other technique which transform into frequency domain because after applying it to the source, we get the spectrum of that source.

### 3.1.1 Short Term Fourier Transform (STFT)

As we all know, when we want to plot the waveform of any audio, it differentiates with amplitude and time. But not only time importance but also frequency. Time tells us *when we hear something,* and frequency tell us *what we heard*. The waveform of the audio is expressing as the sum of sinusoidal function, i.e., the waveform can be breaking down into sinusoidal function which is also frequency.

Suppose, we have a waveform of time domain function. This waveform shows the summation of sinusoidal function with 50 and 80 Hz frequency (in Fig 3.1).

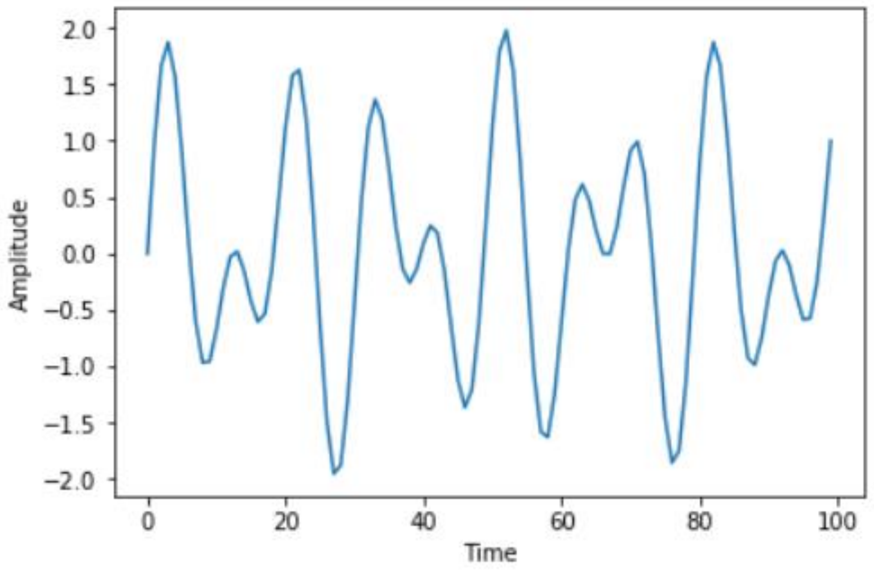


Fig 3.1 Waveform of Time Domain Function

So, in order to obtain frequency domain from time domain, this transformation is called Fourier Transform. Because analysing at frequency domain is much easier and simpler than time domain. There is another term called Discrete Fourier Transform (DFT) because we are dealing with discrete values not continuous values. By definition of DFT,

$$\boxed{X(k) = \sum_{n=0}^{N-1} x(n) * e^{-j2\pi nk/N}} \qquad \text{Eq. (3.1)}$$

And to convert time domain from frequency domain, its called inverse of DFT. By definition of IDFT,

$$\boxed{x(n) = \sum_{k=1}^{N-1} X(k).e^{j2\pi nk/N}} \qquad \text{Eq. (3.2)}$$

So, here $x(n)$ is discrete time function and $X(k)$ is Fourier transform of $x(n)$.

But looking them make it complex to evaluate with summation. Let assume that

$$\boxed{D_N = W_N^{nk} = e^{-i2\pi nk/N}} \qquad \text{Eq. (3.3)}$$

This is called twiddle factor. So, in order to compute faster, we create the generalize form after putting value $k = 0,1,2,3$ to form twiddle matrix.

$$\boxed{X(k) = D_N \times x(n)} \qquad \text{Eq. (3.4)}$$

Where $X(k)$ is a Fourier Transform of x(n) in column vector $(N \times 1)$ and $x(n)$ is discrete time signal in column vector $(N \times 1)$ and $D_N$ is Twiddle matrix $(k \times n)$ which is represented by

$$W_N^{nk} = \begin{array}{cc} \xrightarrow{\quad n = 0,1,2,3 \ldots (N-1) \quad} & \\ \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W_N & \cdots & W_N^{(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix} & \Big\downarrow \; k = 0,1,2,3 \ldots (N-1) \end{array}$$

Twiddle Factor Matrix

After finding the Fourier transform of $x(n)$, we get the result in complex number. To do that, we take the absolute value of each $X(k)$. If we apply Fourier transform for signal Fig 3.1., we get

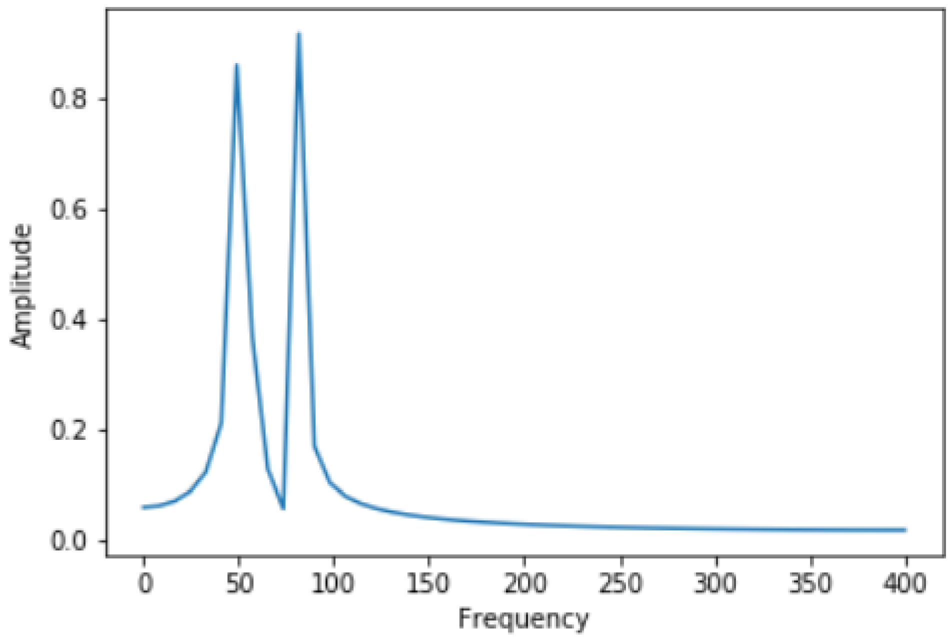


Fig 3.2 Fourier Transform of signal (in Fig 3.1)

In the Fig 3.2, we observe that two frequencies are having high amplitude and peak value which telling us that these two frequencies are present in this signal with respect to other frequencies (showing no changes). At the peak value, we found that frequencies of 50 and 80 Hz are there this signal.

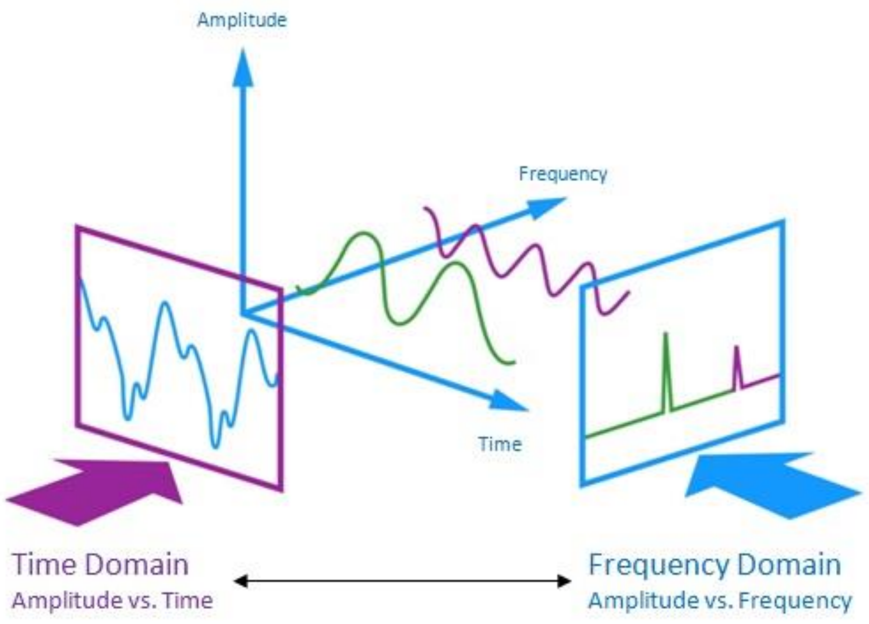


Fig 3.3 Visualization of Time Domain and Frequency Domain [20]

But there is a limitation of applying DFT that for different signal with time showing same frequency. For example, let take a chirp signal and observe the figures below

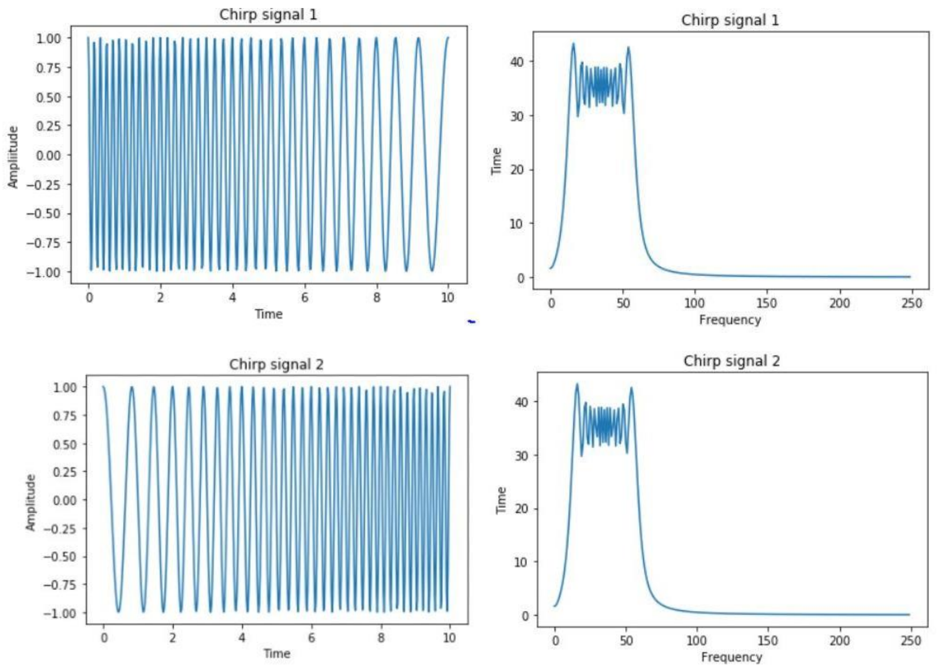


Fig 3.4 Illustrating Different Time Signal showing same Frequency

In Fig 3.4, Even though the signals are different but shows same frequency. This means that Fourier Transform only give us what are the frequency present in this signal, but it will not tell us when this component of frequency occurs.

Here we use short term fourier transform [10, 21]. As the name suggested "short term" means signal are split into small fixed duration of signal and then apply fourier transform of each block. Moving the sliding to create every block in signal called framing. This is computed frequency change with signal over time. Before computing, each block is multiplied with windowing function in order to enhance the ability of Fourier Transform to extract spectral data from signals. We used Hann window which look like bell curved shape (see function $g(t)$ below Fig 3.5).

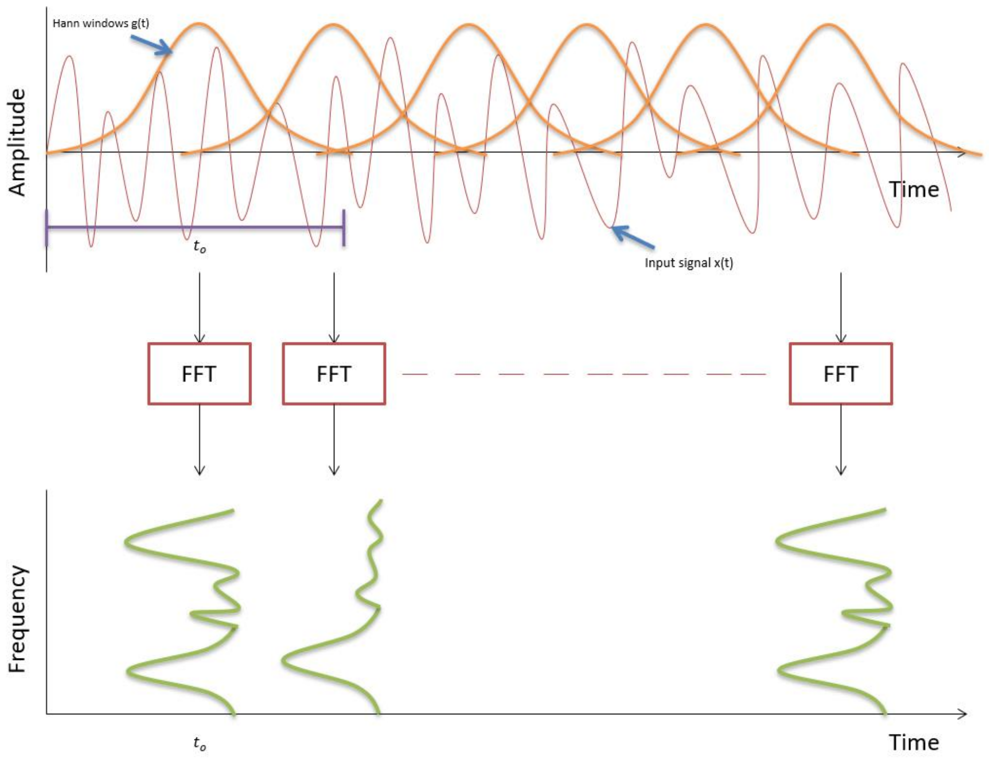


Fig 3.5 Visualization on STFT

Now after applying FFT on each block, we got frequency representation. Now this all frequency representation with time for this signal will be consider as their features which is also called as spectrogram.

## 3.1.2 Mel-Spectrogram

Mel-Spectogram [20] is the part of digital filter bank. Digital filter bank is a set of bandpass filters for a single input.

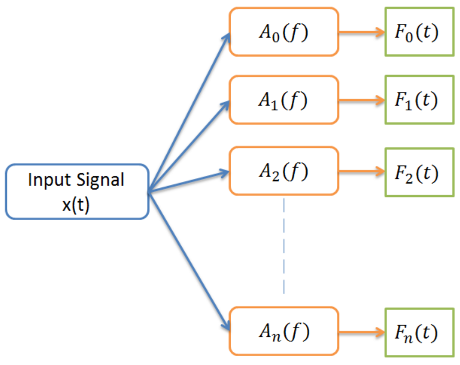


Fig 3.6 Illustration of Digital Filter Bank

From above figure, $x(t)$ is any input signal, $n$ is the number of analysis filter bank and $A_k(f)$ is analysis filters. So, by using these analysis filter bank, a signal is decomposed into a set of sub band signal $F_k(t)$ with each sub band signal occupying a portion of the original frequency band. This is called as analysis in digital filter bank.

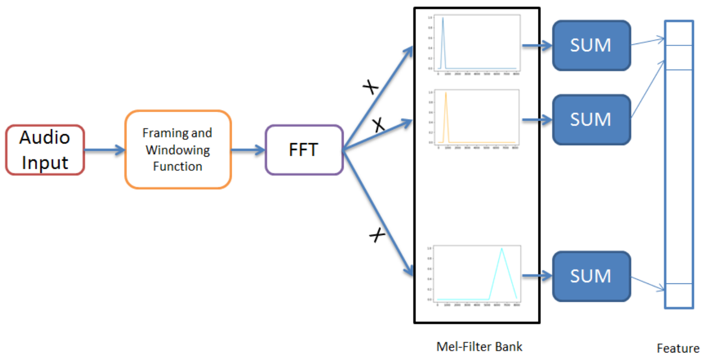


Fig 3.7 Procedure of Evaluating Mel-Spectogram [24]

In Fig 3.7, its exactly looks like analysis part of digital filter bank and Mel Filter Bank block. First thing is, we take audio as an input and then, framing the audio with window size done to get samples into the buffer block and apply Window block (e.g. Hann windows) and then perform Fast Fourier

Transform (FFT Block) which is convert time domain signal into frequency domain. This frequency domain of signal is passed through every analysis filter bank which is Mel-Spaced Filter Bank. To calculate the filter bank energy, a signal is multiplied with the filter bank and are summed up with their coefficient and every summed coefficient with $n$ filters are created vectors of Mel Spectrogram. That is, for each frame of the signal, we get $n$ vector of Mel-Spectrogram.

Now, Let see how Mel Filter Bank is done? It is calculated by

$$MFB_n(z) = \begin{cases} 0 & , if\ z < frbin(n-1) \\ \frac{z - frbin(n-1)}{frbin(n) - frbin(n-1)} & , else\ if\ frbin(n-1) \le z < frbin(n) \\ \frac{frbin(n+1) - z}{frbin(n+1) - frbin(n)} & , else\ if\ frbin(n) \le z < frbin(n+1) \\ 0 & , else \end{cases} \qquad \text{Eq. (3.5)}$$

where $n$ is number of filter bank we want and $frbin()$ is the list of Mel-Spaced frequencies and z is iteratable range from 0 to maximum frequency.

For converting frequency into mel-scale is defined as

$$mel(freq) = 1125\ln\left(1 + \left(\frac{freq}{700}\right)\right) \qquad \text{Eq. (3.6)}$$

And for vice-versa

$$freq(mel) = 700\left(\exp\left(\frac{mel}{1125}\right) - 1\right) \qquad \text{Eq. (3.7)}$$

where $freq$ and $mel$ are frequency and mel-scale value.

Let take an example, for an audio we have lower frequency of 300 Hz and upper frequency of 8000 Hz with sampling rate 8000 Hz. So, we convert these frequencies into mel-scale using equation (3.6).

$$300\ Hz \rightarrow 401.25\ Mels$$

$$8000\ Hz \rightarrow 2835.99\ Mels$$

Now between two mels scale, we divide into $m$ number of filters we want to get additional points. Let consider $n = 10 + 2$, where +2 is for starting and ending mel scale value.

Table 3.1 List of Mel scale value with n = 10

| $m(i)$ | |
|---|---|
| 1 | 401.25 |
| 2 | 622.5 |
| 3 | 843.75 |
| 4 | 1065 |
| 5 | 1286.25 |
| 6 | 1507.5 |
| 7 | 1728.74 |
| 8 | 1950 |
| 9 | 2171.24 |
| 10 | 2392.5 |
| 11 | 2613.74 |
| 12 | 2834.99 |

Now, convert back to frequency from mel scale $m(i)$ using equation (3.7).

Table 3.2 List of Frequencies converting from Mel Scale value

| $h(i)$ | |
|---|---|
| 1 | 299.9917 |
| 2 | 517.3281 |
| 3 | 781.9 |
| 4 | 1103.973 |
| 5 | 1496.046 |
| 6 | 1973.33 |
| 7 | 2554.318 |
| 8 | 3261.641 |
| 9 | 4122.614 |
| 10 | 5170.805 |
| 11 | 6446.691 |
| 12 | 7999.94 |

We can't use frequency resolution to filter out at exact points calculated above, so we need to create frequency bins from $h(i)$. Assume that FFT size is 512 points, frequency bin can be calculated as

$$\boxed{f(i) = \lfloor FFT_{Size} * h(i)/sampling_{rate} \rfloor} \qquad \text{Eq. (3.8)}$$

Table 3.3 List of frequency bins from the given frequencies

| $f(i)$ | |
|---:|---:|
| 1 | 9 |
| 2 | 16 |
| 3 | 25 |
| 4 | 35 |
| 5 | 47 |
| 6 | 63 |
| 7 | 81 |
| 8 | 104 |
| 9 | 132 |
| 10 | 165 |
| 11 | 206 |
| 12 | 256 |

Maximum bin size we got is 256. Now, apply Mel Filter Bank equation (3.5) by substituting the value of $f(i)$, $m$ and $k$ where $k$ from $0$ $to$ max $(f(i))$.

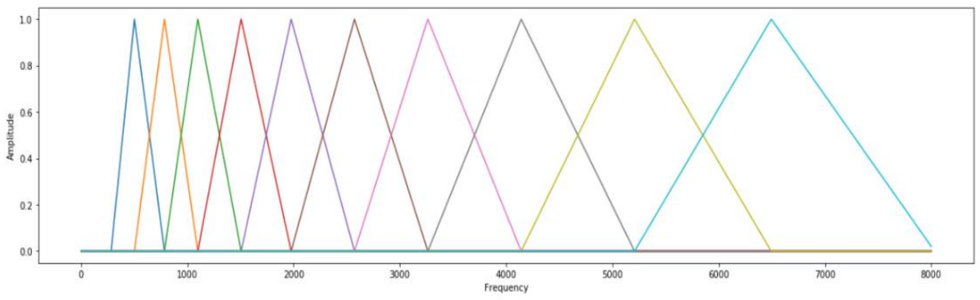


Fig 3.8 Mel Filter Bank For $n = 10$

### 3.1.3 Mel-Frequency Cepstral Coefficient (MFCC)

It is further extension of Mel spectrogram. Mel-Frequency Cepstral Coefficient [20] is another representative way of spectrum of the audio clip, after compressing the frequency. From the mel frequencies output in Mel spectrogram, we take the log of the power and then choose first 13-20 coefficient after Discrete Cosine Transformation (DCT). Increasing number of coefficients represent the incremental changes in estimated energy, thus we got less data. Of course, the most information lost that why we said this technique used for audio compression. We apply DCT instead of taking inverse FFT because it is most like FFT and it is also easy to compute and implement.

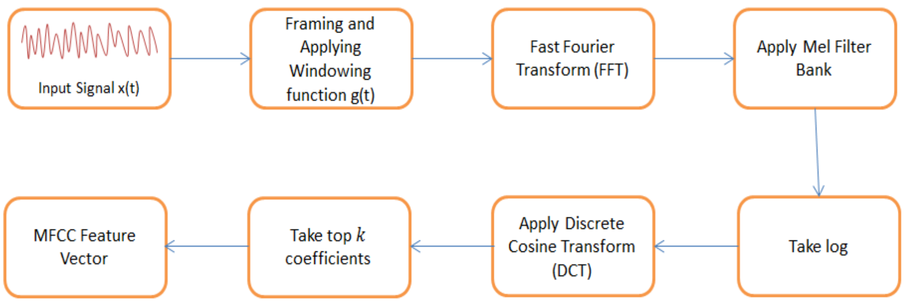

Input Signal x(t)
Framing and Applying Windowing function g(t)
Fast Fourier Transform (FFT)
Apply Mel Filter Bank
Take log
Apply Discrete Cosine Transform (DCT)
Take top $k$ coefficients
MFCC Feature Vector

Fig 3.9 Procedure for MFCC

## 3.2 Deep Learning Used

### 3.2.1 Convolution Neural Network (CNN)

Convolution Neural Network [11, 22] is a class of Neural Network which has been successfully in image classification, recognition and segmentation.

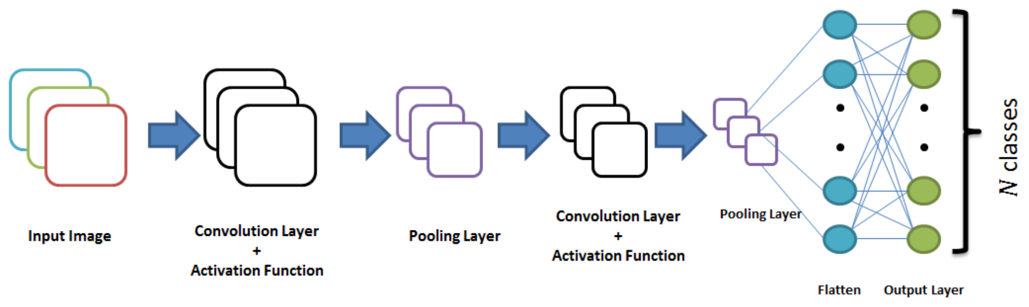

Input Image
Convolution Layer + Activation Function
Pooling Layer
Convolution Layer + Activation Function
Pooling Layer
Flatten
Output Layer
N classes

Fig 3.10 CNN Architecture

In this architecture of Convolution Neural Network, there are four operational stages:

1. Convolutions

   When the input image is fed to this network, its convolute with the convolution kernel.

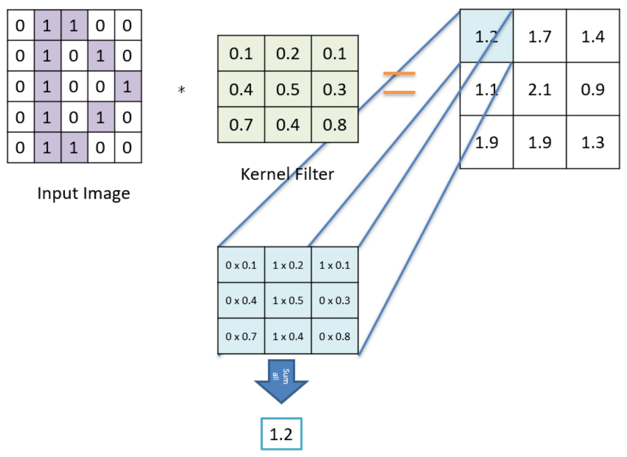


Fig 3.11 Illustration of computing Convolution of Input image (5 x 5) with convolution kernel (3 x 3)

It is noticed from architecture (Convolution Layer) that there are many outputs after performing convolute to every convolution kernel which are not same to other kernels so that for a input image, we can find several features to identify unique from other classes.

2. Rectilinear Unit (ReLU)

   These convolution outputs are then passed through non-linear activation such as ReLU, sigmoid and tanh. The most commonly activation function used as ReLU.

   Mathematically, it is defined as

$$\boxed{R(z) = \max(0, z)} \qquad \text{Eq. (3.9)}$$

   where $z$ is pixel value. The reason is that, after applying convolution filter with input, it is possible that pixel may contain negative pixel value. Generally, image pixel value is range from 0 to $(2^{graylevel} - 1)$. So, by applying these, all negative pixel value will be replaced with 0.

3. Pooling

   In this pooling layer, it basically down sample the image which reduce the dimensionality of the feature input by factor of 2. The most commonly in this layer is MaxPooling. In maxpooling operation, we take the maximum ones with the kernel size from feature map.

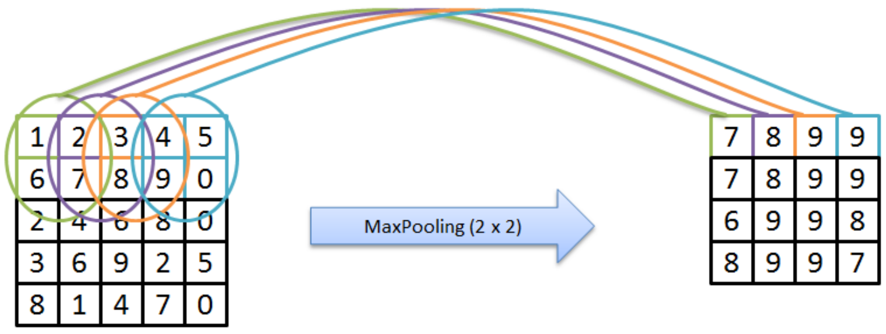


Fig 3.12 Illustration of MaxPooling

4. Fully Connected Layer

   In this layer after performing several convolutions and/or maxpooling layer based on hyper tuning, the feature input are flattened into 1-D array. Now, from these 1-D array input to last layer i.e. output unit. They can now treat as back-propagation neural network. In the output unit, we apply activation function as sigmoid if it is binary classification. And for activation function as SoftMax if it is multiclass classification.

## Updates Weights and Filter Values

Now, once it reached in output units, the loss/error function will generate to see is it identify correctly and by how far off its guesses were. Based on that, we need to update weights and filter weights.

Initially when convolution performed on image, these were calculated as shown figure below.

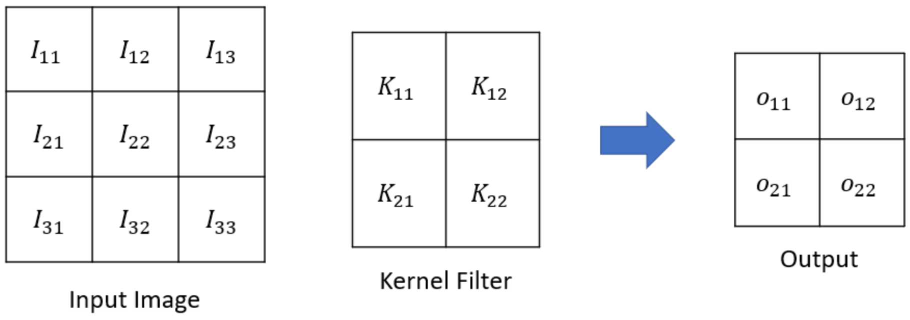


Fig 3.13 Input Image Convolute with Kernel Filter

And above Fig 4.4, this can also written as

$$o_{11} = I_{11}K_{11} + I_{12}K_{12} + I_{21}K_{21} + I_{22}K_{22}$$

$$o_{12} = I_{12}K_{11} + I_{13}K_{12} + I_{22}K_{21} + I_{23}K_{22}$$

$$o_{21} = I_{21}K_{11} + I_{22}K_{12} + I_{31}K_{21} + I_{32}K_{22}$$

$$o_{22} = I_{22}K_{11} + I_{23}K_{12} + I_{32}K_{21} + I_{33}K_{22}$$

When error propagated in backward direction of network, the weight is adjusted by

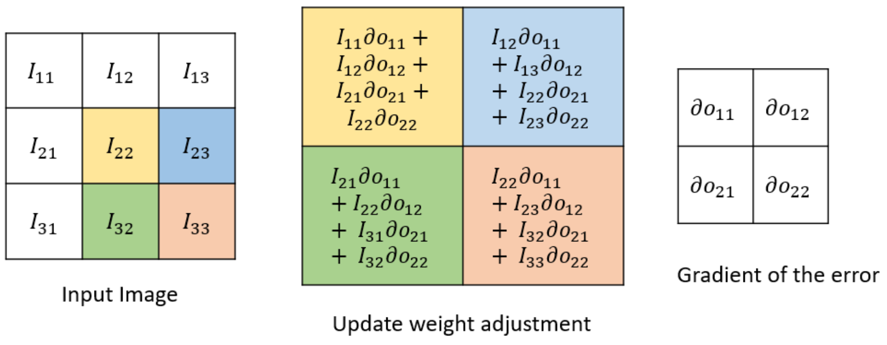

$I_{11}$
$I_{12}$
$I_{13}$
$I_{21}$
$I_{22}$
$I_{23}$
$I_{31}$
$I_{32}$
$I_{33}$
Input Image
$I_{11}\partial o_{11} + I_{12}\partial o_{12} + I_{21}\partial o_{21} + I_{22}\partial o_{22}$
$I_{12}\partial o_{11} + I_{13}\partial o_{12} + I_{22}\partial o_{21} + I_{23}\partial o_{22}$
$I_{21}\partial o_{11} + I_{22}\partial o_{12} + I_{31}\partial o_{21} + I_{32}\partial o_{22}$
$I_{22}\partial o_{11} + I_{23}\partial o_{12} + I_{32}\partial o_{21} + I_{33}\partial o_{22}$
Update weight adjustment
$\partial o_{11}$
$\partial o_{12}$
$\partial o_{21}$
$\partial o_{22}$
Gradient of the error

Fig 3.14 Weight Update Adjustment Procedure

And from the above Fig 4.5, this can also written as

$$\partial K_{11} = I_{11}\partial o_{11} + I_{12}\partial o_{12} + I_{21}\partial o_{21} + I_{22}\partial o_{22}$$

$$\partial K_{12} = I_{12}\partial o_{11} + I_{13}\partial o_{12} + I_{22}\partial o_{21} + I_{23}\partial o_{22}$$

$$\partial K_{21} = I_{21}\partial o_{11} + I_{22}\partial o_{12} + I_{31}\partial o_{21} + I_{32}\partial o_{22}$$

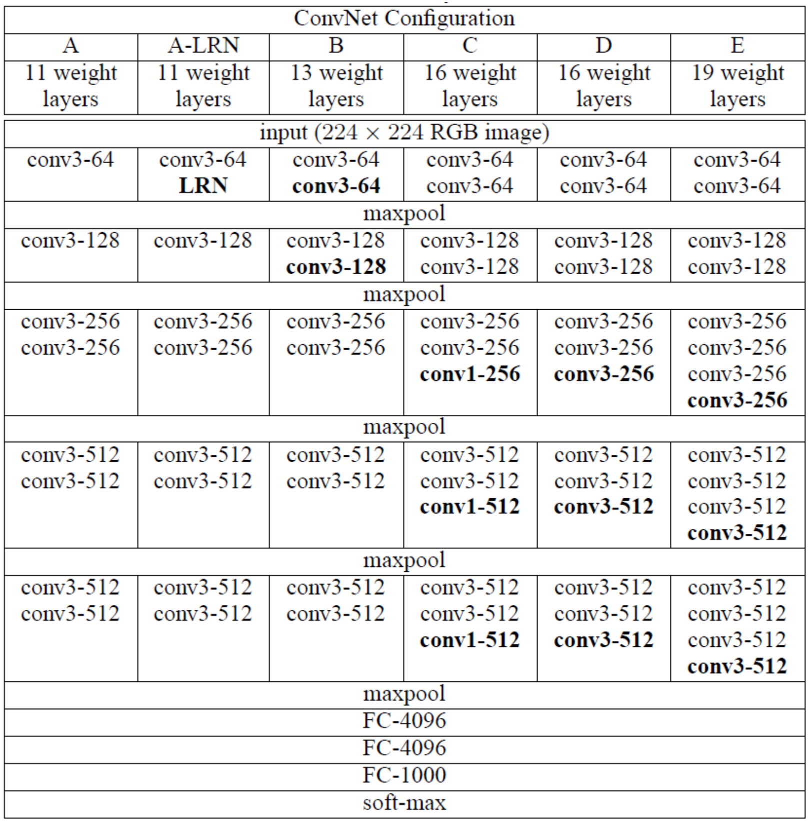
$$\partial K_{22} = I_{22}\partial o_{11} + I_{23}\partial o_{12} + I_{32}\partial o_{21} + I_{33}\partial o_{22}$$

where I represent input image, K represent kernel filter and o represent output of image after convoluting with kernel filter and $\partial$o represents gradient of the error.

So, we got the delta weights (gradient of the error), now we can update by summing the delta weights with the old weights to get the new weight update.

### 3.2.2 VGG Model

This model usually in the field of deep convolution neural network for object recognition which was developed and trained by Oxford's renowned Visual Geometry Group (VGG) [9] and achieved better performance on ImageNet Database. It is totally appealing, however not only that it works well, because Oxford team has also stored the structure of pretrained network and weights which can further use for the other dataset that are available for free online to save time.



| ConvNet Configuration | | | | | |
|---|---|---|---|---|---|
| A | A-LRN | B | C | D | E |
| 11 weight layers | 11 weight layers | 13 weight layers | 16 weight layers | 16 weight layers | 19 weight layers |
| input (224 × 224 RGB image) | | | | | |
| conv3-64 | conv3-64<br>**LRN** | conv3-64<br>**conv3-64** | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 |
| maxpool | | | | | |
| conv3-128 | conv3-128 | conv3-128<br>**conv3-128** | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 |
| maxpool | | | | | |
| conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256<br>**conv1-256** | conv3-256<br>conv3-256<br>**conv3-256** | conv3-256<br>conv3-256<br>conv3-256<br>**conv3-256** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| soft-max | | | | | |

Fig 3.15 Configuration of ConvNet [12]

We've used VGG16 i.e. ConvNet Configuration of 16 weight layers (D) and added final layer as corresponding number of classes present in the dataset. So, instead of using pretrained weight network, we configure the network manually with ConvNet Configuration D with initializing random weights (not trained ones).

### 3.2.3 Recurrent Neural Network (RNN)

Humans don’t start their thinking from scratch every second. There is always retaining the information in our memory. For example, if you read the newspaper, you always connect the word with the previous word i.e. your mind keeping the track of sentence and predict what next comes. You don’t throw everything away and start thinking from scratch again.

We have only considered the neural networks where input data always move feedforward the network. For instance, our task is to predict next word in a sentence. If we applied them in simplest feedforward neural network, the input node received the input data and pass to the next layer which is hidden layer and apply activation function to these data and pass to next layer and so on. And once the data reached to output layer, it will generate probability of each class and chose the maximum probability ones that consider to be class where belongs. Now, next time the data receive to input layer, then pass to hidden layer and applied activation and pass to output layer. It seems that they have different weight assigned to different nodes at every hidden layer.

Since, the weight and bias of hidden layer are different from each other that why they are independent and can’t be combined to work together. In order to combine hidden layers, we need to consider as same weight and bias for these hidden layers.

To overcome this issue, RNN [13] plays roles which retain some of the memory store for future references. RNN have loops.

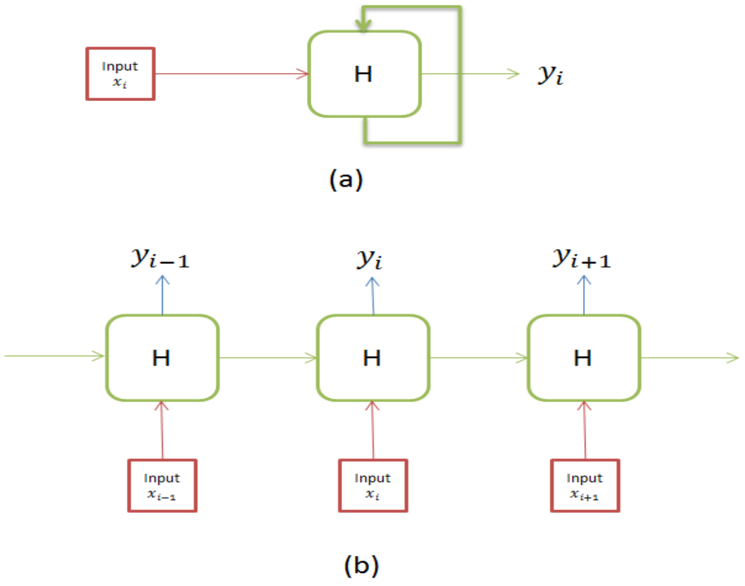


Fig 3.16 (a) RNN have loops (b) RNN when unrolled

A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor.

So, for every time steps, when we fed the input unit to the hidden layer, the weights of this architecture or recurrent neuron (H) are same. The idea is that by using this architecture, we need two parameters: previous input state and current input state. With that two parameters, we want to find the relationship between them by feeding these two parameters together to this RNN network. So, in order to store the relationship between these two parameters to proceed with the next time step of next input state, the recurrence formula is applied.

For example, let say input as "hello". From the RNN architecture in Fig 3.16, the Green RNN block (H), has something called a recurrence formula for the current input state and its previous state. So, for the first-time input, the letter "h" fed to the network, at that time there is nothing preceding it as it is coming as first time occur in the network. Now, when we take the next input letter "e", at that time step when the letter "e" is fed to the network, a recurrence formula is performed inside Green RNN block (H) which take two parameters that is current input state ("e") and previous state ("h").

These transaction of state from one input to next input state is known as time step. So, if the current input is “e” at the time step $t$ then for the letter “h” had to be appeared in at timestep $t-1$. When a recurrent formula is applied for current and previous input state, we generate a new state which further pass to the next time step. General formula for the current input state is written as

$$\boxed{h_t = f(h_{t-1}, X_t)} \qquad \text{Eq. (3.10)}$$

where $h_t$ is new state, $h_{t-1}$ is previous state and $X_t$ is current input. The weight matrices are like filters value which help us to determine which one should keep the value and by how much for both the present input state and the previous input state. The gradient which they generate can be used to adjust their weights using Backpropagation through Time (BPTT). The weight summation of input and hidden state is squashed by the activation function $f$ (sigmoid function or tanh function).

## Vanishing Problem

Vanishing gradient [14] problem occur by applying choice of activation function which have a squashing property. Most commonly used the activation functions are sigmoid function and tanh function as these both functions have the squashing property. “Squashing” means converting the larger scale value into the smaller scale at range. For example, when activation function- sigmoid function performed on input units which have larger scale value, it will squash it and make it into the range of (0,1) for the first layer. Then the output of first layer will fed as the input to the next layer and activation function is same as previous, it will further reduce the value and so on. At the time of final output layer, it will tend to smaller changes as a gradient which is even negligence value as it doesn’t change much the overall network. So, adding more and more layer will tends to gradient as a negligence value and it wouldn’t help the network to adjust the weights.

We can overcome this kind of problem by using activation functions which don't have the squashing property. A popular choice is Rectified Linear Unit activation function (ReLU).

Limitation of RNNs are

1. Short term memory which discards the information from early times steps when moving to later ones, which result in loss of importance information.
2. Vanishing Gradient which updates the value of weight used in neural network. If a gradient is too small and have several hidden layers present in the network, then it won’t change the weight of the early time steps i.e. it will not contribute too much learning.
3. Exploding gradient which occurs when the network assigns unreasonably high importance of weights.

### 3.2.3.1 Long Term Short Memory (LSTM) Model

Long-Short-Term Memory (LSTM) [15, 19] models are a type of Recurrent Neural Networks (RNNs) which can learn and remember over long sequences of input data using “gates” which regulate the information flow of the network. LSTM address the following limitation of RNN.

LSTMs makes use of gates to decide if it should keep or forget information. An LSTM cell consists of a cell state and input, forget and output gates which make use of several activation functions.

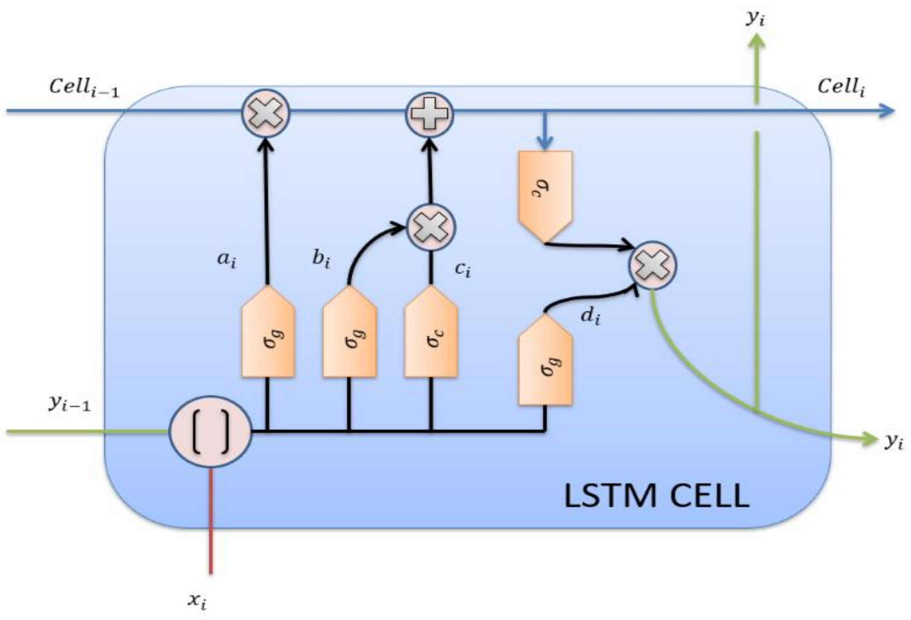


Fig 3.17 Architecture of LSTM

From the architecture in Fig 3.17, we can define the equation of LSTM unit with gates are:

$$a_i = \sigma_g(W_a[x_i + y_{i-1}] + B_a) \quad \text{Eq. (3.11)}$$

$$b_i = \sigma_g(W_b[x_i + y_{i-1}] + B_b) \quad \text{Eq. (3.12)}$$

$$d_i = \sigma_g(W_d[x_i + y_{i-1}] + B_d) \quad \text{Eq. (3.13)}$$

$$Cell_i = a_i \times Cell_{i-1} + b_i \times \sigma_c(W_c[x_i + y_{i-1}] + B_c) \quad \text{Eq. (3.14)}$$

$$y_i = \sigma_c(Cell_i) \times d_t \quad \text{Eq. (3.15)}$$

where $x_i$: input vector to a LSTM unit

$a_i$: forget gate's vector

$b_i$: input or update's vector

$d_i$: output gate's vector

$C_i$: cell state vector

$y_i$: hidden state vector or output vector of LSTM

$\sigma_g$: Sigmoid Function

$\sigma_c$: Hyperbolic tangent function,

W and B are the weights and bias of their corresponding gates (written in subscript)

### 3.2.3.2 Gated Recurrent Units (GRU)

In simple words, the GRU [16] unit does not have to use a memory unit to control the flow of information like the LSTM unit. It can directly make use of the all hidden states without any control. GRUs have fewer parameters and thus may train a bit faster or need less data to generalize. But, with large data, the LSTMs with higher expressiveness may lead to better results.

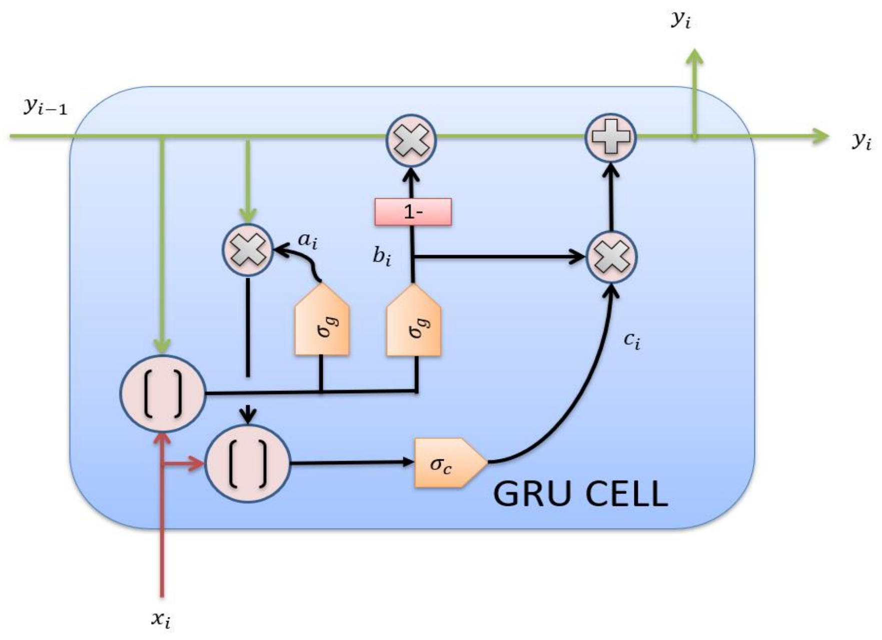


Fig 3.18 Architecture of GRU

They are almost like LSTMs except that they have two gates: reset gate and update gate. Reset gate determines how to combine new input to previous memory and update gate determines how much of the previous state to keep. Update gate in GRU is what input gate and forget gate were in LSTM. We don't have the second nonlinearity in GRU before calculating the output, neither they have the output gate.

From the architecture in Fig 3.18, we can define equation for GRU unit with gates (in the similar fashion as we defined in LSTM unit) which are:

$$b_i = \sigma_g(W_b[x_i + y_{i-1}] + B_b) \quad \text{Eq. (3.16)}$$

$$a_i = \sigma_g(W_a[x_i + y_{i-1}] + B_a) \quad \text{Eq. (3.17)}$$

$$c_i = \sigma_c(W_c[x_i + (a_i \times y_{i-1})] + B_c) \quad \text{Eq. (3.18)}$$

$$y_i = (1 - b_i) \times y_{i-1} + b_i \times c_i \quad \text{Eq. (3.19)}$$

where $x_i$: input vector to GRU

$y_i$: output vector of GRU

$b_i$: update gate's vector

$a_i$: reset gate's vector

$\sigma_g$: Sigmoid Function

$\sigma_c$: Hyperbolic tangent function

W and B are the weights and bias of their corresponding gates (written in subscript)

### 3.2.3.3 Bi-directional RNN

As a name suggested “Bi-Directional” [17] means two directions. Two directions are forward direction and backward direction.

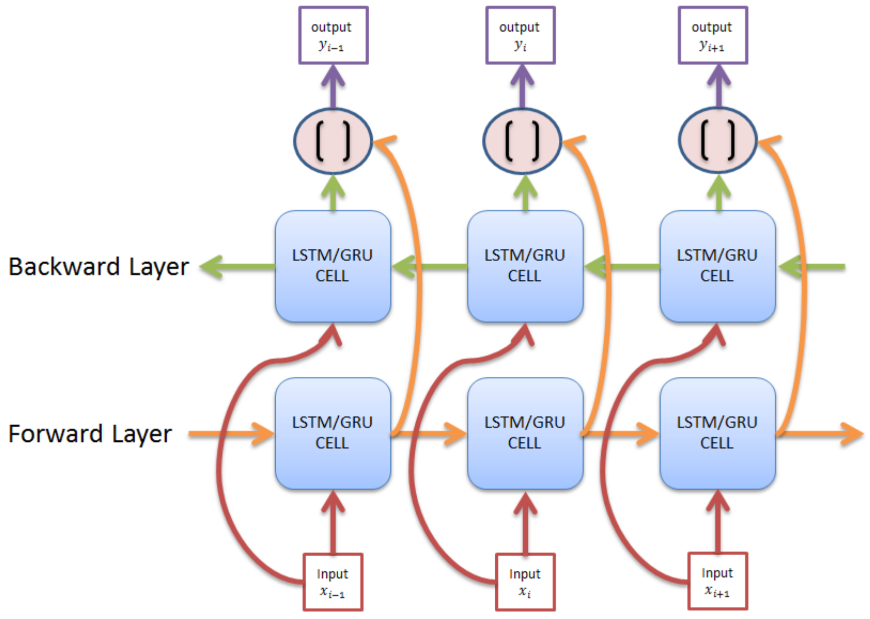


Fig 3.19 Architecture of Bi-Directional RNN

From the figure 3.19, the forward layer will proceed as RNN works normally and each of the timesteps, their corresponding output will be generated (let say, $y_{f(i)}$). At the timestep reach at the last, the backward layer will proceed in the backward direction and at each timestep, their corresponding output will be generated (let say, $y_{b(i)}$). Then these two layers corresponding outputs are concatenated to each other.

$$\boxed{y_i = [y_{f(i)}, y_{b(i)}]} \qquad \text{Eq. (3.20)}$$

We can apply Bi-Directional RNN to create Bi-Directional LSTM or Bi-Directional GRU. Just replace the RNN unit with the LSTM unit or GRU unit. Also, since we are doing for both directions, the time consuming will be double as RNN time consume because at the backward layer, there is also need of timestep in backward direction also.

# CHAPTER 4

# EXPERIMENTAL RESULT

## 4.1 Proposed Architecture

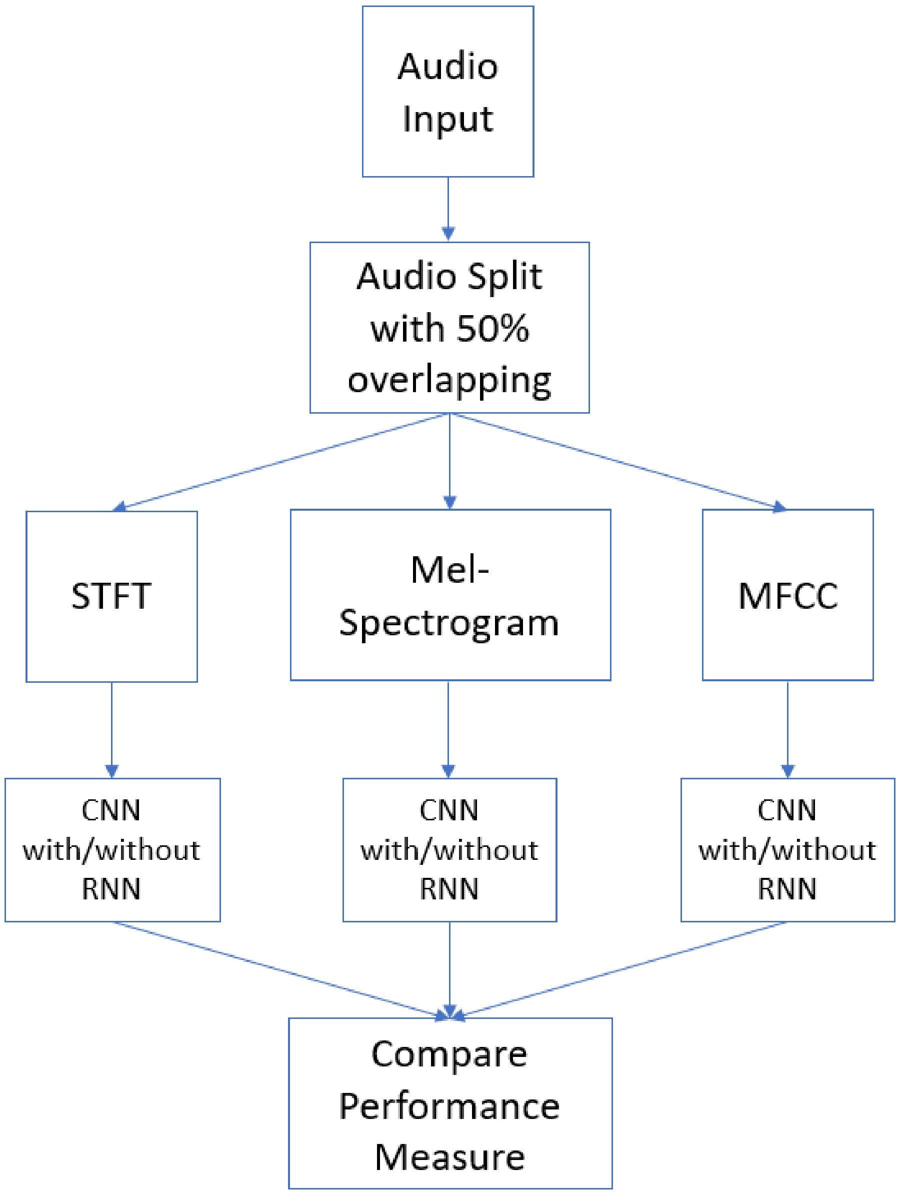


Fig 4.1 Proposed Architecture

## 4.2 Dataset

We used GTZAN [18] dataset. It contains 10 classes and each class contains 100 audios. Each audio contains 30 seconds of track. The tracks are in 22050 Hz with mono 16-bit audio file in .wav format. Below table are the class list name of the music.

Table 4.1 Music Genre Classes

| | |
|---|---|
| **1** | Blues |
| **2** | Classical |
| **3** | Country |
| **4** | Disco |
| **5** | Hip hop |
| **6** | Jazz |
| **7** | Metal |
| **8** | Pop |
| **9** | Reggae |
| **10** | Rock |

## 4.3 Data Pre processing

We read the audio with the sampling rate of 22050 Hz. After that, we split the audio of 30 seconds durations into 3 seconds durations of audio clips. The problem is that when split the audio, the computer will consider that each clip are not relate to each other and are independent. So, in order to avoid this, we have taken 50% of previous duration of data and next 50% of next duration of data. This will help to understand the computer that the first 50% duration of data are from the previous audio clip and from that data, it continues to next 50% duration of data.

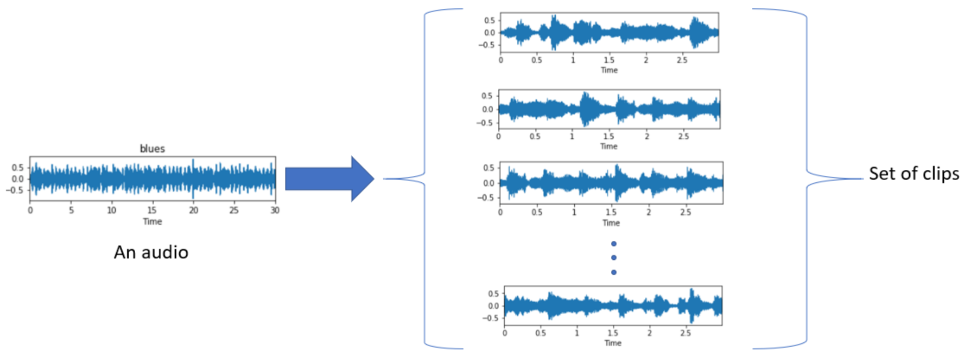


Fig 4.2 Data Pre-processing Step of an audio

## 4.4 Feature Extraction

Each clip has performed different feature extraction to see the analysis which one been performed better. First, we applied Short Term Fourier Transform (STFT) where FFT window size (frame size) is 1024 and hop length (frame increment) is 512 and windowing function is Hann function. Each clip gets (513, 129) dimensions that is 513 frequency bins and 129 frames in time.

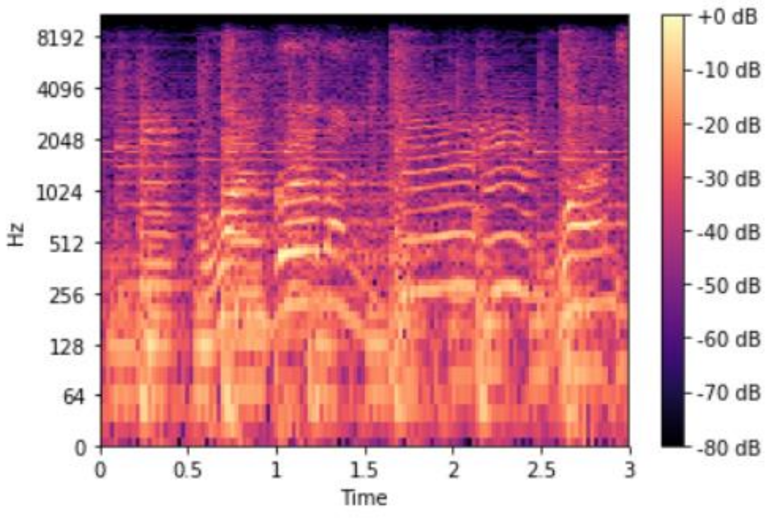


Fig 4.3 STFT for a blue class audio with 3 seconds duration

Second, we applied Mel-Spectrogram o each clip where FFT window size is 1024 and hop length is 512. Each clip gets (128,129) dimensions.

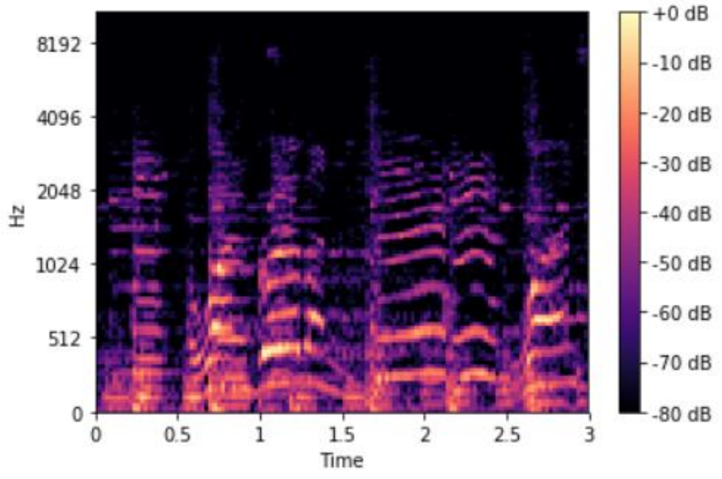


Fig 4.4 Mel-Spectrogram for a blue class audio with 3 seconds duration

Lastly, we applied further extended to Mel-Spectrogram to MFCC. I observed with different top $k$ coefficient values after applying DCT, I got 13 to be the best optimal choice.

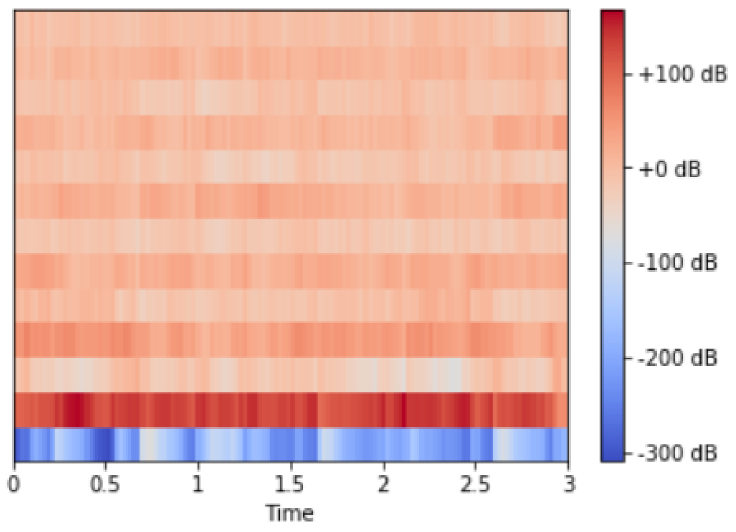


Fig 4.5 MFCC for a blue class audio with 3 seconds duration

## 4.5 Learning Algorithm

Data are divided into three sections: training data, validation data and test data. And these data are split into 80% of training data, 10% of validation data and 10% of testing data. These training data are then fed to Convolution Neural Network (CNN) before reshaping the data. Once an epoch is done, we check the performance of validation data to see how much can generalize the unknown data after learnt from training data. Once it completed all trained network, we evaluate for the test data after seeking training and validation data performance.

VGG16 has been trained on Mel-Spectrogram feature extraction data only because of the limited resource present in our system. This network has outperformed than CNN in these features. You will see the result in tabular form clearly in the next section part.

In configuration of proposed CNN network, we create five Convolution Block layer. Each block layer contain Convolution Kernel with different kernel size followed by MaxPooling Layer followed by Dropout with droprate 25%. Then we flatten them into 1D array and output layer.

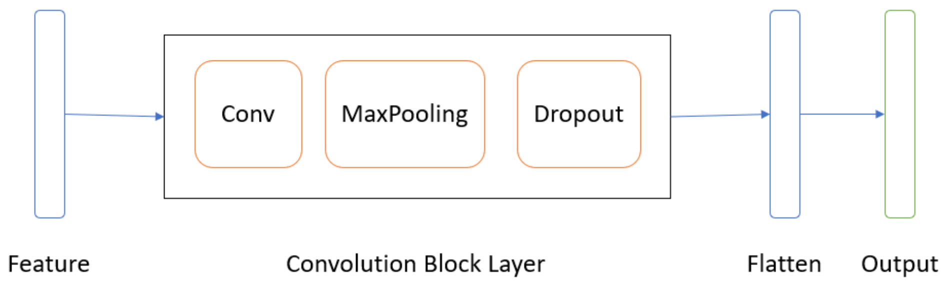


Fig 4.6 Proposed CNN Architecture

For the configuration of proposed CNN with LSTM (or GRU) network, we create five Convolution Block layer. After that we followed by three LSTM/GRU layer with two 128 lstm (or gru) unit and one 64 lstm (or gru) unit then we flatten them into 1D array and apply one dense layer and finally, that last one as, output layer.

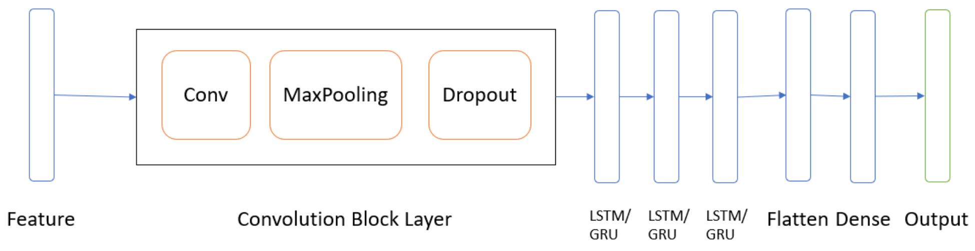
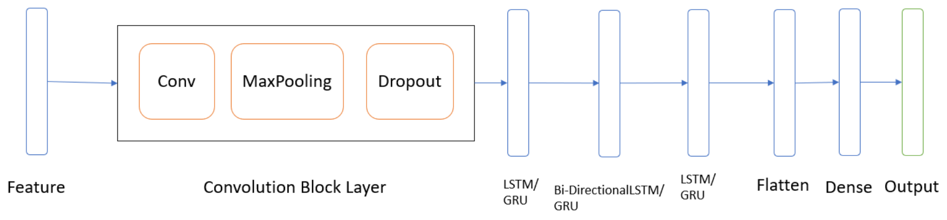


Fig 4.7 Proposed CNN with LSTM (or GRU) Architecture

For configuration of proposed CNN with Bi-RNN architecture network, we create four Convolution Block layer. After that we followed by one LSTM/GRU layer with one 128 lstm (or gru) unit followed by bi-Directional LSTM (or GRU) layer with 128 units followed by one 64 lstm (or gru) units then we flatten them into 1D array and apply one dense layer with 32 number of nodes and at last, output layer which contain 10 number of nodes as same as number of classes.



.

Fig 4.8 Proposed CNN with Bi-RNN Architecture

We have done several tuning hyperparameters to obtain best performance as possible. For the best optimal parameter to achieve this work, we take 5 number of CNN layers and we added dropout in each layer about 25% to avoid overfitting and apply kernel regularizer (L1) which is LASSO regularizer and lambda = 0.01 to adjust the weight to reduce cost function. We have shown the summary of architecture in given above for the best ones which the overall accuracy of 95.5% with STFT feature extractions data.

## 4.6 Result with different feature extraction and learning algorithm

Table 4.2 Performance Evaluation & Result

| ***Feature Extraction Techniques:*** | ***Training Method*** | ***Epoch*** | ***Train*** | ***Valid*** | ***Test*** |
|---|---|---|---|---|---|
| *Mel-Spectrogram* | CNN | 120 | 0.8788 | 0.9023 | 0.901 |
| | CNN + LSTM | 120 | 0.9214 | 0.914 | 0.903 |
| | CNN + GRU | 120 | 0.9186 | 0.9187 | 0.904 |
| | CNN + B-LSTM | 120 | 0.9273 | 0.9146 | 0.889 |
| | CNN + B-GRU | 120 | 0.9254 | 0.9041 | 0.895 |
| | | | | | |
| | VGG16 | 25 | 0.9831 | 0.9351 | 0.919 |
| | VGG16 + LSTM | 20 | 0.9335 | 0.8719 | 0.877 |
| | VGG16 + GRU | 20 | 0.9551 | 0.8877 | 0.884 |
| | VGG16 + BGRU | 20 | 0.9457 | 0.886 | 0.892 |
| | VGG16 + BLSTM | 15 | 0.8528 | 0.8339 | 0.803 |
| | | | | | |
| *MFCC (13)* | CNN | 200 | 0.8949 | 0.8491 | 0.862 |
| | CNN + LSTM | 200 | 0.8559 | 0.8474 | 0.853 |
| | CNN + GRU | 200 | 0.8331 | 0.8281 | 0.831 |
| | CNN + B-LSTM | 200 | 0.8616 | 0.8509 | 0.852 |
| | CNN + B-GRU | 200 | 0.8601 | 0.8 | 0.791 |
| | | | | | |
| *STFT* | **CNN** | **70** | **0.9589** | **0.9485** | **0.955** |
| | CNN + GRU | 70 | 0.9227 | 0.9193 | 0.928 |
| | CNN + LSTM | 70 | 0.9283 | 0.9117 | 0.922 |
| | CNN + B-GRU | 50 | 0.9041 | 0.9041 | 0.902 |
| | CNN + B-LSTM | 120 | 0.7723 | 0.773 | 0.773 |

## 4.7 Graph between Training Data and Validation Data with respect to Accuracy and Loss

In this section, we are showing the graph for the best ones of each feature extractions data.

With Mel-Spectogram Feature extractions data, we performed two networks (CNN and VGG16). Graph plots are shown below Fig 4.9 & Fig 4.10.

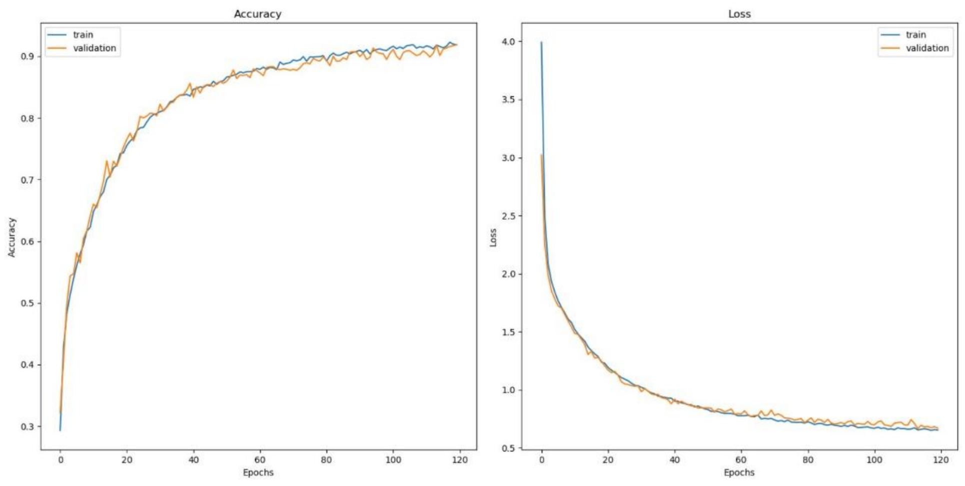


Fig 4.9 Graph plot for CNN with GRU trained network with Mel-Spectrogram input features

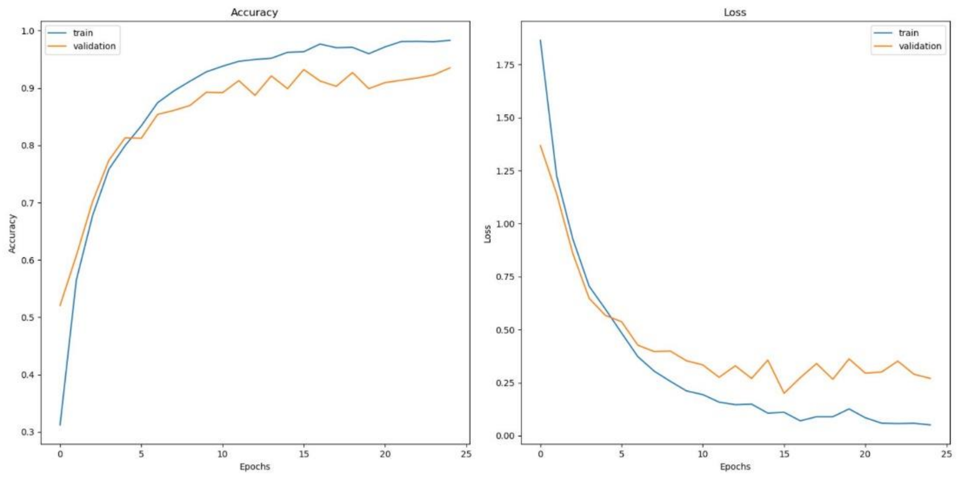


Fig 4.10 Graph plot for VGG16 trained network with Mel-Spectrogram input features

For MFCC feature extraction data, graph plot is shown below (Fig 4.11).

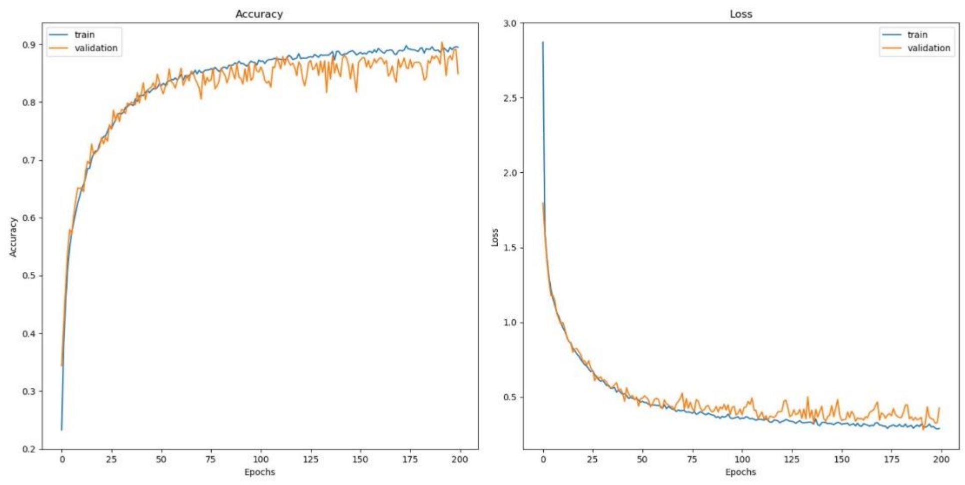


Fig 4.11 Graph Plot for CNN trained network with MFCC input features

For STFT feature extraction data, graph plot is shown below (Fig 4.12).

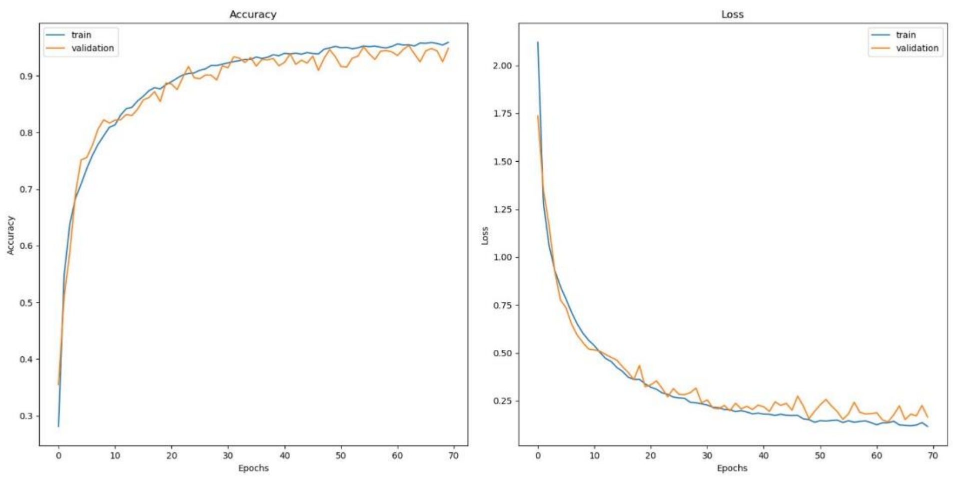
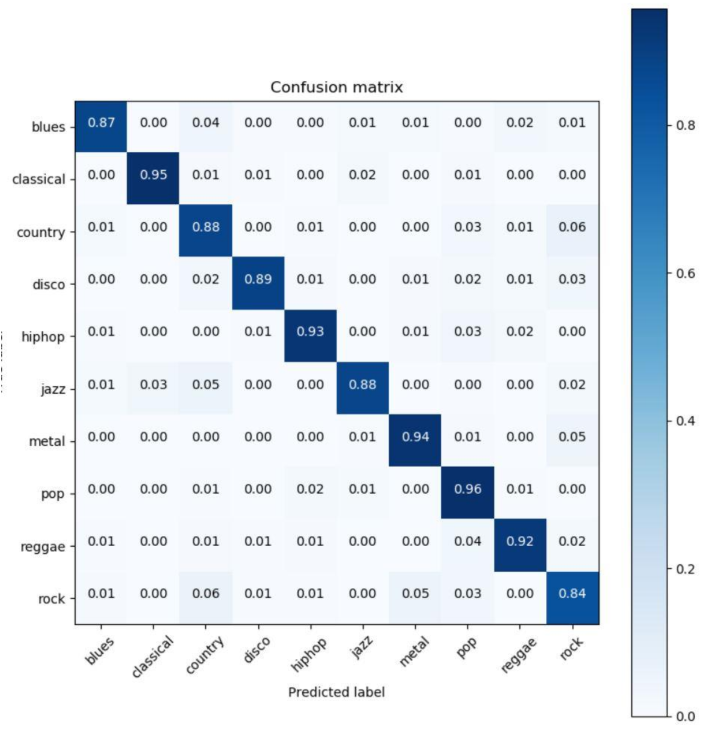


Fig 4.12 Graph Plot for CNN trained network with STFT input features

## 4.8 Confusion Matrix For Test Data

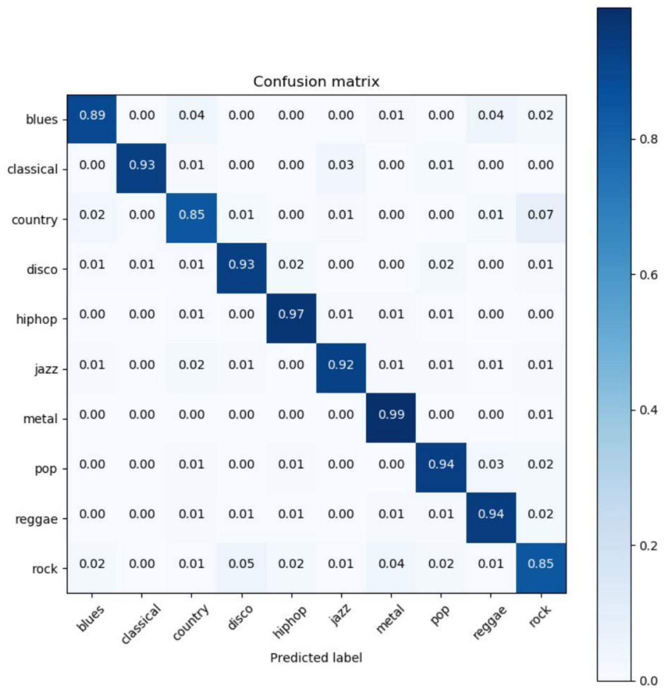
In this section, we are going to show confusion matrix for the best ones only like previous section.

For Test Data with Mel-Spectrogram Feature Extraction, confusion matrixes are shown in Fig 4.13 and Fig 4.14 since we have performed with this feature data into two networks (CNN & VGG16).



| | blues | classical | country | disco | hiphop | jazz | metal | pop | reggae | rock |
|---|---|---|---|---|---|---|---|---|---|---|
| blues | 0.87 | 0.00 | 0.04 | 0.00 | 0.00 | 0.01 | 0.01 | 0.00 | 0.02 | 0.01 |
| classical | 0.00 | 0.95 | 0.01 | 0.01 | 0.00 | 0.02 | 0.00 | 0.01 | 0.00 | 0.00 |
| country | 0.01 | 0.00 | 0.88 | 0.00 | 0.01 | 0.00 | 0.00 | 0.03 | 0.01 | 0.06 |
| disco | 0.00 | 0.00 | 0.02 | 0.89 | 0.01 | 0.00 | 0.01 | 0.02 | 0.01 | 0.03 |
| hiphop | 0.01 | 0.00 | 0.00 | 0.01 | 0.93 | 0.00 | 0.01 | 0.03 | 0.02 | 0.00 |
| jazz | 0.01 | 0.03 | 0.05 | 0.00 | 0.00 | 0.88 | 0.00 | 0.00 | 0.00 | 0.02 |
| metal | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 | 0.94 | 0.01 | 0.00 | 0.05 |
| pop | 0.00 | 0.00 | 0.01 | 0.00 | 0.02 | 0.01 | 0.00 | 0.96 | 0.01 | 0.00 |
| reggae | 0.01 | 0.00 | 0.01 | 0.01 | 0.01 | 0.00 | 0.00 | 0.04 | 0.92 | 0.02 |
| rock | 0.01 | 0.00 | 0.06 | 0.01 | 0.01 | 0.00 | 0.05 | 0.03 | 0.00 | 0.84 |

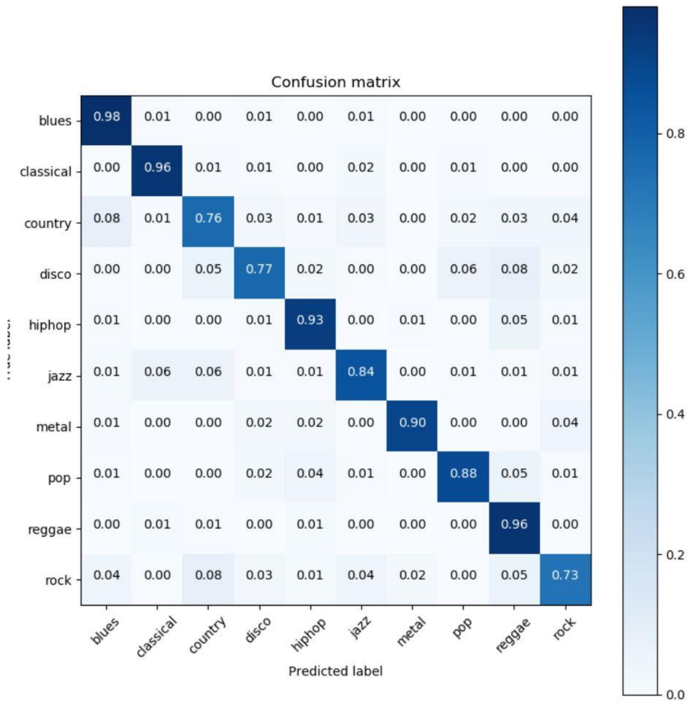
Fig 4.13 Confusion Matrix (CNN + GRU) with Mel-Spectrogram input features



| | blues | classical | country | disco | hiphop | jazz | metal | pop | reggae | rock |
|---|---|---|---|---|---|---|---|---|---|---|
| blues | 0.89 | 0.00 | 0.04 | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 | 0.04 | 0.02 |
| classical | 0.00 | 0.93 | 0.01 | 0.00 | 0.00 | 0.03 | 0.00 | 0.01 | 0.00 | 0.00 |
| country | 0.02 | 0.00 | 0.85 | 0.01 | 0.00 | 0.01 | 0.00 | 0.00 | 0.01 | 0.07 |
| disco | 0.01 | 0.01 | 0.01 | 0.93 | 0.02 | 0.00 | 0.00 | 0.02 | 0.00 | 0.01 |
| hiphop | 0.00 | 0.00 | 0.01 | 0.00 | 0.97 | 0.01 | 0.01 | 0.01 | 0.00 | 0.00 |
| jazz | 0.01 | 0.00 | 0.02 | 0.01 | 0.00 | 0.92 | 0.01 | 0.01 | 0.01 | 0.01 |
| metal | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.99 | 0.00 | 0.00 | 0.01 |
| pop | 0.00 | 0.00 | 0.01 | 0.00 | 0.01 | 0.00 | 0.00 | 0.94 | 0.03 | 0.02 |
| reggae | 0.00 | 0.00 | 0.01 | 0.01 | 0.01 | 0.00 | 0.01 | 0.01 | 0.94 | 0.02 |
| rock | 0.02 | 0.00 | 0.01 | 0.05 | 0.02 | 0.01 | 0.04 | 0.02 | 0.01 | 0.85 |

Fig 4.14 Confusion Matrix (VGG16) with Mel-Spectrogram input features

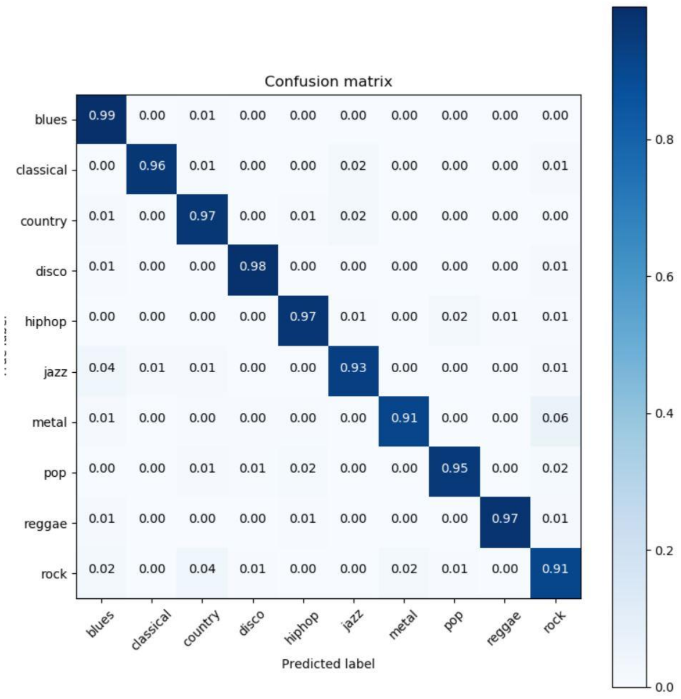
For Test Data with MFCC feature extraction, confusion matrix is shown below Fig 4.15.



| | blues | classical | country | disco | hiphop | jazz | metal | pop | reggae | rock |
|---|---|---|---|---|---|---|---|---|---|---|
| blues | 0.98 | 0.01 | 0.00 | 0.01 | 0.00 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 |
| classical | 0.00 | 0.96 | 0.01 | 0.01 | 0.00 | 0.02 | 0.00 | 0.01 | 0.00 | 0.00 |
| country | 0.08 | 0.01 | 0.76 | 0.03 | 0.01 | 0.03 | 0.00 | 0.02 | 0.03 | 0.04 |
| disco | 0.00 | 0.00 | 0.05 | 0.77 | 0.02 | 0.00 | 0.00 | 0.06 | 0.08 | 0.02 |
| hiphop | 0.01 | 0.00 | 0.00 | 0.01 | 0.93 | 0.00 | 0.01 | 0.00 | 0.05 | 0.01 |
| jazz | 0.01 | 0.06 | 0.06 | 0.01 | 0.01 | 0.84 | 0.00 | 0.01 | 0.01 | 0.01 |
| metal | 0.01 | 0.00 | 0.00 | 0.02 | 0.02 | 0.00 | 0.90 | 0.00 | 0.00 | 0.04 |
| pop | 0.01 | 0.00 | 0.00 | 0.02 | 0.04 | 0.01 | 0.00 | 0.88 | 0.05 | 0.01 |
| reggae | 0.00 | 0.01 | 0.01 | 0.00 | 0.01 | 0.00 | 0.00 | 0.00 | 0.96 | 0.00 |
| rock | 0.04 | 0.00 | 0.08 | 0.03 | 0.01 | 0.04 | 0.02 | 0.00 | 0.05 | 0.73 |

Fig 4.15 Confusion Matrix (CNN) with MFCC input features

For Test Data with Short Term Fourier Transform (STFT), confusion matrix is shown below Fig 4.16.



| | blues | classical | country | disco | hiphop | jazz | metal | pop | reggae | rock |
|---|---|---|---|---|---|---|---|---|---|---|
| blues | 0.99 | 0.00 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| classical | 0.00 | 0.96 | 0.01 | 0.00 | 0.00 | 0.02 | 0.00 | 0.00 | 0.00 | 0.01 |
| country | 0.01 | 0.00 | 0.97 | 0.00 | 0.01 | 0.02 | 0.00 | 0.00 | 0.00 | 0.00 |
| disco | 0.01 | 0.00 | 0.00 | 0.98 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 |
| hiphop | 0.00 | 0.00 | 0.00 | 0.00 | 0.97 | 0.01 | 0.00 | 0.02 | 0.01 | 0.01 |
| jazz | 0.04 | 0.01 | 0.01 | 0.00 | 0.00 | 0.93 | 0.00 | 0.00 | 0.00 | 0.01 |
| metal | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.91 | 0.00 | 0.00 | 0.06 |
| pop | 0.00 | 0.00 | 0.01 | 0.01 | 0.02 | 0.00 | 0.00 | 0.95 | 0.00 | 0.02 |
| reggae | 0.01 | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 | 0.00 | 0.00 | 0.97 | 0.01 |
| rock | 0.02 | 0.00 | 0.04 | 0.01 | 0.00 | 0.00 | 0.02 | 0.01 | 0.00 | 0.91 |

Fig 4.16 Confusion Matrix (CNN) with STFT input features

# CONCLUSION AND FUTURE WORK

In this work, we have performed with new terminology called audio split. Due to rate of change of every signal was changing every interval of time. To overcome that, we split an audio of 30 seconds into clips with 3 seconds duration and 50% overlapping with previous signal to remain dependent. Then we performed top three feature extraction techniques which has been performed for several applications related music and speech are Short term Fourier Transform (STFT), Mel-Spectrogram and Mel-Frequency Cepstral Coefficients (MFCC). And train the network with combining Convolution Neural Network with Recurrent Neural Network in order to achieved better performance measure. We achieved 96% training accuracy, 95% valid accuracy and 95.5 % percent test accuracy using short term fourier transform (STFT) with Convolution Neural Network (CNN).

We can apply on other feature extraction techniques which has not been applied in this work yet because the training time was taking too long for training and had to tuning the hyperparameter of the network. Furthermore, we can try to implement this technique using an application on real time based.

## REFERENCES

[1] Caifeng Liu, Lin Feng, Guochao Liu, Huibing Wang, Shenglan Liu,” Bottom-up Broadcast Neural Network For Music Genre Classification”, ScienceDirect,Elsevier,Pattern Recognition, Jan, 2019

[2] Prasenjeet Fulzele, Rajat Singh, Naman Kaushik, Kavita Pandey,” A Hybrid Model For Music Genre Classification”, Proceedings of 2018 Eleventh International Conference on Contemporary Computing (IC3), Aug, 2018

[3] Nilesh M. Patil, Dr. Milind U. Nemade,” Music Genre Classification Using MFCC, K-NN and SVM Classifier”, International Journal Of Computer Engineering In Research Trends, Vol. 4, Issue. 2, Feb, 2017

[4] Ahmed Elbir, Hilmi Bilal Cam, Mehmet Emre lyican, Berkay Ozturk, Nizamettin Aydin,” Music Genre Classification and Recommendation by using Machine Learning Techniques“,Innovations in Intelligent Systems and Applications Conference, Oct, 2018

[5] Pradeep Kumar D, Sowmya B. J., Chetan, and K. G. Srinivasa ,” A Comparative Study of Classifiers for Music Genre Classification based on Feature Extractors”, IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics, Aug, 2016

[6] Anshuman Goel, Mohd. Sheezan, Sarfaraz Msaood, Aadam Saleem,” Genre Classification of Songs Using Neural Network”, International Conference on Computer and Communication Technology (ICCCT), Sept, 2014

[7] Jan Jakubik,” Evaluation of Gated Recurrent Networks in Music Classification Tasks”, Information System Arxhitecture and Technology: Proceedings of 38th International Conference on Information Systems Architecture and Technology, 2017

[8] George Tzanetakis,” Musical Genre Classification of Audio Signals”, IEEE Transactions on Speech and Audio Processing, July, 2002

[9] Lin Feng, Shenlan Liu, Jianing Yao,” Music Genre Classification with Paralleling Recurrent Convolutional Neural Network”, arXiv, Dec, 2017

[10] Jont B. Allen and Lawrence R. Rabiner,” A Unified Approach to Short Time Fourier Analysis and Synthesis”, Proceeding of the IEEE, VOL 65, No.11, Nov, 1977

[11] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, Kaori Togashi,” Convolution Neural Network: an overview and application in radiology”, Springer, Volume 9, Issue 4, Aug, 2018

[12] Keren Simonyan and Andrew Zisserman,” Very Deep Convolution Networks For Large Scale Image Recognition”, ICLR 2015

[13] Alex Graves and Navdeep Jaitly,” Towards End-to-End Speech Recognition with Recurrent Neural Networks”, Proceedings of the 31st International Conference on Machine Learning, vol 32, 2014

[14] Yoshua Bengio, Patrica Simard and Paolo Fransconi,” Learning Dependencies with Gradient Descent is Difficult”, IEEE Transactions On Neural Networks, Vol 5, No 2, Mar, 1994

[15] Sepp Hochreiter, Jurgen Schmidhuber," Long Short-term Memory", Neural Computation, 1997

[16] Yoshua Bengio, Cho et. al," Learning Phase Representation using RNN Encoder-Decoder for Statistical Machine Transalation", arXiv, Sept, 2014

[17] Mike Schuster and Kuldip K. Paliwal," Bidirectional Recurrent Neural Networks", IEEE Transactions on Signal Processing, vol 45, no. 11, 1997

[18] Bob L. Strum," The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use", arXiv, Jun, 2013

[19] Understanding LSTM network," https://colah.github.io/posts/2015-08-Understanding-LSTMs/ "

[20] Mel Frequency Ceptral Coefficient (MFCC) tutorial, " http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/ "

[21] Short term fourier transform, Digital signal Processing," https://www.youtube.com/watch?v=g1_wcbGUcDY "

[22] An Intuitive Explanation of Convolutional Neural Networks," https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/ "

[23] Time Domain vs Frequency domain," http://play.fallows.ca/wp/radio/ham-radio/signal-analysis-morse-decoder/ "

[24] Procedure of evaluating Mel-Spectrogram Architecture," https://in.mathworks.com/help/audio/ref/melspectrogram.html "