URBAN SOUND CLASSIFICATION USING DEEP LEARNING

Professional Practice/Seminar (IT890) Project Report Submitted in

partial fulfilment of the requirements for the degree of

MASTER OF TECHNOLOGY
in

INFORMATION TECHNOLOGY

by

TARUSHI JAT (202204)

DEPARTMENT OF INFORMATION TECHNOLOGY
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL, MANGALORE -575025

May, 2021

D E C L A R A T I O N

I hereby *declare* that the *Professional Practice/Seminar (IT890) Project Work Report* of the M.Tech.(IT) entitled Urban Sound Classification using Deep Learning which is being submitted to the National Institute of Technology Karnataka Surathkal, in partial fulfilment of the requirements for the award of the Degree of Master of Technology in the department of Information Technology, is a ***bonafide report of the work carried out by me.*** The material contained in this project report has not been submitted to any University or Institution for the award of any degree.

[illegible]

Tarushi Jat, 202204

Department of Information Technology

Place : NITK, SURATHKAL

Date : May 18, 2021

C E R T I F I C A T E

This is to *certify* that the Professional Practice/Seminar (IT890) Project Work Report entitled URBAN SOUND CLASSIFICATION USING DEEP LEARNING submitted by Tarushi Jat, (Register Number: 202204) as the record of the work carried out by him/her, is *accepted as the Professional Practice/Seminar (IT890) Project Work Report submission* in partial fulfilment of the requirements for the award of degree of Master of Technology in the Department of Information Technology.

ACKNOWLEDGEMENT

I would like to express my deep gratitude to **Prof. Ananthanarayana V. S.**, our research supervisors, for his patient guidance, enthusiastic encouragement and useful critiques of this research work. I would also like to thank him for his advice and assistance in keeping our progress on schedule.

I would also thank my parents for their support and encouragement throughout my study.

Tarushi Jat

ABSTRACT

The classification of urban sounds has some potential application in the implementation of smart cities since audio carries a large amount of real life physical events in the city, therefore in this project, our aim is to develop a deep learning approach to extract the information from the urban sounds and classify them into into different classes with respect to their frequency spectrum. As an output our model will classify an input audio file into one of the following 10 classes : *Air Conditioner, Car Horn, Children Playing, Dog Bark, Drilling, Engine Idling, Gun Shot, Jackhammer, Siren, and Street Music*. Here convolutional neural networks are used for training and classification. Feature Extraction is the most crucial task for audio analysis and here Mel Frequency Cepstral Coefficient (MFCC) is used as a feature vector for sound samples. Our result shows the accuracy of our model on test data is around 91% and this will greatly improve the classification of urban sounds into different classes.

Keywords: Urban Sound Classification, CNN, Audio, Frequency.

CONTENTS

INTRODUCTION

The potential benefits of robust sound classification are immense, from surveillance to house monitoring, it is necessary to differentiate between sounds like car horn, siren or street music for smart city systems that could soon be important to have. Everyday sounds that are characteristic of the background of everyday living is the focus of this project. By everyday sounds or urban sounds, the intention is thereby any sound that is not speech or music and that is produced from environments common in day to day life. The prime objective of effort in this project is to represent those sounds and to classify them into their respective classes. In the past few years, many attempts have been made to recognize these urban sounds and presently there is an increasing focus on classification of these urban sounds using deep learning techniques. Therefore, in this project also, efforts are made for the classification of urban sounds using convolutional neural networks as the neural network is very widely used in classification problems and it is helpful in training a huge database.

Sound classification is considered as a very challenging task as selection of appropriate features is a very hard task. For the representation of our sound data, the best way is to use the state of the art results to represent the audio files using their power spectrum, and more commonly known as the mel-frequency cepstrum, this transformation provides a heat map like representation of the signal breaking down the signal based on how much of its total power is in each frequency as time progresses, with darker colors (or higher values) indicating more relative power at that frequency. For all the proposed solutions regarding classification of urban sounds, I have thus used the mel frequency cepstral coefficient of the audio data as the input to the deep learning algorithm. Having the defined inputs, the objective is to classify the testing dataset into the appropriate classes, and so the output is simply the class label of the data input. There are mainly two tasks in this project:

- Understanding and visualizing urban sounds
- Extract features from audios
- Use a few deep learning models to classify urban sounds and compare their performance.

The implementation was done with python programming using Collaboratory as base. The average accuracy obtained by using the MFCC feature vectors and CNN model is 91%.

LITERATURE SURVEY

After Deep Neural Networks (DNN) and Convolutional Neural Networks (CNN) became popular in recent years, more and more research tends to exploit the inherent feature learning that comes with these architectures. They input less processed data including raw waveform or spectrum into the architecture and let the model extract the feature. Zilong Huang, Chen Liu, Hongbo Fei, Wei Li, Jinghu Yu, Yi Cao[1] have proposed a Convolutional Neural Network Model to perform the classification for urban sounds with the extraction of Mel Frequency Cepstral Coefficient and Gammatone frequency cepstral coefficients as features from audio files using UrbanSound8K dataset and they achieved an accuracy of 84.83% with it. Jiaxing Ye, Takumi Kobayashi, Masahiro Murakawa[2] have proposed a support vector machine learning model to classify urban sounds using UrbanSound8K dataset with the help of local and global feature aggregation.

Here in this project, I tried to implement an CNN model for our urban sound dataset and tried to achieve higher accuracies.

METHODOLOGY

A. Dataset Details

The dataset used in this project is UrbanSound8K Dataset[3]. In this dataset, there are 8732 urban sound excerpts which are labeled and divided into 10 folders. Each audio file has a name formatted as [fsID]-[classID]- [occurrenceID]-[sliceID].wav, where

- fsID is the Freesound ID of the recording from where this excerpt (slice) is taken.
- classID is a numeric identifier of the sound class.
- occurrenceID is a numeric identifier to distinguish different occurrences of the sound within the original recording.
- sliceID is a numeric identifier to distinguish different slices taken from the same occurrence.

All the 8732 audio files are divided into 10 classes with different numbers of tracks. The 10 classes are *Air Conditioner, Car Horn, Children Playing, Dog Bark, Drilling, Engine Idling, Gun Shot, Jackhammer, Siren, and Street Music*. Each of the classes is represented by a label. Following is the table that shows the number of audio tracks available in each of the classes and its respective label number.

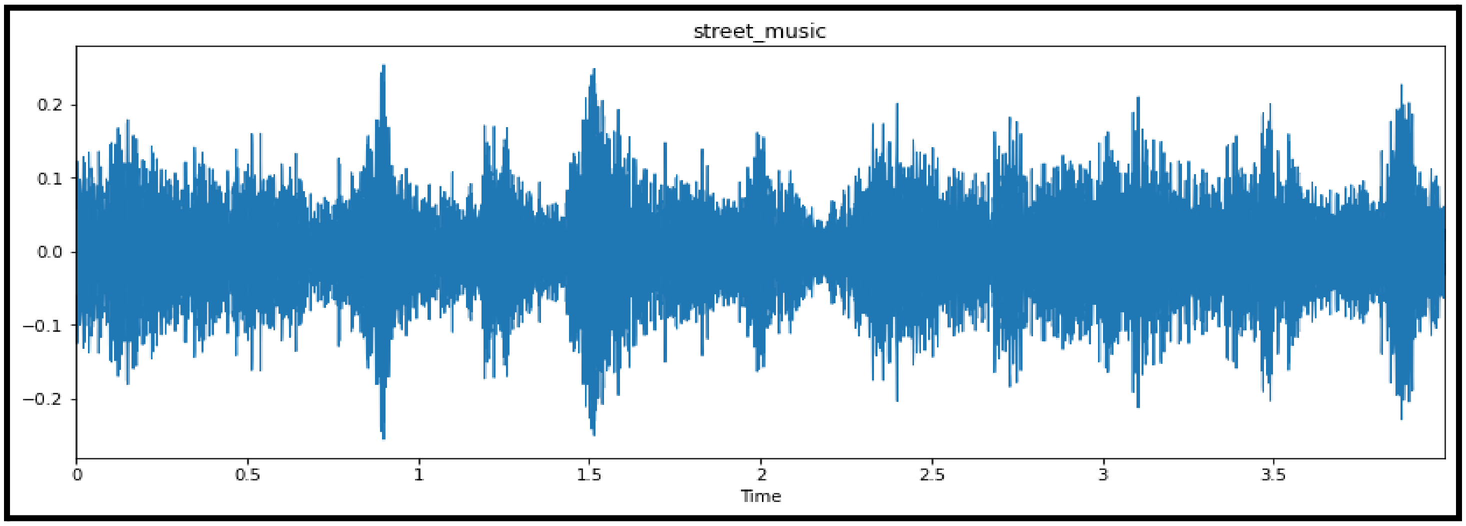
Distribution of Dataset

Class	**Label**	**Number of Tracks**
Air Conditioner	0	1000
Car Horn	1	429
Children Playing	2	1000
Dog bark	3	1000
Drilling	4	1000
Engine Idling	5	1000

Gun Shot	6	374
Jackhammer	7	1000
Siren	8	929
Street Music	9	1000

In this project, I have used data from 80% audios to train our model, and used the other 20% to test the performance of the models. Fig. 1 is an example of a sound wave from raw data.

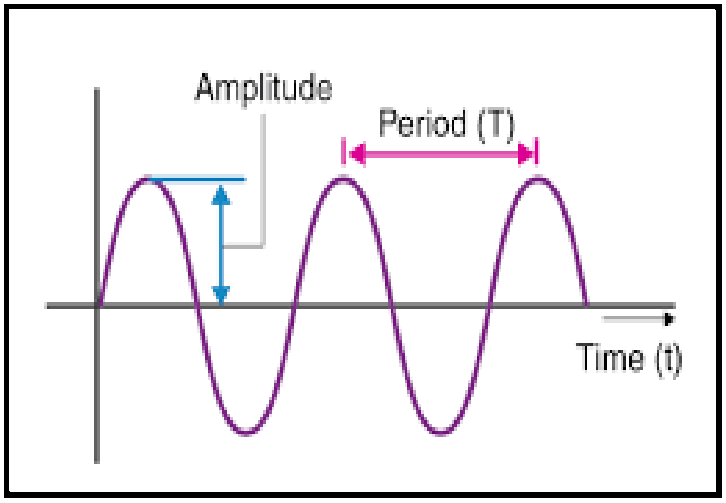
Sample of soundwave of street music

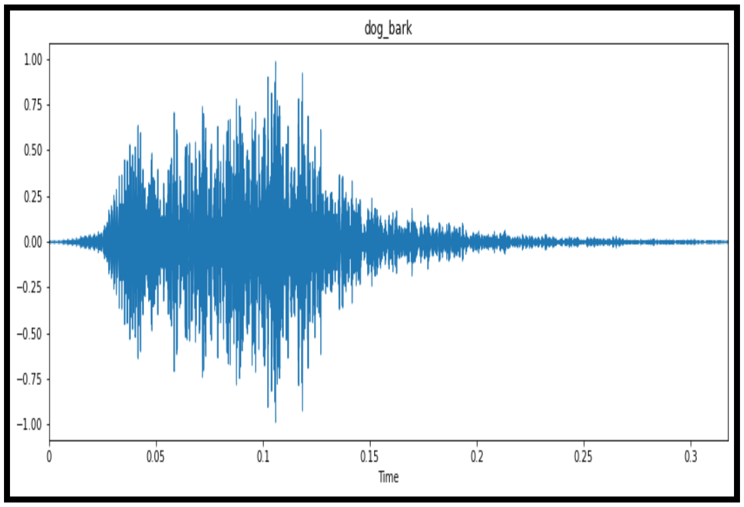


B. Understanding the Audio Data

Audio Signal: Audio Signal: A complex signal composed of multiple "single - frequency" sound waves. When sound is recorded, we only capture the resultant amplitude of those multiple waves. A signal is a variation of a certain quantity over time.

For audio, the quantity that varies is air pressure.





Two common ways to represent sound:

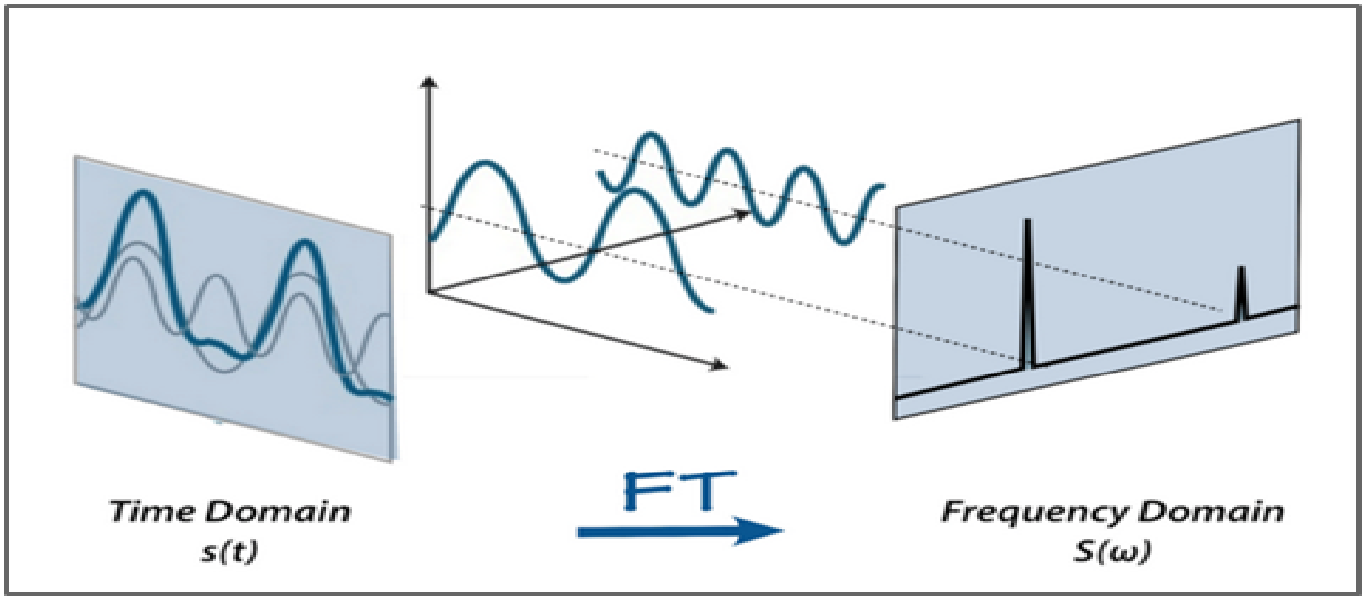
- **Time domain:** each sample represents the variation in air pressure.
- **Frequency domain:** at each time stamp we indicate the amplitude for each frequency.

To capture this audio data, we take samples of the air pressure over time. The rate at which we sample the data can vary, but is most commonly 44.1kHz, or 44,100 amplitude samples per second. This data that we have captured is called Audio Waveform. This audio waveform can be plotted using amplitude and sampling-rate that we can get using python's librosa library's load() function.

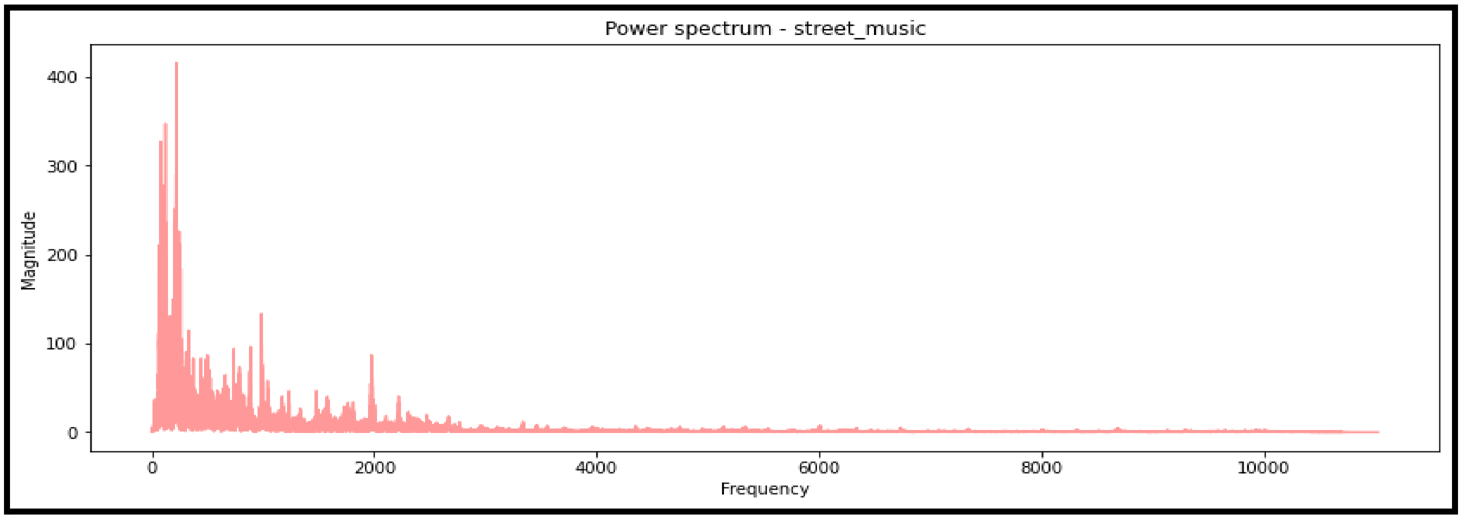
When the sound wave is recorded we only capture the resultant amplitudes of those waves.And these amplitudes are not very informative because they only talk about loudness of audio recording (Amplitude = 0 means silence). To understand audio signals, it is necessary to transform it into frequency-domain. Therefore, to capture other important details, we perform various kinds of transformations.

Fast Fourier Transform

An audio signal is composed of several single-frequency sound waves. When taking samples of the signal over time, we only capture the resulting amplitudes. The Fourier transform is a mathematical formula that allows us to decompose a signal into its individual frequencies and the frequency's amplitude.



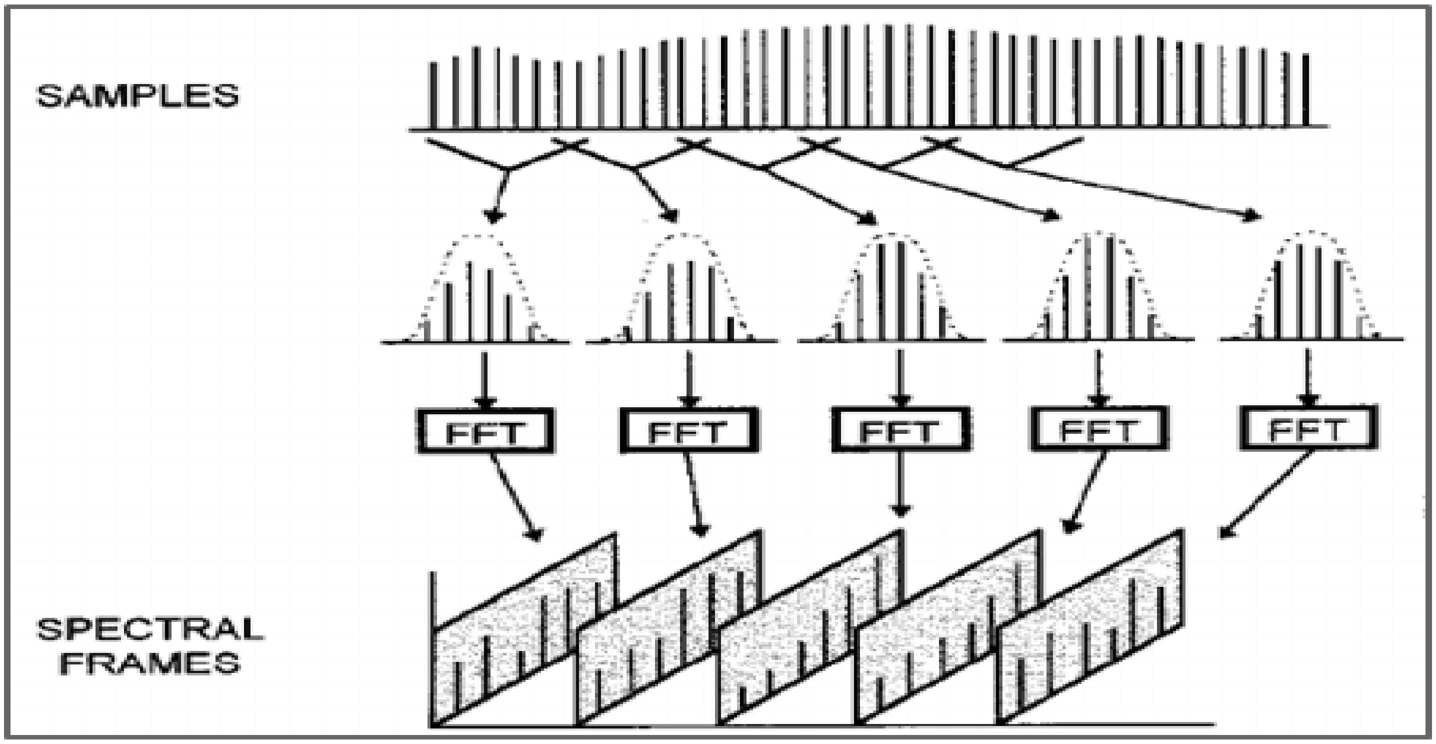
It converts the signal from the time domain into the frequency domain. The result is called a spectrum. Basically, Fourier Transform decomposes complex periodic sound into a sum of sine waves oscillating at different frequencies. Following is the spectrum I got after applying Fourier Transformation of audio from street music class:



But, when we apply FFT, we lose time information. And, if we lose time information then our system won't be able to tell which music sound played first if we use only frequencies as a feature. And thus we will go for the next transformation.

Spectrograms & Short Term Fourier Transform(STFT)

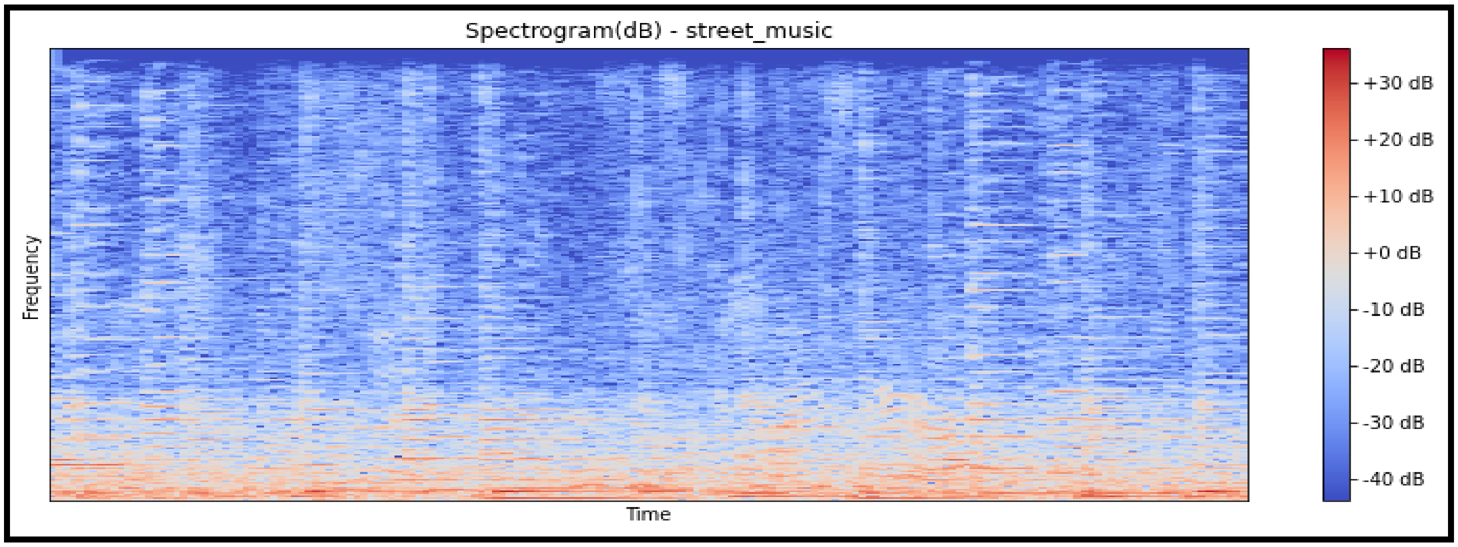
Spectrograms are introduced because we were losing time information after applying Fast Fourier Transformation. A spectrogram shows how the frequency content of a signal changes over time and can be calculated from the time domain signal. Short Term Fourier Transform comes to our rescue here.

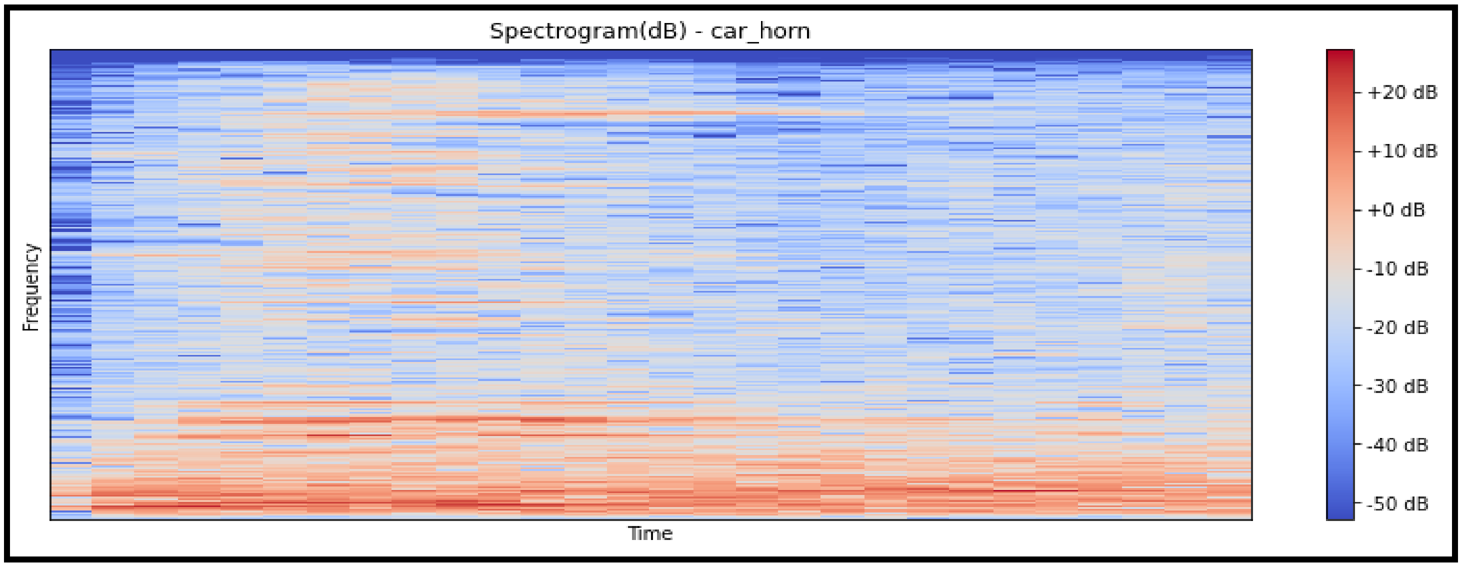


A spectrogram is a visual representation of the spectrum of frequencies of sound or other signals as they vary with time. It's a representation of frequencies changing with respect to time for given music signals.

How Spectrogram Preserves Time Information?

When we perform STFT on any audio data, the first step is to break the entire audio signal into smaller frames (or windows) and then for each window we calculate FFT. This way we will be getting frequencies for each window and window number will represent time information. Generally, window size is between 20ms to 30ms long. These frames must overlap with each other such that we do not lose any information. Generally, overlapping of 25% to 75% is done. Following is the Spectrogram after applying STFT on our one of the audio file from street music and car horn class:





As a result of STFT, we got a spectrogram which gives us information about magnitude as a function of frequency and time. Audio that sounds drastically different, results in drastically different spectrograms.

Steps involved in extracting features from audio data :

- I took samples of air pressure over time to digitally represent an audio signal.
- Mapped the audio signal from the time domain to the frequency domain using the fast Fourier transform, and we performed this on overlapping windowed segments of the audio signal.
- Converted the y-axis (frequency) to a log scale and the color dimension (amplitude) to decibels to form the spectrogram.
- Mapped the y-axis (frequency) onto the mel scale to form the mel spectrogram.

C. Pre-Processing & Feature Extraction from Audio Files

UrbanSound8K dataset has 8732 music files and each file is less than or equal to 4 seconds long. Now, to train the CNN model, I have extracted MFCC features from these audio files. Entire database of extracted features was created and stored in the drive.

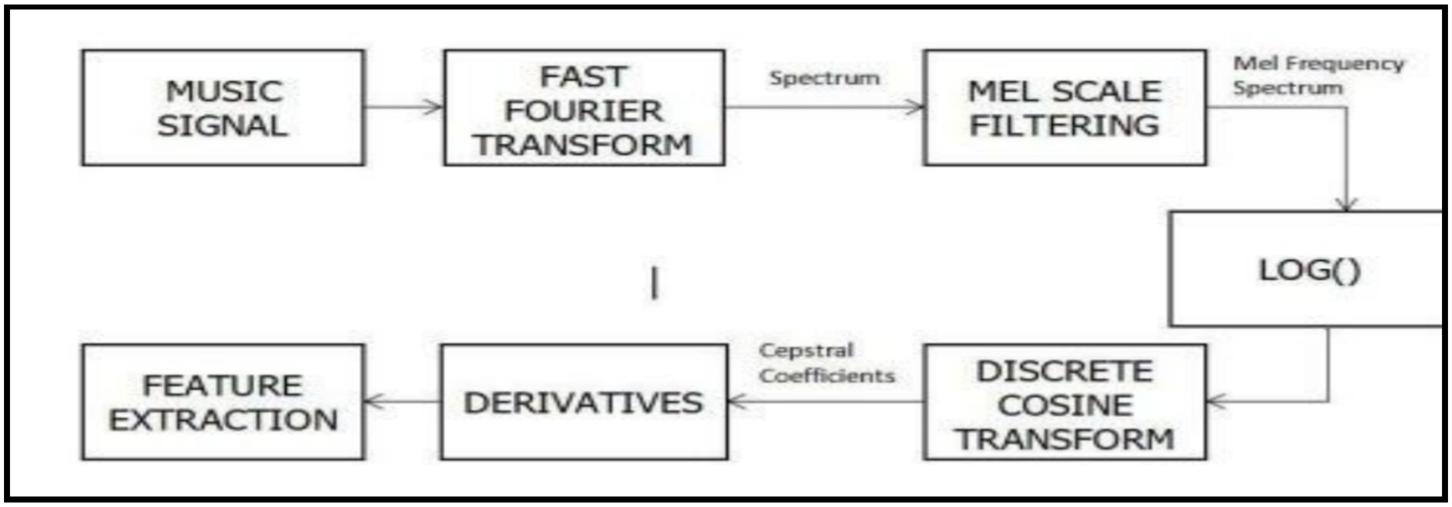
Mel - Frequency Cepstral Coefficient (MFCC)

The Mel frequency cepstral coefficients (MFCCs) of a signal are a small set of features (usually about 10–20) which concisely describe the overall shape of a spectral envelope. It models the characteristics of the human voice.

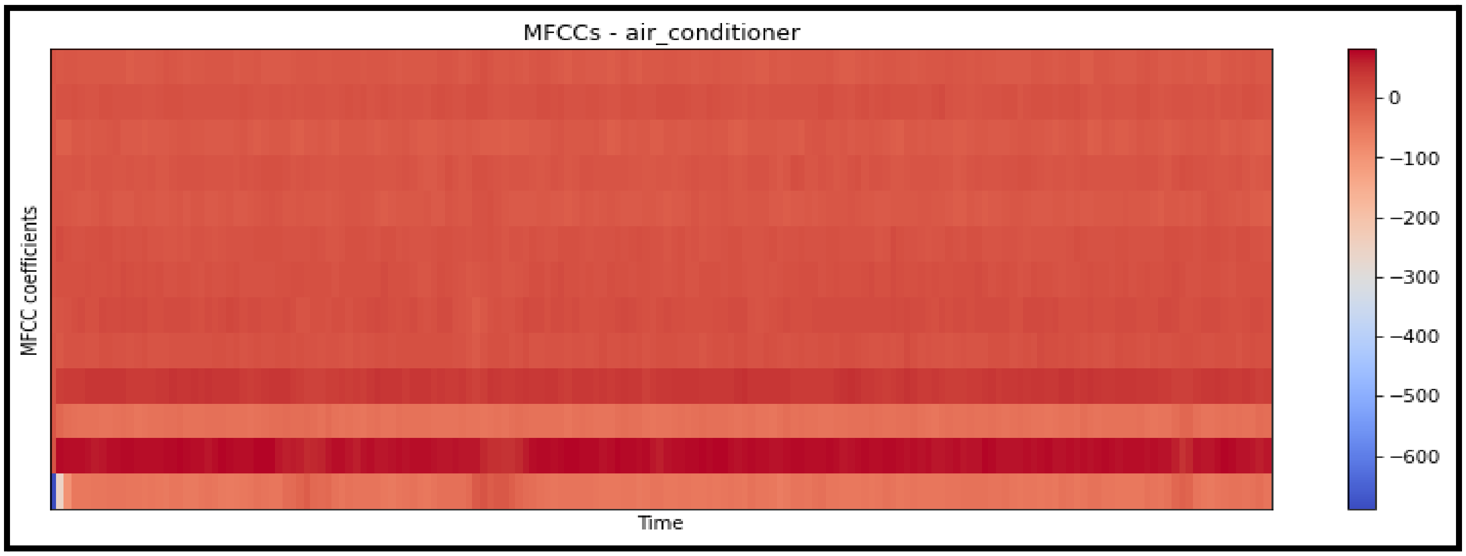
Pitch is one of the characteristics of a speech signal and is measured as the frequency of the signal. Mel scale is a scale that relates the perceived frequency of a tone to the actual measured frequency. It scales the frequency in order to match more closely what the human ear can hear (humans are better at identifying small changes in speech at lower frequencies). A frequency measured in Hertz (f) is converted to the Mel scale using the following formula :

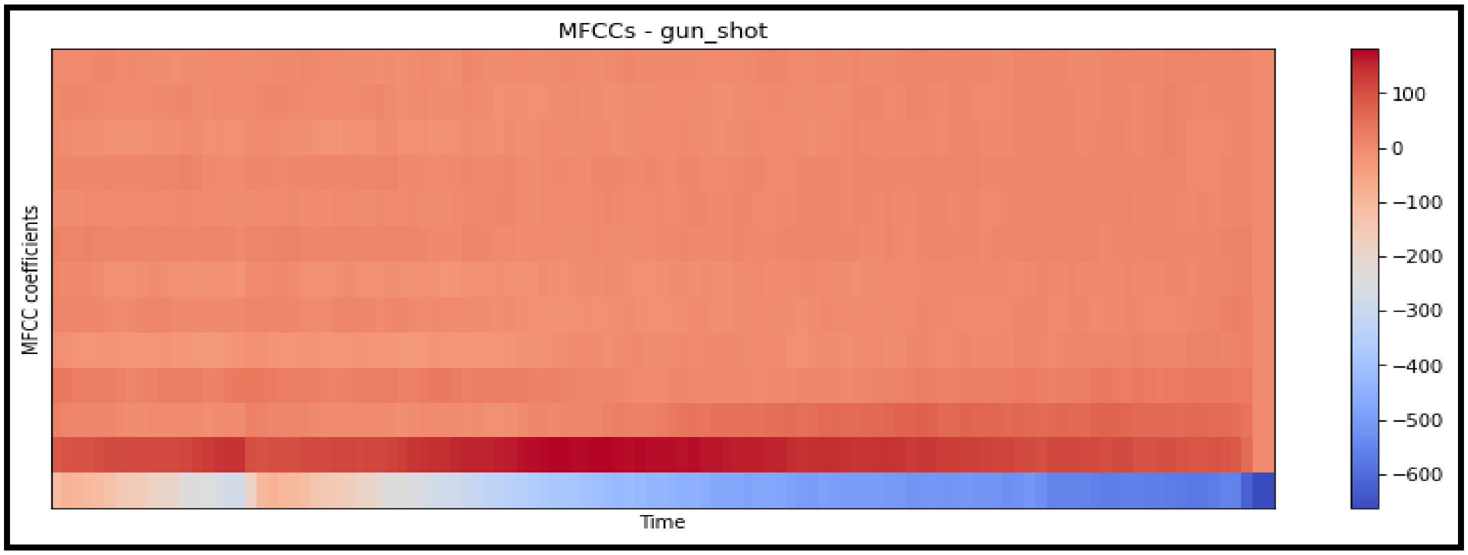
$$\boxed{\mathrm{Mel}(f) = 2595 \log\left(1 + \frac{f}{700}\right)}$$

Any sound generated by humans is determined by the shape of their vocal tract. If this shape can be determined correctly, any sound produced can be accurately represented. The envelope of the time power spectrum of the speech signal is representative of the vocal tract and MFCC (which is nothing but the coefficients that make up the Mel-frequency cepstrum) accurately represents this envelope. The following block diagram is a step-wise summary of how we arrived at MFCCs:



After performing all these steps I got the extracted MFCC features. Following are the visual representation of audio file after extracting MFCC:

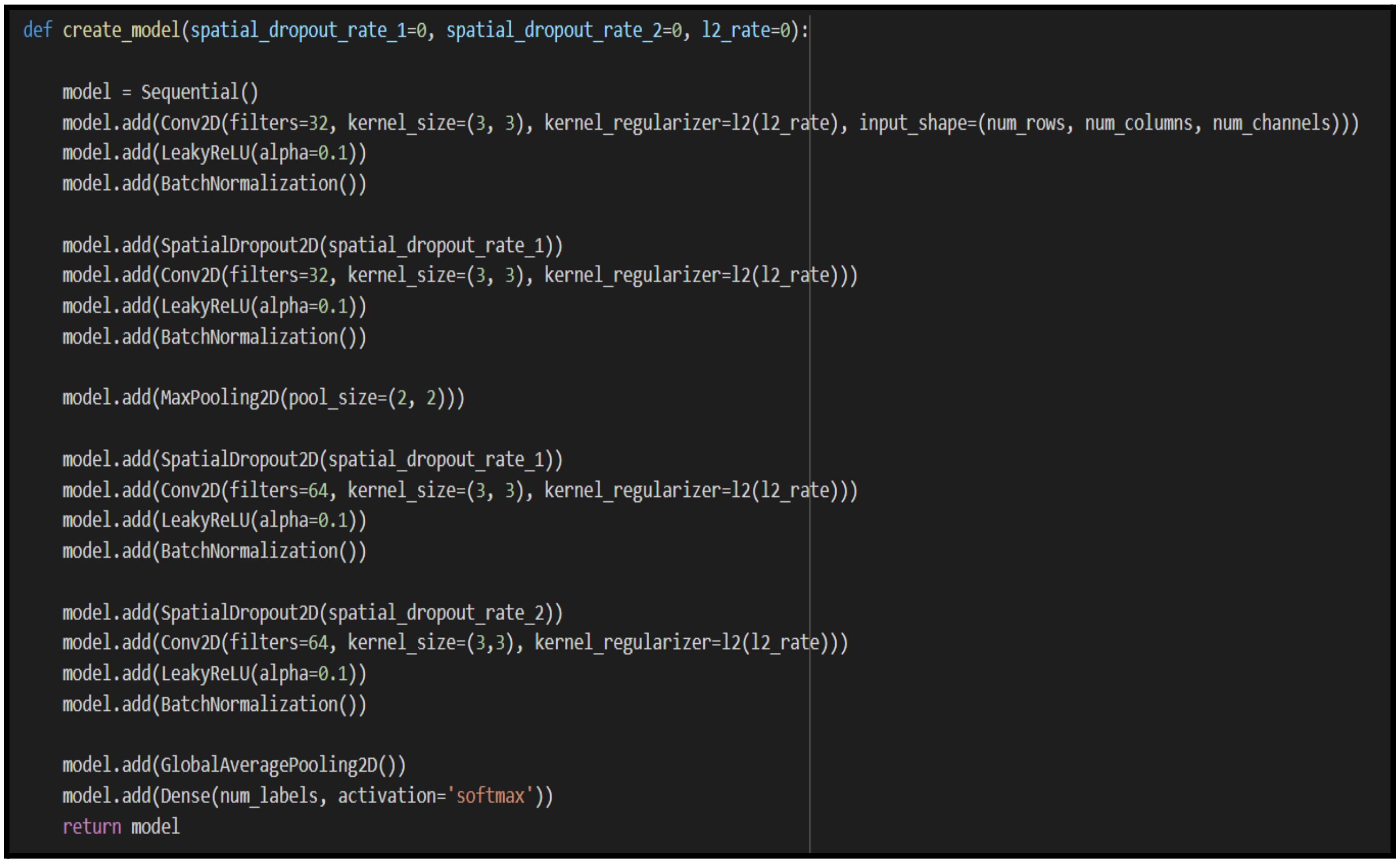




D. Convolutional Neural Network

CNN is a Deep Learning algorithm which can take an input image as input, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. A CNN has various layers such as Convolutional layers, ReLU layers, Pooling layers and Fully connected dense layers. CNN is widely used for image classification because it does automatic feature extraction using convolution. Following are the details of our CNN model :

Code Snapshot for CNN model :

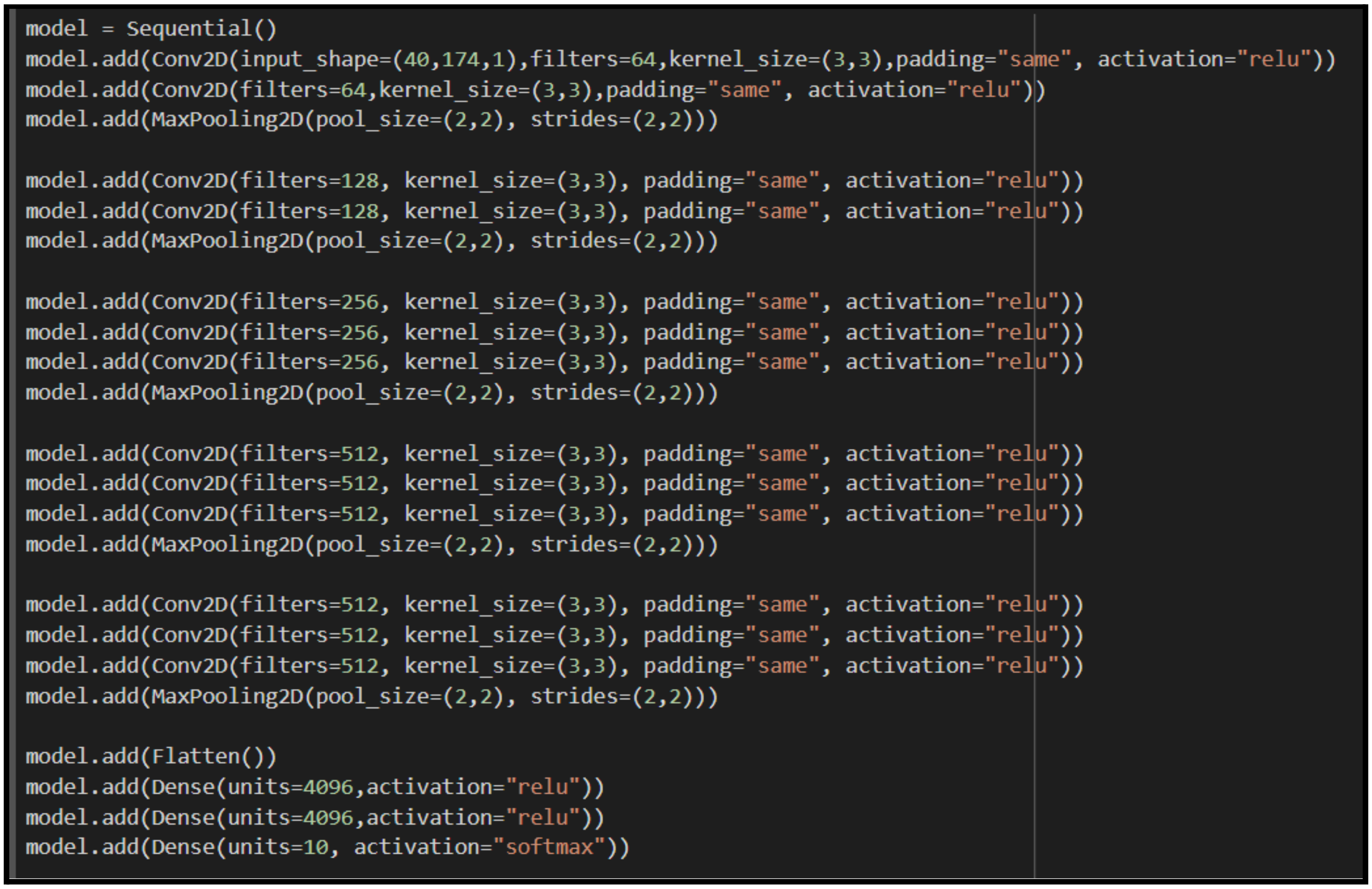


```
def create_model(spatial_dropout_rate_1=0, spatial_dropout_rate_2=0, l2_rate=0):

    model = Sequential()
    model.add(Conv2D(filters=32, kernel_size=(3, 3), kernel_regularizer=l2(l2_rate), input_shape=(num_rows, num_columns, num_channels)))
    model.add(LeakyReLU(alpha=0.1))
    model.add(BatchNormalization())

    model.add(SpatialDropout2D(spatial_dropout_rate_1))
    model.add(Conv2D(filters=32, kernel_size=(3, 3), kernel_regularizer=l2(l2_rate)))
    model.add(LeakyReLU(alpha=0.1))
    model.add(BatchNormalization())

    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(SpatialDropout2D(spatial_dropout_rate_1))
    model.add(Conv2D(filters=64, kernel_size=(3, 3), kernel_regularizer=l2(l2_rate)))
    model.add(LeakyReLU(alpha=0.1))
    model.add(BatchNormalization())

    model.add(SpatialDropout2D(spatial_dropout_rate_2))
    model.add(Conv2D(filters=64, kernel_size=(3,3), kernel_regularizer=l2(l2_rate)))
    model.add(LeakyReLU(alpha=0.1))
    model.add(BatchNormalization())

    model.add(GlobalAveragePooling2D())
    model.add(Dense(num_labels, activation='softmax'))
    return model
```

VGG-16 Model

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.

Code Snapshot for VGG-16 model :

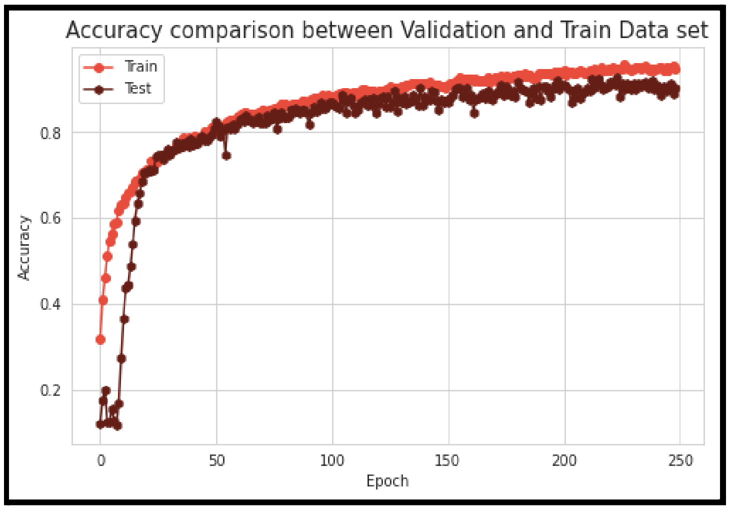


```
model = Sequential()
model.add(Conv2D(input_shape=(40,174,1),filters=64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(Conv2D(filters=64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Flatten())
model.add(Dense(units=4096,activation="relu"))
model.add(Dense(units=4096,activation="relu"))
model.add(Dense(units=10, activation="softmax"))
```

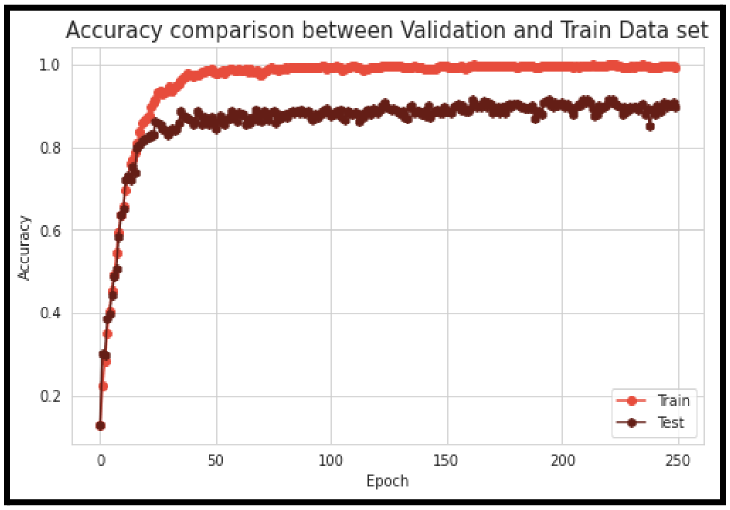
E. Experiments & Results

Both the above CNN and VGG16 model were trained for 250 epochs with a leaning rate of 1e-4 using Adam optimization algorithm. Batch size of 128 was used while training. Following is the accuracy plots and table that shows the train and test accuracy achieved by both these models:

Accuracy plot of CNN



Accuracy plot of VGG16



Model	**Train Accuracy**	**Test Accuracy**
CNN	95.80	90.98
VGG16	98.38	91.28

CONCLUSION

This project implemented the application which performs Urban Sound Classification using Deep Learning techniques. The application uses Convolutional Neural Network model and VGG16 model to perform the classification. MFCC features of each track from the UrbanSound8K dataset are obtained. This is done by using the librosa package of python. After training the model for 250 epochs I got the training accuracy of 95.80% and test accuracy of 90.98% with CNN model and achieved the training accuracy of 98.38% and test accuracy of 91.28%.

REFERENCES

1. Urban sound classification based on 2-order dense convolutional network using dual features

https://www.sciencedirect.com/science/article/abs/pii/S0003682X19312691#:~:text=A%20novel%20urban%20sound%20classification%20model%20is%20proposed%20based%20on,method%20for%20urban%20sound%20classification.

2. Urban sound event classification based on local and global features aggregation

https://www.sciencedirect.com/science/article/abs/pii/S0003682X16302274#:~:text=Experimental%20results%20demonstrated%20that%20the,for%20urban%20sound%20events%20classification.&text=According%20to%20experimental%20results%2C%20conditional,obtained%20superior%20sound%20classification%20accuracy.

3. UrbanSound8K Dataset

https://www.kaggle.com/chrisfilo/urbansound8k