**A Project on**

# Audio Classification Using Deep Learning

**Submitted for partial fulfillment of award of**

**MASTER OF SCIENCE**

## Dr. Homi Bhabha State University

**Degree in**
**MATHEMATICS**

By


**Srushti Kadam**
**Deepak Patil**


Under the Guidance of

**Dr. Selby Jose**


**Department of Mathematics**
**The Institute of Science**
**Mumbai - 400032**


April, 2023

# Declaration

I hereby declare that the project work entitled "**AUDIO CLASSIFICATION USING DEEP LEARNING**" carried out at the Department of Mathematics, The Institute of Science, Mumbai, is a record of an original work done by me under the guidance of **Dr. Selby Jose**, The Institute of Science, and this project work is submitted in the partial fulfillment of the requirements for the award of the degree of Master of Science in Mathematics, University of Mumbai. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

**Srushti Kadam**
Seat No. 43180008
**Deepak Patil**
Seat No.43180014

# Certificate

This is to certify that the project report entitled **AUDIO CLASSIFICATION USING DEEP LEARNING**, carried out at the Department of Mathematics, The Institute of Science, Mumbai, in partial fulfillment for the award of the degree of Master of Science in Mathematics, University of Mumbai, is a record of bonafide work carried out by **Ms.Srushti Kadam**, Seat No. **43180008** and **Mr.Deepak Patil**,Seat No.**43180014**, under the supervision and guidance of **Dr. Selby Jose** during the academic year 2022 – 2023.

| | | |
|---|---|---|
| **Dr. Selby Jose** | **External Examiner** | **Head of Dept.** |
| **Project Guide** | | **Mathematics** |

**Place: Mumbai**

**Date: 26 April 2023**

# Acknowledgment

We want to start by thanking Dr. Selby Jose, our guide, for everything. His extensive expertise and experience in the area of Machine Learning helped us complete this assignment successfully. Without his assistance and direction, this undertaking would not have been possible. The Dr. Homi Bhabha State University gave us the chance to work on the Audio Classification using Deep Learning, and we would like to thank them for that. Last but not least, we would like to express our sincere gratitude to everyone who helped make this initiative a success. ...

**Srushti Kadam**

**Deepak Patil**

# Abstract

In the context of Audio classification, deep learning algorithms are used to extract features from audio signals, such as pitch, tempo, and frequency, which are then used to identify words and phrases spoken by a user. First, deep learning algorithms are able to learn from vast amounts of data, making them ideal for tasks such as speech recognition that require large amounts of training data. Finally, deep learning algorithms are highly flexible and can be used for a wide range of voice recognition tasks, from simple keyword spotting to complex natural language understanding. One of the most common deep learning architectures used for Audio classification is the recurrent neural network (RNN). RNNs are able to capture temporal dependencies in audio signals, making them well-suited for speech recognition tasks. Another popular deep learning architecture for Audio classification is the convolutional neural network (CNN). CNNs are commonly used in image recognition tasks, but they have also been applied successfully to audio signals. CNNs work by applying convolutional filters to audio signals to extract features at different levels of abstraction. Key words - recurrent neural network (RNN), the long short-term memory (LSTM) architecture,convolutional neural network (CNN). We are using Artificial Neural Networking(ANN) for the model creation purpose in deep learning.

# Contents

**1 Introduction** **1**

**2 Preliminaries** **3**

2.0.1 Working of ANN model in General . . . . . 5

2.1 Exploratory Data Analysis . . . . . 6

**3 Data Pre-Processing** **12**

3.1 Mel Frequency Cepstral Coefficient (MFCC) . . . . . 15

3.1.1 Mel Frequency/Spectrogram . . . . . 15

3.1.2 Cepstral Coefficient/Cepstrum . . . . . 22

3.1.3 MFCC Extract Features . . . . . 29

**4 Model Creation & Testing ANN Model** **35**

**Bibliography** **43**

# Chapter 1

# Introduction

Audio classification using deep learning is a relatively new field of research that has gained momentum in recent years. Traditional audio classification systems relied on hand-crafted features and classical machine learning algorithms, which were often limited in their ability to handle complex audio classification tasks. However, deep learning techniques have shown the ability to learn complex patterns and features directly from raw audio data, resulting in more accurate and robust audio classification systems. With the availability of large-scale audio datasets and advancements in deep learning techniques, there has been a growing interest in exploring new architectures, optimization techniques, and training strategies for audio classification using deep learning. This has resulted in the development of various deep learning-based audio classification systems for different applications.

We were interested in audio classification using deep learning because of its potential real-world applications in areas such as speech recognition, music recommendation, and environmental monitoring. Moreover, deep learning techniques have shown promising results in improving the accuracy and

efficiency of audio classification systems compared to traditional methods. Additionally, the availability of large-scale audio datasets has fueled interest in this field, allowing researchers to train and test their models on a diverse range of audio samples.

This project is further divided into four chapter/parts

1. EDA part (Exploratory data analysis)
2. Data pre-processing part
3. To create a model
4. To implement model on training data set(i.e Testing the Model)

Audio is also a very difficult domain we need to understand the basic knowledge of audio itself like whenever we talking there is a disturbance in the air it create a sound. Just imagine that This is stagnant air so no sound will come unless and until you don't disturb that specific part of the air itself and usually audio is basically given in a signals that we are going to see.
The main agenda for audio classification using deep learning is to develop accurate and efficient systems that can automatically classify audio signals into different categories or classes. This involves training deep neural network models on large-scale audio datasets to learn patterns and features that are representative of different audio classes. The trained models can then be used to classify new audio samples into the appropriate classes.

The ultimate goal is to develop audio classification systems that are robust, reliable, and can accurately classify a diverse range of audio signals, including music, speech, and environmental sounds. Such systems have many potential applications, such as in speech recognition, music recommendation, audio search and retrieval, and environmental monitoring.

# Chapter 2

# Preliminaries

This chapter contains the basic definitions which are referred from the books.After the definitions we are actually start our Exploratory Data Analysis Part.

**Definition 2.0.1 (Audio classification)** *: The task of assigning one or more labels to an audio sample based on its content*

**Definition 2.0.2 (Deep learning)** *: A subset of Machine Learning that uses Artificial Neural Networks with multiple layers to extract high-level features from data.*

**Definition 2.0.3 (Artificial Neural Network (ANN))** *: ANN is a computational model that mimics the structure and function of the human brain to learn from data.*

**Definition 2.0.4 (Convolutional neural network (CNN))** *: A type of neural network commonly used for image and audio classification that uses convolutional layers to learn feature representations.*

**Definition 2.0.5 (Mel spectrogram)** *: A visual representation of the power spectrum of an audio signal, commonly used as input for audio classification models*

**Definition 2.0.6 (Transfer learning)** *: A technique in deep learning where a pre-trained model is used as a starting point for a new task, often fine-tuning the weights for specific purposes*

**Definition 2.0.7 (Overfitting)** *: When a model becomes too complex and starts to fit the training data too closely, resulting in poor performance on new, unseen data.*

**Definition 2.0.8 (Underfitting)** *: When a model is too simple and is not able to capture the complexity of the data, resulting in poor performance on both the training and test data.*

**Definition 2.0.9 (Accuracy)** *: The percentage of correctly classified audio samples in a dataset, often used as a performance metric for audio classification models.*

**Definition 2.0.10 (Loss function)** *: A function that measures the difference between the predicted output of a model and the true output, often used to train deep learning models.*

**Definition 2.0.11 (Epoch)** *: A single pass through the entire training dataset during the training process of a deep learning model.*

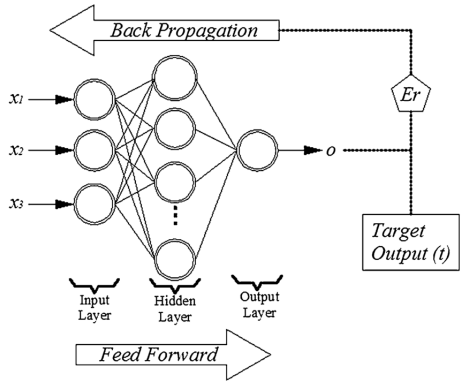
### 2.0.1 Working of ANN model in General

An Artificial Neural Network (ANN) works by processing input data through a series of interconnected layers of artificial neurons, also called nodes or units. Each neuron receives one or more inputs, performs a computation, and produces an output that is passed on to the next layer of neurons.

The computation performed by a neuron is typically a weighted sum of its inputs, followed by a non-linear activation function that introduces non-linearity into the network. The weights of the connections between neurons are learned during training using an optimization algorithm such as back-propagation.

The input layer of the ANN receives the raw data, such as an image, audio clip, or text. The input data is then transformed and propagated through one or more hidden layers, where the network learns to extract higher-level features from the data. Finally, the output layer produces a prediction or classification based on the learned features.

ANNs can be used for a variety of tasks, such as image and speech recognition, natural language processing, and control systems. They can be trained using supervised, unsupervised, or reinforcement learning algorithms, depending on the task and the available data.



## 2.1 Exploratory Data Analysis

Audio is also a very difficult domain we need to understand the basic knowledge of audio itself like whenever we talking there is a disturbance in the air it create a sound. Just imagine that This is stagnant air so no sound will come unless and until you don't disturb that specific part of the air itself and usually audio is basically given in a signals that we are going to see. So our main agenda is to implement the project in a such way that later on whichever project or whichever audio classification we want to do it will help us to do that.

Let's begin now first of all we will start with what will be the data set itself, data set is pretty important. So this dataset contains 8732 labeled sound which are ($\leq 4s$) of urban sounds from 10 classes: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, and street music. The classes are drawn from the urban sound toxinonmy 8732 audio files of urban sounds in WAV format. The sampling rate, bit depth, and number of channels are the same as those of the original file uploaded to Free sound. So, for our project we are installing following libraries:

- **Librosa:-** This library works very well with sound and signals probably the sound data set itself. To read that particular signals, to find out the sample rate or to find how many channels it is and many more things. So Librosa is the library that we are going to use for this project.
- **Skype library:-** Apart from Librosa library there is also a library called Skype which will actually help us to read the wave signals. Usually all the audio files that you are seeing will be in the form of extension

as **.wav file.** So we are going to use two libraries.

- **Ipython library:-** -Ipython means Interactive python. It is an interactive command line terminal for python.it will provide an Ipython terminal and web-based platform for python computing. It has more advanced features than the python standard interpreter and will quickly execute a single line of a python code. This library will actually help us to display some of the graphs in particular manner of the sound waves which is called as **python.display as ipd.**

- **Matplotlib:-** This library is very useful for Plotting graphs.

- **Pandas:-** Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis".

```
In [2]: import matplotlib.pyplot as plt
        %matplotlib inline

In [3]: filename="C:\\Users\\hp\\Downloads\\UrbanSound8K\\UrbanSound8K\\Dog_bark.wav"

In [4]: import IPython.display as ipd
        import librosa
        import librosa.display

In [5]: !dir

 Volume in drive C has no label.
 Volume Serial Number is 4EDD-13EA

 Directory of C:\Users\hp\OneDrive\Desktop\Test jupyter

2023-04-23  02:51 PM    <DIR>          .
2023-04-23  09:50 AM    <DIR>          ..
2023-04-23  01:59 PM    <DIR>          .ipynb_checkpoints
2023-04-23  11:26 AM            40,403 Audio Classification Using Deep learni
ng.ipynb
2023-04-23  02:51 PM         2,027,168 Audio Classification.ipynb
2023-04-06  11:39 AM            52,086 Machine learning assginment.ipynb
               3 File(s)      2,119,657 bytes
               3 Dir(s)  429,198,901,248 bytes free
```

**.Audio** is a function which will actually help us to load the audio that is in the form of wave file. It will create an audio object.

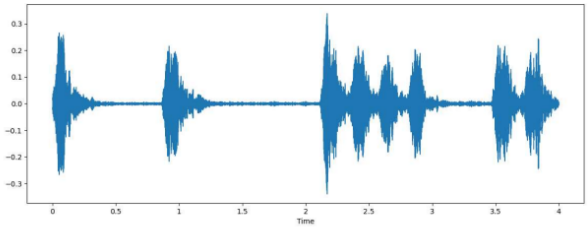
**Librosa.Load** will load the audio file

**Sample Rate:-** In audio production, a sample rate (or "sampling rate") defines how many times per second a sound is sampled. The standard sample rate used for audio CDs is 44.1 Kilohertz (44,100 hertz). That means each second of a song on a CD contains 44,100 individual samples. Audio wanted some kind of sample rate and just by displaying the wave plot we will be able to find out some information regarding the sample rate and this is by default read that particular sample rate.The data we have is a filename and where it is present so let us explore 1st file, so it is present in fold 5 with category as a dog bark. Now we know about the audio files and how to visualize

In [6]:
```
plt.figure(figsize=(14,5))
data,sample_rate=librosa.load(filename)
librosa.display.waveshow(data,sr=sample_rate)
ipd.Audio(filename)
```

Out[6]:

0:00 / 0:00




In [7]:
```
sample_rate
```

Out[7]: 22050

In [50]:
```
import pandas as pd
metadata=pd.read_csv("C:\\Users\\hp\\Downloads\\UrbanSound8K\\UrbanSound8K\\me
metadata.head()
```

Out[50]:

| | slice_file_name | fsID | start | end | salience | fold | classID | class |
|---|---|---|---|---|---|---|---|---|
| 0 | 100032-3-0-0.wav | 100032.0 | 0.0 | 0.317551 | 1.0 | 5.0 | 3.0 | dog_bark |
| 1 | 100263-2-0-117.wav | 100263.0 | 58.5 | 62.500000 | 1.0 | 5.0 | 2.0 | children_playing |
| 2 | 100263-2-0-121.wav | 100263.0 | 60.5 | 64.500000 | 1.0 | 5.0 | 2.0 | children_playing |
| 3 | 100263-2-0-126.wav | 100263.0 | 63.0 | 67.000000 | 1.0 | 5.0 | 2.0 | children_playing |
| 4 | 100263-2-0-137.wav | 100263.0 | 68.5 | 72.500000 | 1.0 | 5.0 | 2.0 | children_playing |

them in audio format. Moving format to data exploration we will load the CSV data file provided for each audio file and check how many records we have for each class. So, What we are doing in EDA part is that When we are reading this information with Librosa then what happens is that we are reading the signals with the sample rate, That basically means whenever through Librosa we reading a dataset with respect to some specific audio we are actually getting the sample rate similarly if we try to see not only this particular way which will be reading the audio. There are different ways we can also use Skype library but understand one thing over here Librosa will play a very important role here, because in a audio there may be different different audio sample rates also like sometimes in that particular audio if we talk about channels there are two channels in a specific audio one is called as stereo and one is called as mono.

```
In [52]: metadata.isnull().value_counts()

Out[52]: slice_file_name  fsID   start  end    salience  fold   classID  class
         False            False  False  False  False     False  False    False    8732
         True             True   True   True   True      True   True     True        4
         dtype: int64

In [53]: import pandas as pd
         metadata.dropna(inplace= True)
         print(metadata.to_string())

                 slice_file_name      fsID       start         end  salience  fold
         classID            class
         0       100032-3-0-0.wav  100032.0    0.000000    0.317551       1.0   5.0
         3.0             dog_bark
         1     100263-2-0-117.wav  100263.0   58.500000   62.500000       1.0   5.0
         2.0   children_playing
         2     100263-2-0-121.wav  100263.0   60.500000   64.500000       1.0   5.0
         2.0   children_playing
         3     100263-2-0-126.wav  100263.0   63.000000   67.000000       1.0   5.0
         2.0   children_playing
         4     100263-2-0-137.wav  100263.0   68.500000   72.500000       1.0   5.0
         2.0   children_playing
         5     100263-2-0-143.wav  100263.0   71.500000   75.500000       1.0   5.0
         2.0   children_playing
         6     100263-2-0-161.wav  100263.0   80.500000   84.500000       1.0   5.0
         2.0   children_playing
         7       100263-2-0-3.wav  100263.0    1.500000    5.500000       1.0   5.0
         2.0   children_playing
         8      100263-2-0-36.wav  100263.0   18.000000   22.000000       1.0   5.0

In [54]: metadata.isnull().value_counts()

Out[54]: slice_file_name  fsID   start  end    salience  fold   classID  class
         False            False  False  False  False     False  False    False    8732
         dtype: int64
```

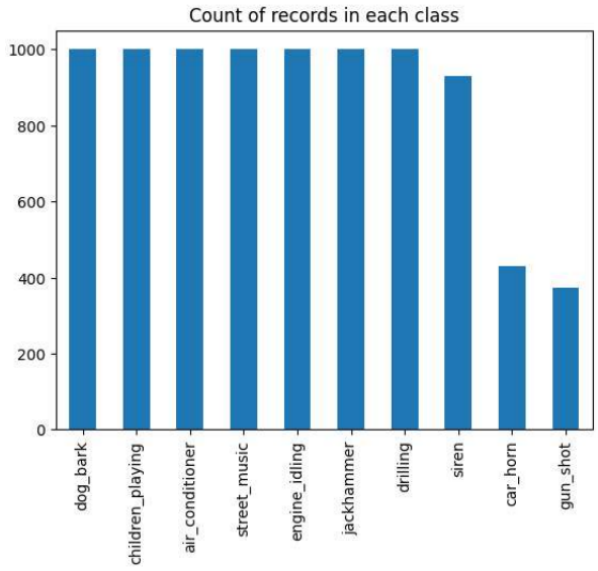
We check for null values and we found that there are some null values in our dataset so we remove null values using **Dropna** method. after removing null values we check again for confirmation is there null values removed or not. So we found out that there are no null values in the dataset After removing null values we check for how many rows are there in the metadata with the help of **.value counts** and we plot the bar graph.

```
In [58]: metadata['class'].value_counts()

Out[58]: dog_bark            1000
         children_playing    1000
         air_conditioner     1000
         street_music        1000
         engine_idling       1000
         jackhammer          1000
         drilling            1000
         siren                929
         car_horn             429
         gun_shot             374
         Name: class, dtype: int64

In [79]: metadata['class'].value_counts().plot(kind='bar')
         plt.title('Count of records in each class')

Out[79]: Text(0.5, 1.0, 'Count of records in each class')
```



Let us take an example of mono, suppose we have particular headphones, in mono what happens if I wear this headphones both the signals will equally reaching to both my earbuds probably the ear parts. Suppose I'm listening to a song so that particular signals will get divided from here and it will be giving us the information equally Whereas in the case of stereo some of the different signals may come at the right hand side of my ear plug or the left hand side of my ear plug. Suppose there may be audio signals which will give a bar sound over here and there will be a sound signal over here like that so that is the basic difference between mono and stereo.
Here Our Exploratory Data Analysis part is Over.....

# Chapter 3

# Data Pre-Processing

In previous Chapter we did the Data Analysis part and Now we are doing Data Pre-Processing Part in this Chapter

Librosa is a Python package for music and audio processing by Brian McFee and will allow us to load audio in our notebook as a numpy array for analysis and manipulation. For much of the preprocessing we will be able to use **Librosa.load()** function, which by default converts the sampling rate to 22.05 KHz, normalise the data so, the bit-depth values range between -1 and 1 and flattens the audio channels into mono.

```
audio_file_path
```

Out[21]:

```
'C:/Users/Srushti Kadam/Contacts/UrbanSound8K/audio/fold5/100263-2-0-3.wa
v'
```

In [22]:

```
librosa.load(audio_file_path)
```

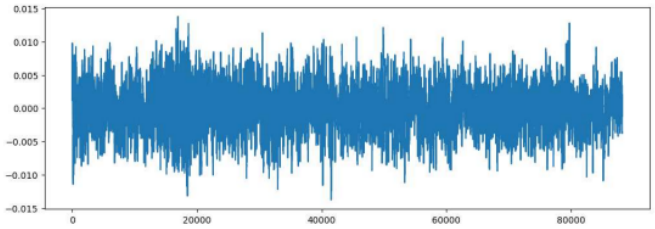
Out[22]:

```
(array([ 0.00331575,  0.00467553,  0.00361099, ..., -0.00376796,
        -0.00347471, -0.00357828], dtype=float32),
 22050)
```

```
### Lets plot the librosa audio data
import matplotlib.pyplot as plt
# Original audio with 1 channel
plt.figure(figsize=(12, 4))
plt.plot(librosa_audio_data)
```

Out[23]:

```
[<matplotlib.lines.Line2D at 0x231a2ee22e0>]
```




**observation**

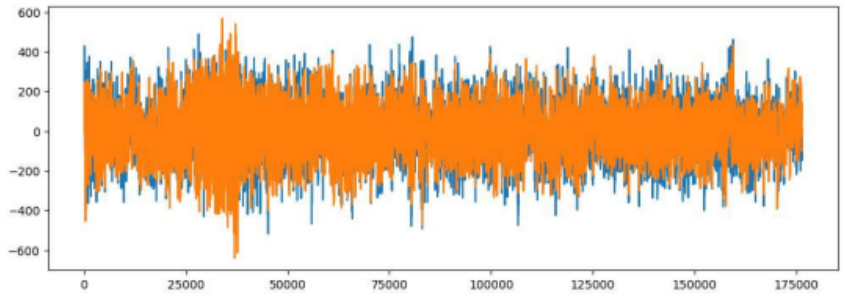
Here librosa converts the signals to mono

As we say in part 1 that audio has 2 channels mono and stero and librosa converts by default stereo into mono channel. There is another library called skype which is also used for audio classification but Librosa play an important because skype libarary dont change the audio channel it will keep that channel as it as. we try to visualize this,

```
import matplotlib.pyplot as plt
# Original audio with 2 channel
plt.figure(figsize=(12, 4))
plt.plot(wave_audio)
```

```
Out[26]:

[<matplotlib.lines.Line2D at 0x231a2fc5670>,
 <matplotlib.lines.Line2D at 0x231a2fc56d0>]
```




**Extract Features**

The next step is to extract the features we will need to train our model. To do this, we are going to create a visual representation of each of the audio samples which will allow us to identify features for classification, using the same techniques used to classify images with high accuracy. Spectrograms are a useful technique for visualising the spectrum of frequencies of a sound and how they vary during a very short period of time. we will be using a similar technique known as **Mel-Frequency Cepstral Coefficients (MFCC)** . The main difference is that a spectrogram uses a linear spaced frequency scale (so, each frequency bin is spaced an equal number of Hertz apart), whereas an MFCC uses a quasi-logarithmic spaced frequency scale, which is more similar to how the human auditory system processes sounds.

**Time Domain Representation:-** comparing amplitude over time.

**Spectrogram:-** showing the energy in different frequency bands changing over time.

## 3.1 Mel Frequency Cepstral Coefficient (MFCC)

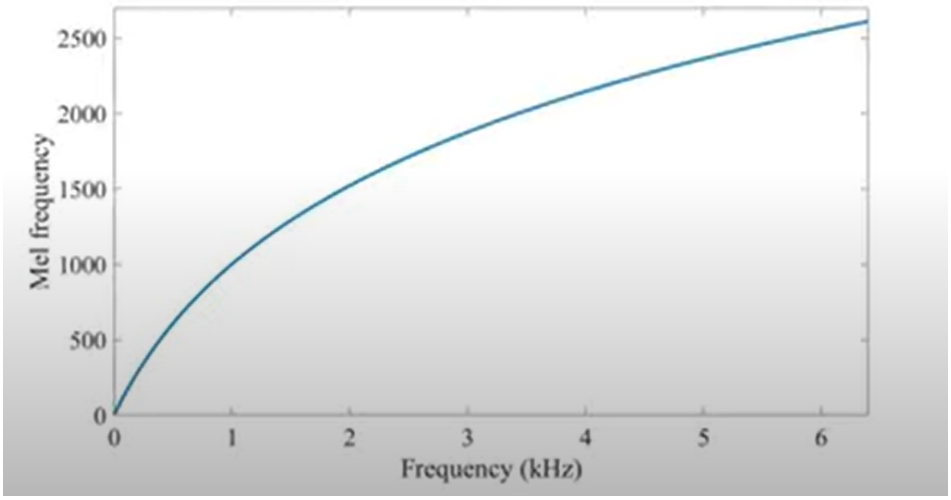
To understand MFCC we just go Deeper inside it. So we Divide MFCC in to parts to understand easily first part is Mel frequency and another one is Cepstral Coefficient.

### 3.1.1 Mel Frequency/Spectrogram

Here we start with Mel Frequency and how it will created, So we Know that there are three types of Audio features.

- Time-frequency representation.
- Perceptually-relevant amplitude representation.
- Perceptually-relevant frequency representation.

This is what Mel-spectrogram all about and they check out all these conditions. Now Mel-spectrogram is concept with two words one is spectrogram and you should know this and the other one is Mel so, what is this thing? what's a Mel? This concept is connected with Mel-scale and Mel-scale is perceptually relevant for pitch and here we have nice representation of this,



On the X-axis we have frequency expressed in hertz and on the Y-axis, we have Melfrequency expressed in Mel and the blue curve is just a mapping between hertz and Mel. You can see it follows a logarithmic curve and as we know that mouth scale is equal distances on the scale have same perceptual distance. Let's see it by an example: Let's say we have couple of pairs of notes so, the first pair is 500 mel's and 510 mel's and second one is 1000 mel's and 1010 mel's. now the perceptual distance between these two notes in the two pairs is same so, the pitch distance is going to be same and that's obviously not true for hertz as we know. Now the one thing that we know like the researchers did with mouth scale is that they standardized it so, that at 1000 hertz we have 1000 mel's now you may be wondering but how did we arrive at such a perceptual scale well we arrived at in an empirical way doing psychological experiments. Now let's take a look at how we can move from the frequency expressed in hertz and frequency expressed in mel's and following is like an empirical formula is found by trial-and-error experiments and the important thing apart from this is constants is like a logarithm relationship.

$$\mathbf{m = 2595.\ Log(1+f/500)}$$

Now we can take the inverse of this function with the help of this we are going to be able to move from mel's back to hertz.

$$f = 700(10^{m/2595} - 1)$$

Now we move forward. So, now you should have a basic idea about mel scale and why it is important in in terms of why we perceive pitch in this case. But how we relate it with spectrogram so, for this we have three recipes to extract Mel spectrogram.

1. Extract Short Time Fourier Transform
2. Convert amplitude to Decibels
3. Convert frequencies to Mel Scale

These two parts are well known for us the new one is the 3rd part that is to converting frequencies to mel scale so, this is of the whole point of the mel spectrogram. So, we take a spectrogram and then we convert the frequencies in that spectrogram to the mel frequency representation. For this conversion we have Three steps like

- **Choose number of mel bands**

Let's start with the first step That's to choose number of mel bands. Mel bands returns a Mel-Frequency spectrum which is computed by bundling subset of FFT bin magnitudes according to their relationship to the Mel scale (The "Mel" derives from the word "melody").

- **Construct mel filter banks**

In this step we are actually understand what is mel bands? and how we actually construct a filter bank? this kind of like a multi-step process. So, to build the filter bank we take a lowest and highest frequencies like in the range or in the short time Fourier transformation that we just extracted and convert the lowest and highest frequencies to a mel representation so, how do we do this we use following little formula that we already know when we are talking about the mouth scale.

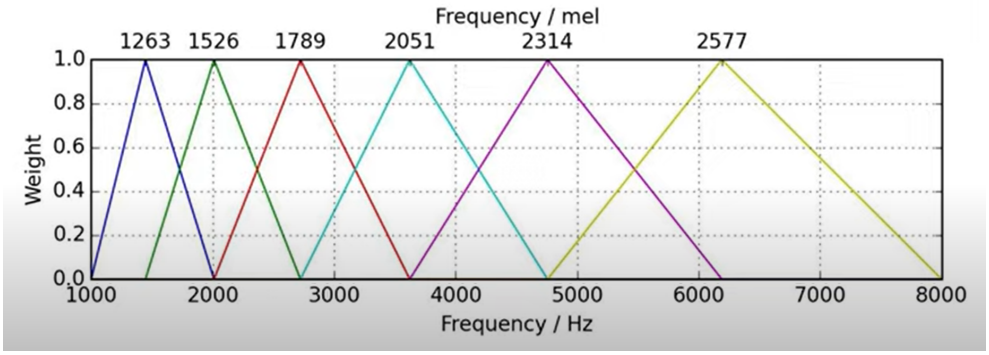
$$\mathbf{m = 2595.\ Log(1+f/500)}$$

We have lowest and highest frequencies that are converted into the mel's. So, the next step is now given like we've already chosen the number of Mel bands that we want to use for our mel spectrogram. Take the frequency range from the Lowest mel and the highest mel and we have to create as

many points equally spaced points as the number of mel bands that we want to use. In other words, if we take an example as lets, we take six mel bands we want to create points so, we take points like they are equally spaced in the frequency from lowest to highest. Next step is basically just converting these points back to hertz by using this function here which we are talking about in the mel scale,

$$f = 700(10^{m/2595} - 1)$$

Now if you are wondering about this mel points and what are they so, they are frequency of the different mel bands that we ae talking about here and we are see that from visualization in couple of moments. Next step is to converting the centre frequency points for our mel bands back to hertz and what we want do is we want to round these points to the nearest frequency bin. So, why do we need to do that? well it's because we are dealing with discrete signals and that means we don't have like infinite perfect resolutions on the frequency and the resolution is somewhat like constrained by the frame size of the short time Fourier transform so, what that basically means is that when we'll get back to the centre frequency points and bin them or round them to the nearest frequency bin and finally we have the last step which is probably the most important step which is like creating triangular filters and these triangular filters are basically building blocks of a mel filter bank and they are connected with this idea of the different mel bands. So, now we trying to visualize all these things to understand it in better way its way easier than it looks like.




Here, on the x-axis we have frequency in bottom is expressed terms of hertz and the top in here is expressed in terms of mel's and here on the y-axis we have weight and the weight is between one and zero. Well, why do we have weight there? Because, these are filters. Basically, what they do is just tends to filter sound and we have weight equal to one that time you are not touching that signal but the weight below one what here are happening is dumping down the signal because we are scaling it with value that below one.

So, now we can see here we have an example here like six bands here and indeed we have overall six points and these are the centre of frequencies for our six mel bands lets take an example as the second point at 2000 hertz which is 1526 mel's what you can notice is that here the difference between all the centre of points for the mel bands is the same when it's calculated in mel. So, we basically have always the same difference between to subsequent like centre frequency points but this is not true in the case of hertz and that's the whole point because when we go like higher frequencies, we just have to spread out frequencies to have the same perceptual distance in terms of frequency distance. Now focus on to constructing a single triangular filter for one mel band. Let's take number two point on the mel frequency. Their

lower end and higher end are taken by the respectively centre of the previous mel band and the centre frequency of the subsequent mel band and weight for the lower end and higher end is equal to zero so the signal is completely muted is put zero is same thing for the frequency is above the higher end. Now we trace a line between the lower and higher end and we connect that with the centre of frequency where the width is equal to one and so if we do that, we actually build a triangular filter. If you do the same thing again for the third, fourth and sixth we just come with whole mel filter banks and these are the triangular filters. Now you are understanding this what a mel filter bank is and the whole purpose of this. To apply this to a normal spectrogram and then we can filter out different frequencies and convert them into normal frequencies in hertz to frequencies expressed in mel's. but here we have just a visualization of a mel filter bank but in digital signal processing as well as in machine learning we don't do operation with visualization we actually use maths and linear algebra like most of the time. What this means is that we can represent this mel filter using metrics or a two-dimensional array and the shape of the metrics is going to like this,

**(# bands, framesize / 2 + 1)**

So, the numbers of rows that we're going to have in a mel filter bank is equal to the number of bands that we choose in our example it's going to be equal to rows and then the number of columns that we have it's equal to framesize divided by two plus one and so this is the matrix that we'll be using for representing a mel filter bank.

Now with the help of all of this steps we go back to the final step in frequency to mel scale for our spectrogram, in other words we are actually moving from the spectrogram to the mel spectrogram and how do going to

do that we have to apply the mel filter bank to the spectrogram so how we are going to do that. Here, linear algebra is going to help us a lot. So, we got the shape of the mel filter bank matrix from previous,

**(# bands, framesize / 2 + 1)**

And we have another matrix which is associated with spectrogram

**(framesize / 2 + 1, # frames)**

So, the shape of the matrix is given by this numbers here in the framesize. The number of rows that we have in the spectrogram is equal to the ubiquitous frequency plus one and the number of columns is given by the number of frames. The whole point is to apply the mel filter bank to the spectrogram as we are familiar with the linear algebra so that we noticed something interesting about these two matrices and that's the number of columns in the mel filter bank matrix are equal to the rows of the matrix which are represents the spectrogram when this happen, we can do like to apply matrix multiplication so the result of this multiplication is the mel spectrogram.

**(# bands, # frames)**

The mel spectrogram is given by matrix and its shape is given by number of bands so as many rows as we choose the number of bands and the number of columns that we have is equal to the number frames of the original spectrogram. This is basically a mel spectrum which we are going to use in our project.

### 3.1.2 Cepstral Coefficient/Cepstrum

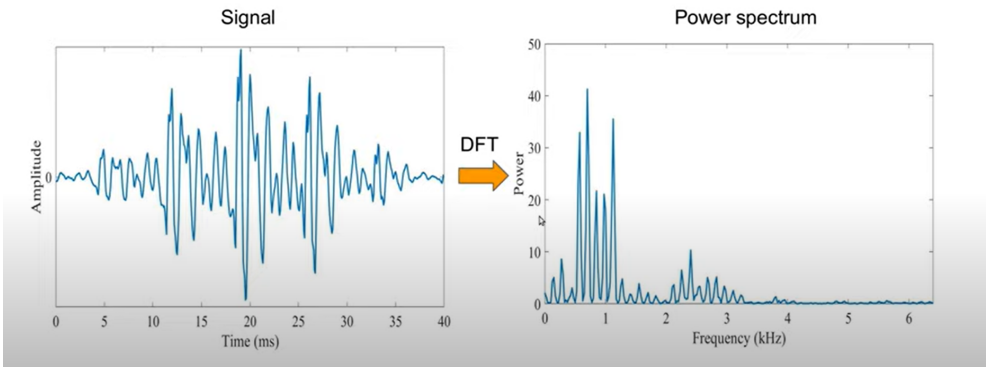
Now we understand about the Cepstrum and how it will look like. So, Cepstrum is provided by this formula

$$C(x(t)) = F^{-1}[log(F[x(t)])] \tag{3.1}$$

Where, C=Cepstrum, x(t)=Normal signal in Time domain just like a wave form, F= Discrete Fourier transformation F[x(t)] = Spectrum and
log[F[x(t)] =logarithmic spectrum

so, we come up with the spectrum and move from the time domain to the frequency domain. Now next step that we want to do is apply the logarithm to the spectrum and in this way, we get the log amplitude spectrum in other words we are applying the logarithm on the amplitude of the spectrum now we take the key step to get the spectrum which is basically applying an inverse Fourier transform to a log amplitude spectrum and when we do that, we came with a Cepstrum.

If you think about it what we actually doing as we are Taking a spectrum specifically a log amplitude spectrum and then we are calculating a spectrum of a spectrum so, that's because we are applying inverse Fourier transformation at this point. So, we are trying to visualize this concept because this going to help us understand what's going on here so, we will try to visualize all those different steps that we saw in the mathematical formalization.

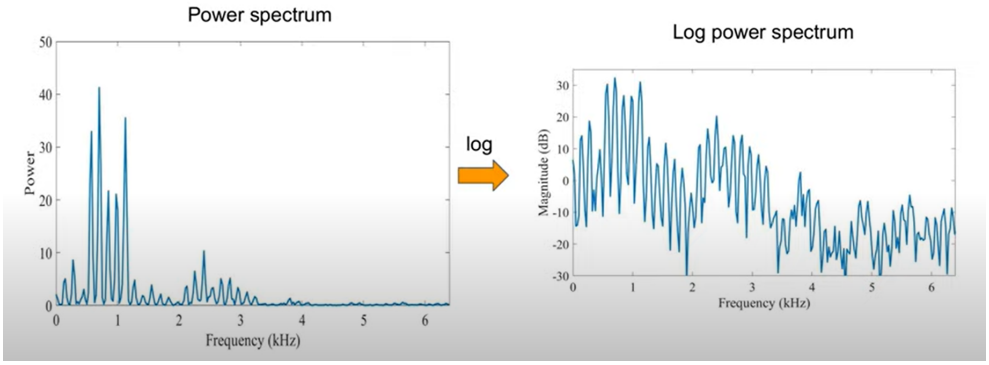



So, we start with a normal waveform we are in the time domain we have very short amount of sound just like 40 milliseconds for example here and then what we want to do here as a first step is taking the **Discrete Fourier Transform** and what we get out of that is our usual power spectrum where on the x-axis we have frequency and on y-axis we have power so, we seen this like time and again time and during this series basically here is moving like from the time domain to the frequency domain and the values that we have for each frequency tells us how much frequency component is present in the original signal in the original waveform so, this is like the first step.

Now the other step is of applying a logarithm to a power spectrum and so, here we do is the simple transformation so, we take all the amplitude and we apply a logarithm so, that we get decibels on the y-axis as the unit of reference and on the x-axis still we have frequency because the transformation was only like on the y-axis only. Now we have log power spectrum.

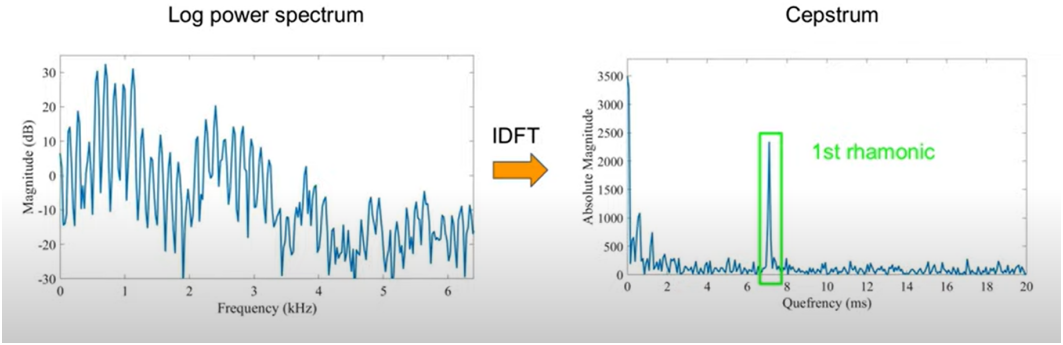
If we see Log power spectrum

1. This is a continues signal
2. it has periodic structure and this periodic structure present because the log power spectrum has some harmonic components or the original signals has some harmonic components that become periodic in the spectrum.




Basically when we have some signal even if its just like a spectrum that has some periodicities so, what we can do is we apply a transformation called Fourier transform to understand like the different components and try to find which frequencies present in the signal so, in other words what we can do here we treat the log power spectrum as a signal at a time domain signal and we can apply Fourier transformation and specifically we apply inverse Fourier transformation and what we will get is a spectrum of this signal which a spectrum in other words it is the spectrum of a spectrum which is the Cepstrum.

Here we are going to apply **Inverse Discrete Fourier Transformation** and we get the Cepstrum which is the spectrum of a spectrum.



Now the cool thing or the things that we should think about is what do we actually have on the x-axis because now we have the spectrum of a spectrum it means if we are in the time domain and we take a Fourier transform then we move in the frequency domain so, on the x-axis we will have frequencies obviously but if you start from a signal that has frequencies on the x-axis what do we actually get on the x-axis of the transformation and the answer is that we get some sort of pseudo frequency axis and this pseudo frequency axis is turned by the researchers by the **Quefrency** and the unit of reference here is millisecond or seconds.

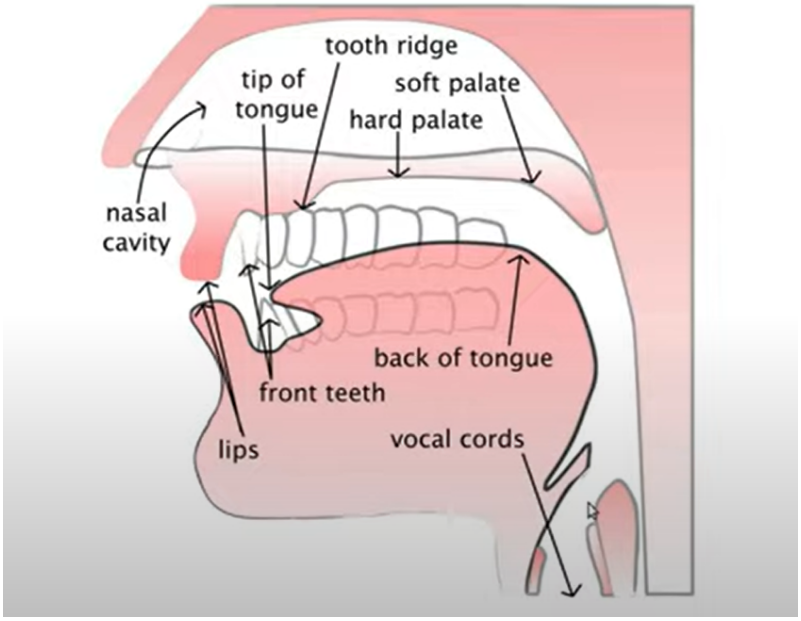
**why we are talking about quefrency and Cepstrum? and why this workplace make sense?**

the reason is because we are starting from the time domain originally with the waveform then we go to the frequency domain with the initial discrete Fourier transformation and now we apply inverse discrete Fourier transformation and we go back to somewhat resembles the frequency domain but it's not a frequency domain and so, they just like decided to take the opposite that so, it's not frequency it's quefrency and this is not a spectrum this is Cepstrum. So, here we have the intuition.what are all this values here, so, in the Cepstrum visualization we have certain pics and here we have a very high pick so, what do this represent doing? so, basically this represents how present these quefrencies in the log power spectrum. So, here we have the huge pick and that is the 1st rhamonic. This is equivalent to harmonic but this is rhamonic that this is the quefrency which is associated with the fundamental frequency of the original signal of the original waveform and indeed one way of using a cepstrum is one of the application for pitch detection because you take the log power spectrum and then you take the cepstrum and the pick that you're goanna like this 1st rhamonic and you can use that

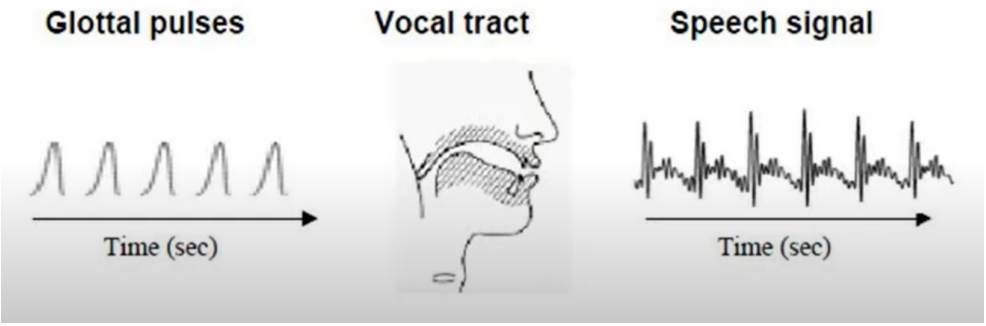
to then move back to the frequency domain and then understand where you have the fundamental frequency in the original signal. now you should ask me why this peak is? so, this peak shows the harmonic structure of the original signal that gets a periodic way here in the log power spectrum so, this like the key idea there. So, now you should understand the maths behind the cepstrum and understood how the cepstrum is looks like.

**what is missing here is to understanding why it is so, important? Why we should bother of taking inverse Fourier transformation of the log power spectrum?**

So, for understanding purpose that we have to take little detour into how speech work into speech processing. So, I want to contextualize cepstrum within speech so, the first thing that we need to understand how we produce speech and key element to understanding and how the human produce speech in the vocal tract.



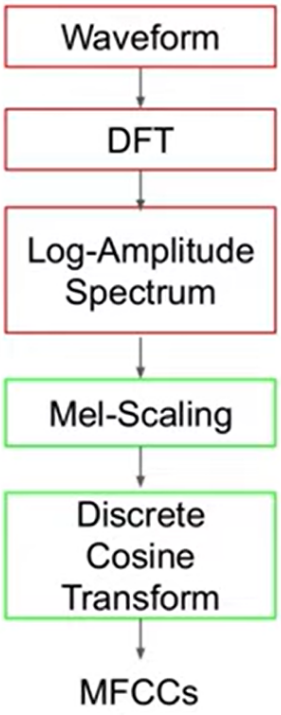
As we know the vocal tract is very complex system that has like multiple elements so, it has like Tongue, Teeth, Nasal cavity and throat. The basic idea is that depending how you shape your vocal tract your goanna produces different sound different phonemes or like different vowels it really depends on how you put your tongue, how you stretch your throat or you contract it, but if you think about that in terms of digital signal processing we can think of the vocal tract as a filter in other words vocal tract act as a filter so, how do we actually generate or produce speech. Well, this is quite fascinating but we will give you simple instrumental them to understand the cepstrum fully. Basically, speech generation acts in a kind of pipeline form.




Initially we have a glottal pulse and this is like a noisy signal or high-pitch signal that gets generated by the vocal falls and that signal passes through the vocal tract and the vocal tract act as a filter on the glottal pulse and by filtering like the initial signal it creates the speech signal. Now the basic idea once again is that depending on how you should shape your vocal tract then you're going to have like a different speech signal starting this more or less in the same glottal pulse. Now the intuition here is that the glottal pulse carries information about pitch or high frequency kind of like information whereas the vocal cord or the frequency response provided by the vocal tract. This filter carries the information about the timer of the sound or the speech

and specifically when we talk about timer about speech is like the actual phonemes that you produce. The different consonants or the different vowels that you can produce. This is called the high-level idea.

The best way of how MFCC's work is by looking at how we can compute them and this is multi-step process many of the steps are shared by how we compute cepstrum and MFCC's.




Let's get started we begin with a simple waveform so, signal in the time domain as usual we apply the Discrete Fourier Transformation and we get a spectrum out of that and next step is to apply a logarithm to the amplitude so, that we get log spectrum and up until this point the process for getting cepstrum and Mel-Frequency Cepstral Coefficients is actually same but here we have the first divergence so, what we do next is applying Mel scaling and

this means that we take the log spectrum and we apply Mel filter banks which are the triangular filters like this. At the end of this step, we have now a Mel spectrum. Now enter into the final step for getting MFCC instead of applying the equivalent if an Inverse Discrete Fourier Transform for the Cepstrum and in this case that one transformation is the discrete cosine transform now I'm not going to get into the details why we're using the discrete cosine transform instead of the Inverse Fourier Transform but we apply cosine transform is that we get number of coefficients a number of values or MFCC's which are the ones that we are interested in. This is complete Mel Frequency Cepstral Coefficient(MFCC)

### 3.1.3 MFCC Extract Features

For extract feature we have to import some libraries called as Librosa, Librosa.Display, IPython.display, Matplotlib and Numpy. After installing this file, we need an audio file for extract feature. Going back to coding, when we have to use Librosa feature we called it as librosa.feature.mfcc. To run this feature it requires an Librosa audio data and sample rate MFCC feature convert all audio signals into array of dimension.

```
In [27]:
#mfcc for single file or audio
#mfcc try to convert the audio into some kind of features based on the frequency and tim
#which will help us to do the classification
#n_mfcc=number of MFCCs to return
#this converts all audio data / signal converted into array of dimension (40,173)
mfccs = librosa.feature.mfcc(y=librosa_audio_data , sr=librosa_sample_rate,n_mfcc=40)
print(mfccs.shape)
```

```
(40, 173)
```

This is nothing but a pattern that have been extracted based on the frequency and time characteristics and this will uniquely able to identify that particular audio signal like in which class it will actually belong, But this is only for one file we have to run it for more than 8000 files that we have actually seen in our Urbansound8K there are so many folders and so many files. We have to apply all these particular things for all these files.

In [35]:

```
### Extracting MFCC's for every audio file
#The OS module in Python provides functions for interacting with the operating system
import pandas as pd
import os
import librosa

audio_dataset_path=('C:/Users/Srushti Kadam/Contacts/UrbanSound8K/audio')
metadata=pd.read_csv('C:/Users/Srushti Kadam/Contacts/UrbanSound8K/metadata/UrbanSound8K
metadata.head()
```

Out[35]:

| | slice_file_name | fsID | start | end | salience | fold | classID | class |
|---|---|---|---|---|---|---|---|---|
| 0 | 100032-3-0-0.wav | 100032 | 0.0 | 0.317551 | 1 | 5 | 3 | dog_bark |
| 1 | 100263-2-0-117.wav | 100263 | 58.5 | 62.500000 | 1 | 5 | 2 | children_playing |
| 2 | 100263-2-0-121.wav | 100263 | 60.5 | 64.500000 | 1 | 5 | 2 | children_playing |
| 3 | 100263-2-0-126.wav | 100263 | 63.0 | 67.000000 | 1 | 5 | 2 | children_playing |
| 4 | 100263-2-0-137.wav | 100263 | 68.5 | 72.500000 | 1 | 5 | 2 | children_playing |

Here we create Feature Extractor. Finally to find out scaled feature we will be doing a mean on the transpose of this particular value that we are actually getting.

In [36]:

```
#for 1 audio file
def features_extractor(file_name):
    audio,sample_rate= librosa.load(file_name, res_type='kaiser_fast')
    #If you need to reduce load time, you can pass in an optional parameter
    #to the load function, 'res_type', short for resample type(default is 'kaiser_best')
    mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate,n_mfcc=40)
    #by using librosa.feature.mfcc function we are extracting features and store it into
    print(mfccs_features.shape)
    mfccs_scaled_features = np.mean(mfccs_features.T,axis=0)
    #we have 40 rows and 173 column,for each row we have 173 column
    #instead of considering all 173 values we can consider their mean
    print(mfccs_scaled_features.shape)
    return mfccs_scaled_features
```

Above code is just for 1 audio file now in order to do it for all the audio file we have to iterate through this particular csv file. So, we take numpy library and tqdm library to see the progress of it. Now we create a list which called extracted feature and iterate with the help of metadata.iterrows. with the help of os.path we just taking that perticular path.

In [37]:

```
import numpy as np
from tqdm import tqdm
#now we iterate through every audio file and extract features
#using mel-frequency cepstral coefficients
extracted_features=[]
for index_num,row in tqdm(metadata.iterrows()):
    file_name = os.path.join(os.path.abspath(audio_dataset_path),'fold'+str(row["fold"])
    final_class_lables=row["class"]
    data=features_extractor(file_name)
    extracted_features.append([data,final_class_lables])
```

```
4870it [15:05,  7.86it/s]

(40, 173)
(40,)
(40, 173)
(40,)

4872it [15:05,  8.11it/s]

(40, 173)
(40,)
(40, 173)
(40,)

4874it [15:05,  8.18it/s]

(40, 173)
(40,)
(40, 173)
(40,)
```

Now what we are going to do here is we are converting the entire list into a data frame so, in order to convert it we are just using **pd.dataframe(extracted feature, column=[feature,class])**. Once we execute this we get,

In [46]:

```
#covering extracted_features to pandas dataframe
import pandas as pd
extracted_features_df=pd.DataFrame(extracted_features,columns=['feature', 'class'])
extracted_features_df.head()
```

Out[46]:

| | feature | class |
|---|---|---|
| 0 | [-217.35526, 70.22338, -130.38527, -53.282898,... | dog_bark |
| 1 | [-424.09818, 109.34077, -52.919525, 60.86475, ... | children_playing |
| 2 | [-458.79114, 121.38419, -46.520657, 52.00812, ... | children_playing |
| 3 | [-413.89984, 101.66373, -35.42945, 53.036354, ... | children_playing |
| 4 | [-446.60352, 113.68541, -52.402206, 60.302044,... | children_playing |

This particular information contain features of particular class. So, everything we can see here.

Basically, we have to split the entire data set into Independent and dependent features, so, it is very simple we are taking the features and we are converting into two lists and we're converting into an array similarly for y, y is nothing but this is my class. **extracted feature data-frame** of class to list this is pretty much simple.

In [48]:

```
#split the dataset into independent and dependent dataset
x=np.array(extracted_features_df['feature'].tolist())
y=np.array(extracted_features_df['class'].tolist())
```

In [49]:

```
x.shape
```

Out[49]:

```
(8732, 40)
```

In [50]:

```
y.shape
```

Out[50]:

```
(8732,)
```

If we go and see Our X.Shape then we found that how many rows and columns are there in X. Two things that we actually going to do we don't required **pd.get dummies** because we are thinking of doing a pd.get dummies to my class feature after going for testing part we just have to decode the features to know the audio is actually belong to which class so, instead of this what we can do is that we directly use **label encoding** from the **Tenserflow and Sklearn** itself so, once it executes and if we check the Y.Shape then we got how many rows and columns are there.

In [51]:

```
#get_dummies used to convert categorical data into numeric data
#it will create 10 dummy class assign zero or one as per the presencenase of audio
#y=np.array(pd.get_dummies(y))
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
labelencoder=LabelEncoder()
y=to_categorical(labelencoder.fit_transform(y))
```

In [52]:

```
y.shape
```

Out[52]:

```
(8732, 10)
```

In [53]:

```
#train test split
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

Now what we are going to do is that we are going to do a train test split and inside the train test split we are going to give our x-feature and y-features test size and random state is equal to zero.

In [54]:

```
x_train.shape
```

Out[54]:

```
(6985, 40)
```

In [55]:

```
x_test.shape
```

Out[55]:

```
(1747, 40)
```

In [56]:

```
y_train.shape
```

Out[56]:

```
(6985, 10)
```

In [57]:

```
y_test.shape
```

Out[57]:

```
(1747, 10)
```

In [58]:

```
x_train
```

Out[58]:

```
array([[-1.31104706e+02,  1.12505905e+02, -2.25746956e+01, ...,
         3.24665260e+00, -1.36902368e+00,  2.75575471e+00],
       [-1.36703424e+01,  9.10850830e+01, -7.79273319e+00, ...,
        -3.25305033e+00, -5.27745247e+00, -1.55697143e+00],
       [-4.98715439e+01,  2.65352994e-01, -2.05009365e+01, ...,
         2.85459447e+00, -1.60920465e+00,  3.52480578e+00],
       ...,
       [-4.27012360e+02,  9.26230469e+01,  3.12939739e+00, ...,
         7.42641389e-01,  7.33490825e-01,  7.11009085e-01],
       [-1.45754608e+02,  1.36265778e+02, -3.35155220e+01, ...,
         1.46811950e+00, -2.00917006e+00, -8.82181883e-01],
       [-4.21031342e+02,  2.10654541e+02,  3.49066067e+00, ...,
        -5.38886690e+00, -3.37136054e+00, -1.56651139e+00]], dtype=float3
```

**So, basically, our Data pre-processing part is completed here. . . ..**

# Chapter 4

# Model Creation & Testing ANN Model

As did in part 2 we extracted features from the audio and now divided that into dependent and independent features. We have extracted features from the audio sample and splitter in the train and test set. Now we will implement an ANN model using Keras sequential API and TensorFlow. Specifically, the version of TensorFlow that we are taking is greater than 2.0. It is not specific to use this version but this is the latest version so we are going to use this version for this project. First, of all we are going to import some libraries for like sequential, dense. Dropout and flatten layer so we will able to create ANN and try to see what is accuracy that we are probably getting it.N model using Keras sequential API. Specifically, the version of TensorFlow that we are taking is greater than 2.0. It is not specific to use this version but this is the latest version so we are going to use this version for this project. First, of all we are going to import some libraries for like sequential, dense. Dropout and flatten layer so we will able to create ANN and try to see what is accuracy

that we are probably getting it.Then we import metrics from sklearn and, we are using an Adam optimizer.

```
In [65]:
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout,Activation,Flatten
from tensorflow.keras.optimizers import Adam
from sklearn import metrics
```

After this we want to know about the number of classes and number of labels. Inside our y_test we have 10 classes so this is value that we need to get. So, we are use this particular class is over y.shape[1] this will actually tell us the number of classes.

```
In [66]:
### No of classes
num_labels=y.shape[1]
```

Model Creation: The number of classes is 10, which is our output shape(number of classes), and we will create ANN with 3 dense layers and architecture is explained below.

**1)** The first layer has 100 neurons. Input shape is 40 according to the number of features with activation function as rectified linear unit (ReLU) , and to avoid any overfitting, we'll use the Dropout layer at a rate of 0.5. The input shape will be (40,) because in our training data set, we had 40 features so we usually give the input shape with the respect to the number of features.

**2)** The second layer has 200 neurons with activation function as Relu and the drop out at a rate of 0.5.

**3)** The third layer again has 100 neurons with activation as Relu and the drop out at a rate of 0.5. Final layer basically has SoftMax because it is multiclass classification problem

In [67]:

```
model=Sequential()
###first layer
model.add(Dense(100,input_shape=(40,)))
model.add(Activation('relu'))
model.add(Dropout(0.5))
###second layer
model.add(Dense(200))
model.add(Activation('relu'))
model.add(Dropout(0.5))
###third layer
model.add(Dense(100))
model.add(Activation('relu'))
model.add(Dropout(0.5))

###final layer
model.add(Dense(num_labels))
model.add(Activation('softmax'))
```

We can observe the model summary using the summary function.

In [68]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 100)               4100

 activation (Activation)     (None, 100)               0

 dropout (Dropout)           (None, 100)               0

 dense_1 (Dense)             (None, 200)               20200

 activation_1 (Activation)   (None, 200)               0

 dropout_1 (Dropout)         (None, 200)               0

 dense_2 (Dense)             (None, 100)               20100

 activation_2 (Activation)   (None, 100)               0

 dropout_2 (Dropout)         (None, 100)               0

 dense_3 (Dense)             (None, 10)                1010

 activation_3 (Activation)   (None, 10)                0

=================================================================
Total params: 45,410
Trainable params: 45,410
Non-trainable params: 0
_________________________________________________________________
```

**Compile the Model**:To compile the model we need to define loss function which is categorical cross-entropy, accuracy metrics which is accuracy score, and an optimizer which is Adam.

In [69]:

```
model.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')
```

**Train the Model** We will train the model and save the model in HDF5 format. We will train a model for 100 epochs and batch size as 32. We'll use callback, which is a checkpoint to know how much time it took to train over data.

In [87]:

```
## Trianing my model
from tensorflow.keras.callbacks import ModelCheckpoint
from datetime import datetime

num_epochs = 100
num_batch_size = 32

checkpointer = ModelCheckpoint(filepath='saved_models/audio_classification.hdf5',
                               verbose=1, save_best_only=True)
start = datetime.now()

model.fit(x_train, y_train, batch_size=num_batch_size, epochs=num_epochs, validation_dat


duration = datetime.now() - start
print("Training completed in time: ", duration)
```

```
curacy: 0.7342
Epoch 12: val_loss improved from 0.63992 to 0.63101, saving model to s
aved_models\audio_classification.hdf5
219/219 [==============================] - 1s 5ms/step - loss: 0.8076
- accuracy: 0.7349 - val_loss: 0.6310 - val_accuracy: 0.7979
Epoch 13/100
218/219 [============================>.] - ETA: 0s - loss: 0.8061 - ac
curacy: 0.7390
Epoch 13: val_loss did not improve from 0.63101
219/219 [==============================] - 1s 5ms/step - loss: 0.8061
- accuracy: 0.7389 - val_loss: 0.6710 - val_accuracy: 0.7894
Epoch 14/100
212/219 [============================>.] - ETA: 0s - loss: 0.8317 - ac
curacy: 0.7376
Epoch 14: val_loss did not improve from 0.63101
219/219 [==============================] - 1s 5ms/step - loss: 0.8310
- accuracy: 0.7373 - val_loss: 0.6690 - val_accuracy: 0.7831
Epoch 15/100
217/219 [============================>.] - ETA: 0s - loss: 0.8077 - ac
curacy: 0.7355
```

**Check the Test Accuracy** Now we will evaluate the model on test data. we got near about 80 percent accuracy on the training dataset and 80 percent on test data.

In [88]:

```
test_accuracy=model.evaluate(x_test,y_test,verbose=0)
print(test_accuracy[1])
```

```
0.8013737797737122
```

After getting the validation accuracy we are going to test 1 data respect to this like how we get an specific data an all and try to have a look into it. So, by selecting one particular audio file we just apply feature extractor and prediction feature to that file after running this step we got an array of something between 0 and 9 and basically, array of that particular data which is nothing but our class.

In [89]:

```
filename="C:/Users/Srushti Kadam/Contacts/UrbanSound8K/audio_test.wav"
predication_feature=features_extractor(filename)
predication_feature=predication_feature.reshape(1,-1)
model.predict(predication_feature)
```

```
(40, 97)
(40,)
1/1 [==============================] - 0s 219ms/step
```

Out[89]:

```
array([[0.00697668, 0.02558697, 0.05141434, 0.15690571, 0.23689039,
        0.03264529, 0.0128081 , 0.0127095 , 0.2857614 , 0.17830156]],
      dtype=float32)
```

**Testing ANN Model** In previous chapter we created an ANN model and we train that specific model for the audio classification and there we just got an 80 percent accuracy. Here we are going to see how to test the new audio data. For the testing of data basically it includes 3 steps

1. Pre-process the new audio data: - where we need to extract features from that specific data itself with help of MFCC

2. Predict the classes: - predict with the help of model in order to predict the classes

3. Inverse transform your predicted label to get the class name

Initially we are trying to use get dummies to do the label encoding with respect to the output feature but whenever we really want to work with the test data, we also need to inverse that transform from that label to the class label that is why we use label encoder. When we are using label encoder after that we just have to put it y= "to_categorical" That basically means initially, we have y is classes of 'dog bark', 'Children playing', ..... , 'car horn' like this. So, it will first of all convert it into class labels as 0,1,2,3,4,5,6,7,8,9 like that, and total numbers of class label is 10. So that we getting the dummy variables with respect to number of features. So, same encoder we are using here which we are basically using in previous chapter. We have created a folder wherein we have taken some of the test data like dog bark,drilling gunshot siren. We will try to test for this particular data and we will see that how the accuracy is. We have taken the dog bark audio . Firstly we are going to use librosa to load that particular file . with the help of function called as features_extractor which we had created . In that we are loading a file and getting the features using mfccs then we are doing the mean and returning features. For one audio file we need to reshape as (1,-1) that is one row with 40 features of values. Using model.predict_classes

we will be getting the predicted label that is 0,1,2,. . . .. After getting a class label we have to do inverse transform to get the class name (using labelencoder.inverse_transform)

```
filename="C:/Users/Srushti Kadam/Contacts/UrbanSound8K/audio/fold1/7383-3-0-0.wav"
audio, sample_rate = librosa.load(filename, res_type='kaiser_fast')
mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
mfccs_scaled_features = np.mean(mfccs_features.T,axis=0)

print(mfccs_scaled_features)
mfccs_scaled_features=mfccs_scaled_features.reshape(1,-1)
print(mfccs_scaled_features)
print(mfccs_scaled_features.shape)
predicted_label_prob=model.predict(mfccs_scaled_features)
predicted_index=np.argmax(predicted_label_prob,axis=1)
print(predicted_index)
prediction_encoded_class =[labelencoder.inverse_transform([i]) for i in predicted_index]
#prediction_class=[s[i] for i in predicted_index]
#print(prediction_class)
print(prediction_encoded_class)
```

```
[-3.9742188e+02  1.4218434e+02  1.0239103e+01 -9.6230564e+00
 -8.1905851e+00  7.6672630e+00 -1.0299530e+01 -7.1226473e+00
 -3.5969064e+00 -5.6251779e+00 -2.4014688e+00 -3.5109861e+00
  5.2140422e+00  9.1520948e+00  9.7396631e+00  1.2672479e+01
  2.4935956e+00 -1.9596183e-01  2.0568910e+00  1.2565480e-01
  2.3887675e-01 -2.2244754e+00 -4.1830082e+00 -8.7841505e-01
  5.5512708e-01  1.4530714e+00  1.5828406e+00  2.5280831e+00
  4.4751968e+00  1.8469352e+00  8.8243917e-02 -2.5079671e-02
  1.9693788e+00  1.1326451e+00  1.8206301e-01 -9.0641987e-01
 -2.5448070e+00 -1.7134562e+00 -1.3600037e+00  3.9560735e-01]
[[-3.9742188e+02  1.4218434e+02  1.0239103e+01 -9.6230564e+00
  -8.1905851e+00  7.6672630e+00 -1.0299530e+01 -7.1226473e+00
  -3.5969064e+00 -5.6251779e+00 -2.4014688e+00 -3.5109861e+00
   5.2140422e+00  9.1520948e+00  9.7396631e+00  1.2672479e+01
   2.4935956e+00 -1.9596183e-01  2.0568910e+00  1.2565480e-01
   2.3887675e-01 -2.2244754e+00 -4.1830082e+00 -8.7841505e-01
   5.5512708e-01  1.4530714e+00  1.5828406e+00  2.5280831e+00
   4.4751968e+00  1.8469352e+00  8.8243917e-02 -2.5079671e-02
   1.9693788e+00  1.1326451e+00  1.8206301e-01 -9.0641987e-01
  -2.5448070e+00 -1.7134562e+00 -1.3600037e+00  3.9560735e-01]]
(1, 40)
1/1 [==============================] - 0s 40ms/step
[3]
[array(['dog_bark'], dtype='<U16')]
```

# Conclusion

Audio classification using deep learning has shown promising results in recent years, offering a powerful tool for automated audio classification tasks. By training deep neural networks on large-scale audio datasets, deep learning techniques can learn complex patterns and features directly from raw audio data, leading to more accurate and efficient audio classification systems.

The availability of large-scale audio datasets and the advancements in deep learning techniques have led to the development of various deep learning-based audio classification systems for different applications, such as speech recognition, music recommendation, audio search and retrieval, and environmental monitoring. These systems offer many benefits over traditional audio classification methods, including higher accuracy, faster processing times, and the ability to handle large and diverse datasets.

However, there are still some challenges and limitations in audio classification using deep learning, such as the need for large amounts of labeled data, potential overfitting, and the difficulty of handling different types of noise and variations in audio signals. Further research is needed to address these challenges and to explore new architectures, optimization techniques, and training strategies for improving the performance and robustness of deep learning-based audio classification systems.

# Bibliography

[1] M. Andar, S.Boxwala and N.B.Limaye; *Cordial labelings of some wheel related graphs*, J.Combin.Math.Combin.Comput, 41 (2002), 203 - 208.

[2] J.Bolland, R.Laskar, C.Turner, G.Domke; *On Mod Sum Graphs*, Congresses Numerantium,70 (1990), 131 - 135.

■✠■✠■