**Industrial Oriented Mini Project Report**

On

**Audio Classification System**

Submitted in partial fulfillment of the requirements for the award of the Degree of

**Bachelor of Technology**

in

**COMPUTER SCIENCE AND ENGINEERING**

By

**K.GOUTHAM REDDY 16241A05E4**

**M.KHUSHAL REDDY 16241A05F3**

**M.SAI VARMA 16241A05F4**

**P.ARAVIND 17245A0534**

Under the Esteemed guidance of

**K.SWANTHANA**

**Assistant Professor**

****

**Department of Computer Science and Engineering**

##### **GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY**

##### **(Autonomous)**

**Bachupally, Kukatpally, Hyderabad- 500090 2018-2019**

##### D:\3-2 Wt\griet.jpg

##### **GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY**

##### **(Autonomous)**

##### **Department of Computer Science and Engineering**

##### **CERTIFICATE**

This is to certify that the Industrial mini project entitled **"Audio Classification System"** is submitted by **K. Goutham (16241A05E4), M.Khushal (16241A05F3), M.Sai Varma (16241A05F4), P.Aravind (17245A0534)** in partial fulfillment of the requirement for the award of the degree in **BACHELOR OF TECHNOLOGY** in Computer Science and Engineering during the academic year 2018-2019.

INTERNAL GUIDE HEAD OF THE DEPARTMENT

**K.Swanthana Dr. K. Madhavi Assistant Professor**

EXTERNAL EXAMINER

DECLARATION

I hereby declare that the industrial mini project entitled **"Research Publications Portal"** is the work done during the period from 10th December 2018 to 4th April 2019 and is submitted in the partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering from Gokaraju Rangaraju Institute of Engineering and Technology (Autonomous under Jawaharlal Nehru Technology University, Hyderabad). The results embodied in this project have not been submitted to any other University or Institution for the award of any degree or diploma.

**K.GOUTHAM REDDY 16241A05E4**

**M.KHUSHAL REDDY 16241A05F3**

**M.SAI VARMA 16241A05F4**

**P.ARAVIND 17245A0534**

ACKNOWLEDGEMENT

There are many people who helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all.

First of all we would like to express our deep gratitude towards our internal guide **K.Swanthana, Assistant Professor,** Department of CSE for her support in the completion of my dissertation. We wish to express our sincere thanks to **Dr.K.Madhavi, HOD, Department of CSE** and our mini project coordinator **G.Mallikarjuna Rao** for his external support. We also thank our principal **Dr.J.Praveen** for providing the facilities to complete the dissertation.

We would like to thank all our faculty and friends for their help and constructive criticism during the project period. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve our goals.

**K.GOUTHAM REDDY - 16241A05E4**

**M.KHUSHAL REDDY - 16241A05F3**

**M.SAI VARMA - 16241A05F4**

**P.ARAVIND - 17245A0534**

**ABSTRACT**

This project aims to classify the type of sound from the given audio file. This is done by extracting features from this audio like the frequency domain features(Amplitude of individual frequencies),time domain features,(root-mean-square error) Mel-frequency cepstral coefficients (MFCCs) , so that our algorithm (CNN) can work on these features and perform the task of classification.

Audio Classification can be used for audio scene understanding which in turn is important so that an artificial agent is able to understand and interact better with its environment.

**CONTENTS Page No**

Certificate i

Declaration ii

Acknowledgement iii

Abstract iv

Contents v

**1.** **Introduction** 1

1.1 Intelligent System 1

1.2 Project Overview 1

1.2.1 Existing System 2

1.2.2 Proposed System 2

**2. System Requirements** 3

2.1 Software Requirements 3

2.2 Hardware Requirements 3

**3.** **Technology** 4

3.1Python Tool 4

3.1.1 NumPy 4

3.1.2 Tensorflow 4

3.1.3 Librosa 5

3.1.4 Tkinter 5

3.2 Convolutional Neural Networks(CNN) 5

3.2.1 Layers in CNN 6

3.2.2 Convolutional Layer 6

3.2.3 Poling Layer 7

3.2.4 Max Poling 2D 7

3.2.5 Rectified Linear Unit (RELU) 8

3.2.6 Loss Layer 8

3.2.7 Flattering 8

3.2.8 Fully Connected Layers 10

3.2.9 Softmax Layer 11

3.2.10 Overview of entire CNN Working 11

**4. System design** 12

4.1 System Architecture 12

4.2 Data Flow Diagram 12

**5. Implementation** 13

5.1 Modules 13

5.1.1 Dataset preparation and Pre-processing Module 13

5.1.2 Feature Extraction and Model Building Module 14

5.1.3 Prediction Module 14

5.2 Code Snippets 15

**6. Testing** 26

6.1 System Testing 26

6.2 Types of Tests 26

**7. Screenshots** 30

7.1 Data Set 30

7.2 Graph related to Pre-processing 30

7.3 Model 31

7.4 Output 31

7.5 GUI 32

**8. Conclusion** 33

**9. Bibliography** 34

1. **INTRODUCTION**

**1.1 Intelligent System**

With the development of the electronic computer and the stored program computer the conditions for research in intelligent systems started. While exploiting the power of the computer systems, the curiosity of human, lead him to wonder, “Can a machine think and behave like humans do?”

Thus, the development of Intelligent System started with the intention of creating similar intelligence in machines that we find and regard high in humans.

After decades of hype, Artificial Intelligence (AI) has arrived. It started to take over the work which cannot be done by human.

According to Wikipedia, “Artificial Intelligence is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and other animals. Computer science defines AI research as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals."

**1.2 Project Overview**

In this project we perform Classification and Recognition of an Audio signal using one of

the key techniques in Artificial Intelligence called as Convolutional Neural Networks.

The data which we have gathered is the audio file of different musical instruments.

* Here we classify the different types of audio samples.
* Recognize a specified sample.
* Provide the accuracy.

**1.2.1 Existing System**

Pre-processing of the audio signal in the existing system is generally done without amplifying the sound signal, which may not be as sounding.

Support Vector Machine (SVM) is being used to classification. As SVM uses CPU, it consumes lot of memory and time. Tradition SVM will not provide more than 90% accuracy on unstructured data.

**1.2.2 Proposed System**

Here we are implementing the classification and recognition using one of the techniques of Deep Learning in Machine Learning i.e., The Convolution Neural Network(CNN).

**2. SYSTEM REQUIREMENTS**

**2.1 Software Requirements**

Supported Operating System

* Windows 7(32 or 64 bit), Windows 8(32 or 64 bit), Windows 10(32 or 64 bit)

Supported Development Environment

* Python
* Tensorflow
* Anaconda
* Spyder
* Jupiter

**2.2 Hardware Requirements**

* Processor: 1.2 GHZ
* RAM: 8 GB
* ROM: 300 GB
* GPU: 2 GB

**3. TECHNOLOGY**

**3.1. Python Tool**

Python is considered as one of the widest used machine learning tool. It has many other features that attract the data science community. Being a data science tool, Python helps to explore the concepts of machine learning in the best way possible. Machine Learning is all about probability, mathematical optimization, and statistics, which are all made easy by Python.

There are lot of inbuilt machine Learning packages which will make machine learning Handy with python. What drives developers to Python is that it is easy to learn and code. It promotes an easy-to-understand syntax especially when compared to other data science languages, such as R and thereby leads to a shorter learning curve.

The reason for growing success of Python is the availability of data science libraries for aspiring candidates. These libraries have been upgraded continuously. The constraints that developers faced a year ago are now treated successfully with Python. Many libraries are available to perform data analysis, here’s an important one to start with

**3.1.1**  **NumPy**

Numpy is important to perform scientific computing with Python. It encompasses an assortment of high-level mathematical functions to operate on multi-dimensional arrays and matrices.

**3.1.2. TensorFlow**

A tensor is an array-like object, and, as you've seen, an array can hold your matrix, your vector, and really even a scalar. They are actually just number-crunching libraries, much like Numpy is. The difference is, however, a package like TensorFlow allows us to perform specific machine learning number-crunching operations like derivatives on huge matrices with large efficiency. We can also easily distribute this processing across our CPU cores, GPU cores, or even multiple devices like multiple GPUs.

**3.1.3. Librosa**

Librosa is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems.

**3.1.4. Tkinter**

It provides a robust and platform independent windowing toolkit that is available to us using the tkinter package. The tkinter package is a thin object-oriented layer on top of Tcl/Tk

**3.2. CONVOLUTIONAL NEURAL NETWORK (CNN)**

Neural network we used is Convolutional neural network to train the model. It is one of the most powerful supervised deep learning techniques.

The final structure of a CNN is actually very similar to Regular Neural Networks (RegularNets) where there are neurons with weights and biases. In addition, just like in RegularNets, we use a loss function (e.g. cross entropy or softmax) and an optimizer (e.g. adam optimizer) in CNNs.

In CNNs, there are Convolutional Layers, Pooling Layers, and Flatten Layers. CNNs are mainly used for image classification although you may find other application areas such as natural language processing, digital signal processing (Audio processing).

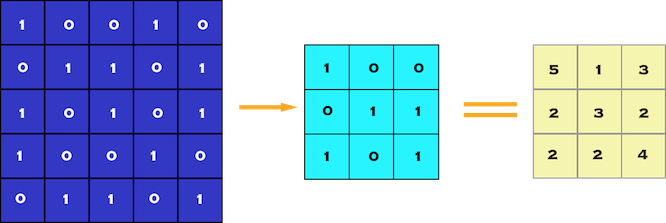
The main structural feature of Regular Nets is that all the neurons are connected to each other. For example, when we have images with 28 by 28 pixels with only grey-scaled, we will end up having 784 (28 x 28 x 1) neurons in a layer which seems manageable. However, most images have way more pixels and they are not grey-scaled. Therefore, assuming that we have a set of color images in 4K Ultra HD, we will have 26,542,080 (4096 x 2160 x 3) different neurons connected to each other in the first layer which is not really manageable. Therefore, we can say that Regular Nets are not scalable for image classification. However, especially when it comes to images, there seems to be little correlation or relation between two individual pixels unless they are close to each other. This leads to the idea of Convolutional Layers and Pooling Layers.

**3.2.1. LAYERS IN CNN**

We are capable of using many different layers in a convolutional neural network. However, convolution, pooling, ReLU layer, Loss layer and, fully connected layers are the some of the layers.

**3.2.2. CONVOLUTIONAL LAYER**

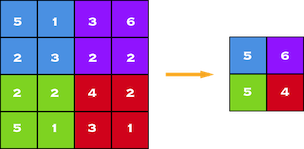
Convolutional layer is the very first layer where we extract features from the images in our datasets. Due to the fact that pixels are only related with the adjacent and close pixels, convolution allows us to preserve the relationship between different parts of an image. Convolution is basically filtering the image with a smaller pixel filter to decrease the size of the image without losing the relationship between pixels. When we apply convolution to 5x5 image by using a 3x3 filter with 1x1 stride (1 pixel shift at each step). We will end up having a 3x3 output (64% decrease in complexity).



**Fig 3.2.2:** Convolution of 5 x 5 pixel image with 3 x 3 pixel filter (stride = 1 x 1 pixel)

**3.2.3. Polling layer**

When constructing CNNs, it is common to insert pooling layers after each convolution layer to reduce the spatial size of the representation to reduce the parameter counts which reduces the computational complexity. In addition, pooling layers also helps with the overfitting problem. Basically we select a pooling size to reduce the amount of the parameters by selecting the maximum, average, or sum values inside these pixels. Max Pooling, one of the most common pooling techniques, may be demonstrated as follows:



**Fig 3.2.3 :** Polling of 5x5 to 3x3

**3.2.4. MAX POOLING 2D**

Let's say we have a 4x4 matrix representing our initial input.

Let's say, as well, that we have a 2x2 filter that we'll run over our input. We'll have a stride of 2 (meaning the (dx, dy) for stepping over our input will be (2, 2)) and won't overlap regions.

For each of the regions represented by the filter, we will take the max of that region and create a new, output matrix where each element is the max of a region in the original input.



**Fig 3.2.4 :** Max pool with 2x2 filters

**3.2.5. Rectified Linear Unit (RELU)**

The purpose of applying the rectifier function is to increase the non-linearity in our images. The reason we want to do that is that images are naturally non-linear.

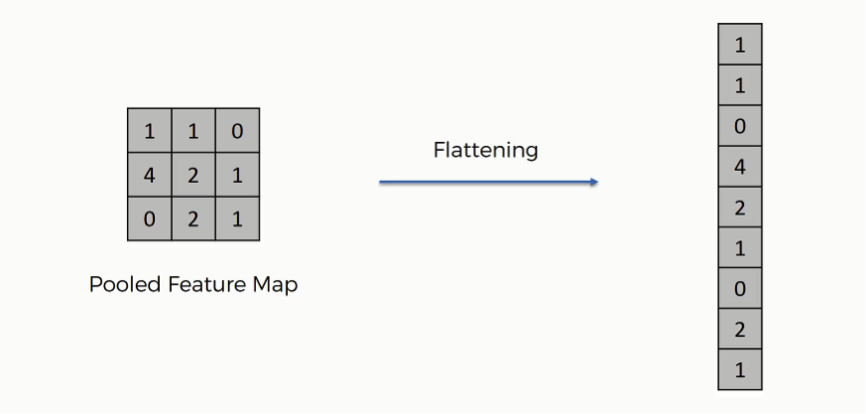
**3.2.6. Loss Layer**

The "loss layer" specifies how training penalizes the deviation between the predicted (output) and true labels and is normally the final layer of a neural network. Various loss functions appropriate for different tasks may be used.

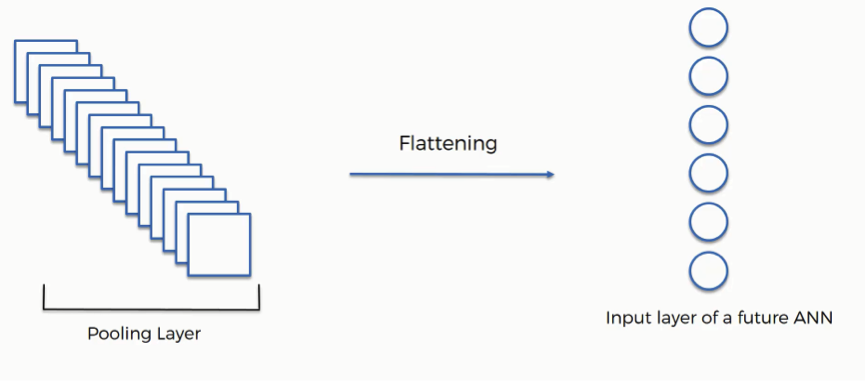
Softmax loss is used for predicting a single class of K mutually exclusive classes.[nb 3] Sigmoid cross-entropy loss is used for predicting K independent probability values in[0,1]. Euclidean loss is used for regressing to real-valued labels.

**3.2.7. Flattering**

Flattering is done after pooling. The reason we do this is that we're going to need to insert this data into an artificial neural network later on.



**Fig 3.2.7.1:** Flattening of feature map



**Fig 3.2.7.2:** Flattening of pooling Layer

As you can see from the above figure after the flattening step you end up with a long vector of input data that you then pass through the artificial neural network to have it processed further.

**3.2.8. Fully Connected Layers**

The role of the artificial neural network is to take this data and combine the features into a wider variety of attributes that make the convolutional network more capable of classifying images, which is the whole purpose from creating a convolutional neural network.

A fully connected network is our RegularNet where each parameter is linked to one another to determine the true relation and effect of each parameter on the labels. Since our time-space complexity is vastly reduced thanks to convolution and pooling layers, we can construct a fully connected network in the end to classify our images. A set of fully connected layers looks like this:



**Fig 3.2.8 :** Fully Connected Layers

For example we have to classify whether the given image is dog or cat The fully connected layer works as follows:

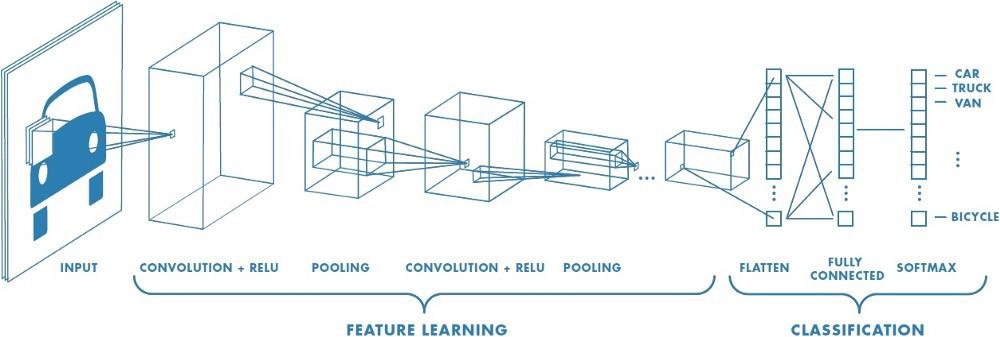
In our example, the weight placed on the nose-dog synapse is high (1.0), which means that the network is confident that this is a dog's nose.

* The neuron in the fully-connected layer detects a certain feature; say, a nose.
* It preserves its value.
* It communicates this value to both the “dog” and the “cat” classes.
* Both classes check out the feature and decide whether it's relevant to them.

**3.2.9. SOFTMAX LAYER**

The softmax activation is normally applied to the very last layer in a neural net, instead of using ReLU, sigmoid, tanh, or another activation function. The reason why softmax is useful is because it converts the output of the last layer in your neural network into what is essentially a probability distribution.

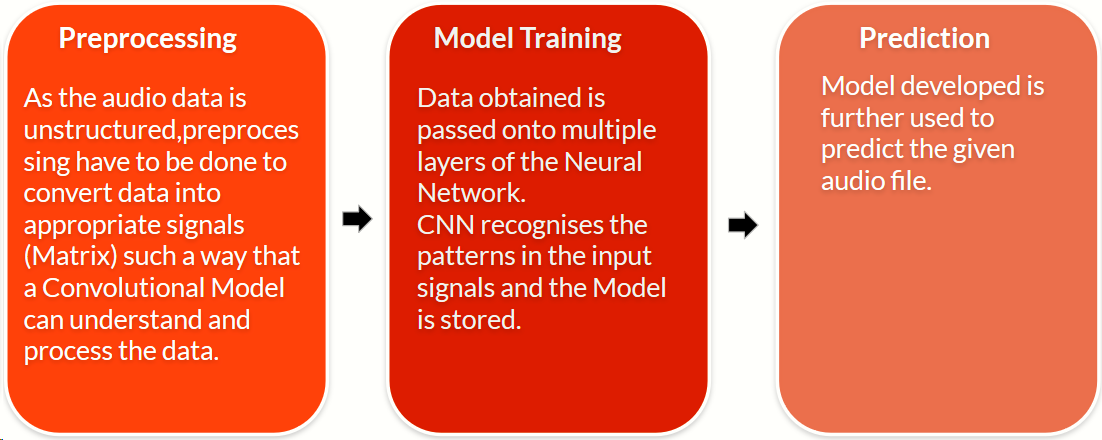
**3.2.10. Overview of entire CNN Working**



**Fig 3.2.10:** Overview of CNN

**4. SYSTEM DESIGN**

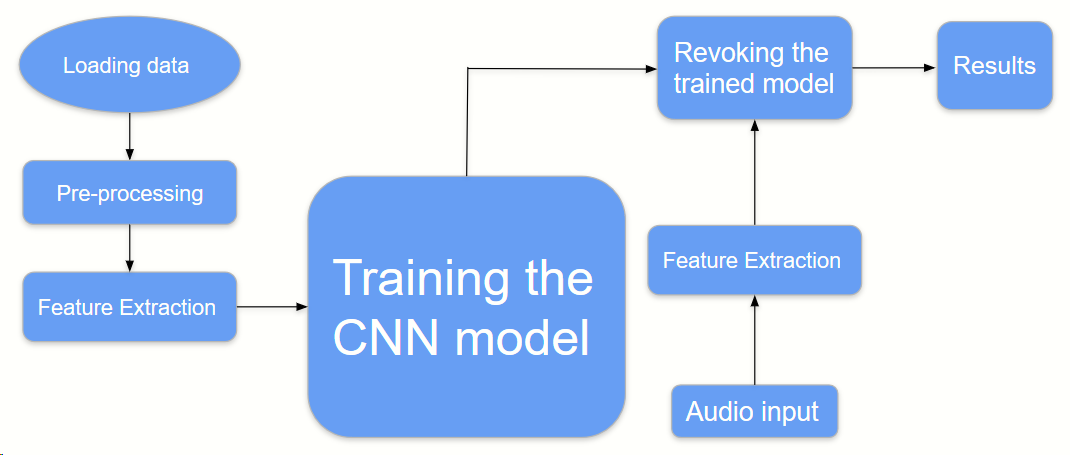
**4.1 SYSTEM ARCHITECTURE**



**Fig 4.1:** System Architecture

As the audio data is unstructured, preprocessing has to be done to convert data into appropriate signals (Matrix) such a way that a Convolutional Model can understand and process the data. Data obtained is passed onto multiple layers of the Neural Network. CNN recognizes the patterns in the input signals and the Model is stored. Model developed is further used to predict the given audio file.

* 1. **Data Flow Diagram / Flow Chart**



**Fig 4.2 :** Flow Chart

**5. IMPLEMENTATION**

**5.1 MODULES**

1. Dataset preparation and Pre-processing Module

2. Feature Extraction and Model Building Module

3. Prediction Module

**5.1.1 Dataset preparation and Pre-processing Module**

Data is the foundation for any machine learning project. The second stage of project implementation is complex and involves data collection, selection, preprocessing, and transformation.

Audio preprocessing is a two stage process, the aim of which is to ensure audio assets from one session to another match before you begin using them in a project.

The first stage being the pre-edit and processing of raw audio ,to a common standard ,before applying FX processing. This typically involves the removal of unwanted sections, to leave a clean audio file, and then measuring the RMS level and normalizing the audio to a predetermined RMS level so that the audio files are at the same RMS level prior to any FX processing.

The second stage is the use of FX to remove unwanted noise, to provide a set of clean, leveled audio assets.

Typically you would save different versions for each of these processes in separate named folders, such as Raw audio, clean , RMS leveled etc.

This process means that audio recorded from one session to another can be easily matched and any FX processing, such as dynamics and EQ, will work the same for all audio files they are applied to.

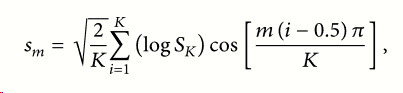
**5.2.2 Feature Extraction and Model Building Module**

**Mel-Frequency Cepstral Coefficients(MFCCs)**

Mel-frequency cepstral coefficients are the logarithmic measure of the Mel magnitude spectrum, which is calculated by triangular band-pass filter. These values are decorrelated using discrete cosine transform. MFCC is a real-valued implementation of complex cepstrum; that is why it is calculated by taking FFT. The steps for calculating MFCC are as follows:

1. Divide the audio signal into short frames.
2. Take Fourier transform of each frame and calculate periodogram-based power spectral estimate for each frame.
3. Take log of all filter bank energies.
4. Take discrete cosine transform of each Mel log power.
5. The amplitudes of resulting spectrum are MFCCs.

MFCCs have good discriminating capability. That is why most of the speech recognition systems use them as a strong feature.



The above extracted features will be fed to the **CNN** model for training**.** And the model will be stored for prediction and classification.

Now we will train our model. To train, we will use the ‘fit()’ function on our model with the following parameters :model.fit(X , y , epochs=10 , batch\_size = 32 ,shuffle=True, validation\_split=0.1, callbacks=[checkpoint])

**5.2.3 Prediction Module**

When the model is fitted, we use the predict method to make predictions using new audio. In order to do this we need to preprocess our audio file and extract the features before we pass them to the predict method. To achieve this we used MFCC audio.

**5.2 Code Snippets**

**Pre Processing and creating clean audio files**

import os

import librosa

from tqdm import tqdm

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from scipy.io import wavfile

from python\_speech\_features import mfcc, logfbank

signals={}

fft={}

fbank={}

mfccs={}

def calc\_fft(y,rate):

n=len(y)

freq=np.fft.rfftfreq(n,d=1/rate)

Y=abs(np.fft.rfft(y)/n)

return (Y,freq)

def plot\_fft(fft):

fig, axes = plt.subplots(nrows=2, ncols=5, sharex=False,

sharey=True, figsize=(20,5))

fig.suptitle('Fourier Transforms', size=16)

i = 0

for x in range(2):

for y in range(5):

data = list(fft.values())[i]

Y, freq = data[0], data[1]

axes[x,y].set\_title(list(fft.keys())[i])

axes[x,y].plot(freq, Y)

axes[x,y].get\_xaxis().set\_visible(False)

axes[x,y].get\_yaxis().set\_visible(False)

i += 1

def plot\_signals(signals):

fig, axes = plt.subplots(nrows=2, ncols=5, sharex=False,

sharey=True, figsize=(20,5))

fig.suptitle('Time Series', size=16)

i = 0

for x in range(2):

for y in range(5):

axes[x,y].set\_title(list(signals.keys())[i])

axes[x,y].plot(list(signals.values())[i])

axes[x,y].get\_xaxis().set\_visible(False)

axes[x,y].get\_yaxis().set\_visible(False)

i += 1

def plot\_fbank(fbank):

fig, axes = plt.subplots(nrows=2, ncols=5, sharex=False,

sharey=True, figsize=(20,5))

fig.suptitle('Filter Bank Coefficients', size=16)

i = 0

for x in range(2):

for y in range(5):

axes[x,y].set\_title(list(fbank.keys())[i])

axes[x,y].imshow(list(fbank.values())[i],

cmap='hot', interpolation='nearest')

axes[x,y].get\_xaxis().set\_visible(False)

axes[x,y].get\_yaxis().set\_visible(False)

i += 1

def envolop(y,rate, threshold):

mask=[]

y=pd.Series(y).apply(np.abs)

y\_mean=y.rolling(window=int(rate/10) , min\_periods=1,center=True).mean()

for mean in y\_mean:

if mean > threshold:

mask.append(True)

else:

mask.append(False)

return mask

def plot\_mfccs(mfccs):

fig, axes = plt.subplots(nrows=2, ncols=5, sharex=False,

sharey=True, figsize=(20,5))

fig.suptitle('Mel Frequency Cepstrum Coefficients', size=16)

i = 0

for x in range(2):

for y in range(5):

axes[x,y].set\_title(list(mfccs.keys())[i])

axes[x,y].imshow(list(mfccs.values())[i],

cmap='hot', interpolation='nearest')

axes[x,y].get\_xaxis().set\_visible(False)

axes[x,y].get\_yaxis().set\_visible(False)

i += 1

df= pd.read\_csv("C://Users//home//Desktop//Audio-Classification-master//instruments.csv")

df.set\_index('fname', inplace= True)

for f in df.index:

rate,signal= wavfile.read("C://Users//home//Desktop//Audio-Classification-master//wavfiles/"+f)

df.at[f,'length']=signal.shape[0]/rate

classes=list(np.unique(df.label))

class\_dist= df.groupby(['label'])['length'].mean()

fig ,ax= plt.subplots()

ax.set\_title('Class Distribution' , y=1.08)

ax.pie(class\_dist,labels=class\_dist.index, autopct='%1.1f%%', shadow= False ,startangle= 90)

ax.axis('equal')

plt.show()

df.reset\_index(inplace= True)

for c in classes:

wav\_file= df[df.label == c ].iloc[0,0]

signal, rate= librosa.load("C://Users//home//Desktop//Audio-Classification-master//wavfiles/"+wav\_file, sr=44100)

mask=envolop(signal,rate,0.0005)

signal=signal[mask]

signals[c]=signal

fft[c]=calc\_fft(signal,rate)

bank =logfbank(signal[:rate], rate, nfilt=26 , nfft=1103).T

fbank[c]=bank

mel=mfcc(signal[:rate], rate, nfilt=26 , nfft=1103,numcep=13).T

mfccs[c]=mel

plot\_signals(signals)

plt.show()

plot\_fft(fft)

plt.show()

plot\_mfccs(mfccs)

plt.show()

if len(os.listdir('C://Users//home//Desktop//Audio-Classification-master//clean1'))==0:

for f in tqdm(df.fname):

signal , rate= librosa.load('C://Users//home//Desktop//Audio-Classification-master//wavfiles/' +f, sr=16000)

mask= envolop(signal, rate, 0.0005)

wavfile.write(filename='C://Users//home//Desktop//Audio-Classification-master//clean1/'+f, rate=rate , data=signal[mask])

**Feature Extraction and Model Building**

import os

from scipy.io import wavfile

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from tensorflow.keras.layers import Convolution2D, MaxPool2D, Flatten, LSTM

from tensorflow.keras.layers import Dropout, Dense, TimeDistributed

from tensorflow.keras.models import Sequential

from keras.utils import to\_categorical

from sklearn.utils.class\_weight import compute\_class\_weight

from tqdm import tqdm

from python\_speech\_features import mfcc

import pickle

from keras.callbacks import ModelCheckpoint

from cfg import Config

def chech\_data():

if os.path.isfile(config.p\_path):

print('Loading existing data for {} model' . format(config.mode))

with open(config.p\_path , 'rb') as handle:

tmp=pickle.load(handle)

return tmp

else:

return None

def build\_rand\_feat():

tmp= chech\_data()

if tmp:

return tmp.data[0],tmp.data[1]

X=[]

y=[]

\_min , \_max =float('inf') , -float('inf')

for \_ in tqdm(range(n\_samples)):

rand\_class=np.random.choice(class\_dist.index, p=prob\_dist)

file= np.random.choice(df[df.label==rand\_class].index)

rate, wav = wavfile.read('C://Users//home//Desktop//Audio-Classification-master//clean1//' +file)

label=df.at[file, 'label']

rand\_index= np.random.randint(0,wav.shape[0]-config.step)

sample=wav[rand\_index:rand\_index+config.step]

X\_sample=mfcc(sample, rate ,

numcep=config.nfeat, nfilt=config.nfilt, nfft=config.nfft)

\_min= min(np.amin(X\_sample) , \_min)

\_max= max(np.amax(X\_sample) , \_max)

X.append(X\_sample if config.mode=='conv' else X\_sample.T)

y.append(classes.index(label))

X.append(X\_sample)

y.append(classes.index(label))

config.min= \_min

config.max=\_max

X ,y = np.array(X) , np.array(y)

X = (X - \_min) / (\_max - \_min)

if config.mode =='conv':

X= X.reshape(X.shape[0] , X.shape[1] , X.shape[2] , 1)

elif config.mode == 'time':

X= X.reshape(X.shape[0] , X.shape[1] , X.shape[2])

y=to\_categorical(y,num\_classes=10)

config.data=(X,y)

with open(config.p\_path , 'wb') as handle:

pickle.dump(config, handle, protocol=2)

return X , y

def get\_conv\_model():

model=Sequential() # rectified linear unit

model.add(Convolution2D(filters=16, kernel\_size=(3,3), activation='relu', strides=(1,1),

padding='same', input\_shape = input\_shape ))

model.add(Convolution2D(32, (3,3), activation='relu', strides=(1,1),

padding='same'))

model.add(Convolution2D(64, (3,3), activation='relu', strides=(1,1),

padding='same'))

model.add(Convolution2D(128, (3,3), activation='relu', strides=(1,1),

padding='same'))

model.add(MaxPool2D((2,2)))

model.add(Dropout(0.5))

model.add(Flatten())

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

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

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

model.summary()

model.compile(loss='categorical\_crossentropy',

optimizer='adam',

metrics=['acc'])

return model

def get\_recurrent\_model():

model=Sequential()

model.add(LSTM(128,return\_sequences=True, input\_shape=input\_shape))

model.add(LSTM(128,return\_sequences=True))

model.add(Dropout(0.5))

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

model.add(TimeDistributed(Dense(32, activation='relu')))

model.add(TimeDistributed(Dense(16, activation='relu')))

model.add(TimeDistributed(Dense(8, activation='relu')))

model.add(Flatten())

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

model.summary()

model.compile(loss='categorical\_crossentropy',

optimizer='adam',

metrics=['acc'])

return model

df = pd.read\_csv('C://Users//home//Desktop//Audio-Classification-master//instruments.csv')

df.set\_index('fname', inplace=True)

for f in df.index:

rate, signal = wavfile.read("C://Users//home//Desktop//Audio-Classification-master//clean1//" +f)

df.at[f, 'length'] = signal.shape[0]/rate

classes = list(np.unique(df.label))

class\_dist = df.groupby(['label'])['length'].mean()

n\_samples= 2 \* int(df['length'].sum()/0.1)

prob\_dist=class\_dist / class\_dist.sum()

choices= np.random.choice(class\_dist.index , p=prob\_dist)

"""fig, ax = plt.subplots()

ax.set\_title('Class Distribution', y=1.08)

ax.pie(class\_dist, labels=class\_dist.index, autopct='%1.1f%%',

shadow=False, startangle=90)

ax.axis('equal')

plt.show()

"""

config=Config(mode= 'conv')

if config.mode=='conv':

X,y=build\_rand\_feat()

y\_flat=np.argmax(y, axis=1)

input\_shape=(X.shape[1], X.shape[2] ,1)

model= get\_conv\_model()

elif config.mode=='time':

X,y=build\_rand\_feat()

y\_flat=np.argmax(y, axis=1)

input\_shape=(X.shape[1], X.shape[2])

model= get\_recurrent\_model()

class\_weights= compute\_class\_weight( 'balanced' , np.unique(y\_flat) , y\_flat)

print(class\_weights)

checkpoint = ModelCheckpoint(config.model\_path, monitor='val\_acc' , verbose=1,

mode= 'max',save\_best\_only=True,

save\_weights\_only=False, period=1 )

model.fit(X , y , epochs=10 , batch\_size = 32 ,shuffle=True, validation\_split=0.1, callbacks=[checkpoint])

model.save(config.model\_path)

**Configuration part**

import os

class Config:

def \_\_init\_\_(self, mode='conv' , nfilt=26 , nfeat=13 , nfft =512 ,rate=16000):

self.mode=mode

self.nfilt=nfilt

self.nfeat=nfeat

self.nfft=nfft

self.rate=rate

self.step=int(rate/10)

self.model\_path=os.path.join('C://Users//home//Desktop//Audio-Classification-master//models', mode + '.model') self.p\_path=os.path.join('C://Users//home//Desktop//Audio-Classification-master//pickles',mode + '.p')

**Classification**

import os

import pandas as pd

import numpy as np

from tensorflow.keras.models import load\_model

import pickle

from scipy.io import wavfile

from python\_speech\_features import mfcc

from tqdm import tqdm

from sklearn.metrics import accuracy\_score

def build\_predictions(audio\_dir):

y\_true=[]

y\_pred=[]

fn\_prob={}

print('Extracting features from Audio')

for fn in tqdm(os.listdir(audio\_dir)):

rate, wav = wavfile.read(os.path.join(audio\_dir, fn))

label=fn2class[fn]

c= classes.index(label)

y\_prob=[]

for i in range(0, wav.shape[0]-config.step , config.step):

sample=wav[i:i+config.step]

x=mfcc(sample, rate ,

numcep=config.nfeat, nfilt=config.nfilt, nfft=config.nfft)

x= (x - config.min)/(config.max - config.min)

if config.mode== 'conv':

x=x.reshape(1,x.shape[0],x.shape[1],1)

else:

x=np.expand\_dims(x,axis=0)

y\_hat=model.predict(x)

y\_prob.append(y\_hat)

y\_pred.append(np.argmax(y\_hat))

y\_true.append(c)

fn\_prob[fn]=np.mean(y\_prob, axis=0).flatten()

return y\_true , y\_pred , fn\_prob

df = pd.read\_csv('C://Users//home//Desktop//Audio-Classification-master//instruments.csv')

classes=list(np.unique(df.label))

fn2class=dict(zip(df.fname , df.label))

p\_path= os.path.join('C://Users//home//Desktop//Audio-Classification-master//pickles', 'conv.p')

with open(p\_path , 'rb') as handle:

config= pickle.load(handle)

model= load\_model(config.model\_path)

y\_true , y\_pred , fn\_prob = build\_predictions('C://Users//home//Desktop//Audio-Classification-master//clean1')

acc\_scr=accuracy\_score(y\_true=y\_true , y\_pred=y\_pred)

y\_probs=[]

for i ,row in df.iterrows():

y\_prob=fn\_prob[row.fname]

y\_probs.append(y\_prob)

for c, p in zip(classes , y\_prob):

df.at[i,c]=p

y\_pred= [classes[np.argmax(y)] for y in y\_probs]

df['y\_pred']=y\_pred

df.to\_csv('C://Users//home//Desktop//Audio-Classification-master//prediction.csv', index=False)

**6. TESTING**

**6.1 SYSTEM TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that theSoftware system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**6.2 TYPES OF TESTS**

**Unit Testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration Testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional Test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Testing**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed

# Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

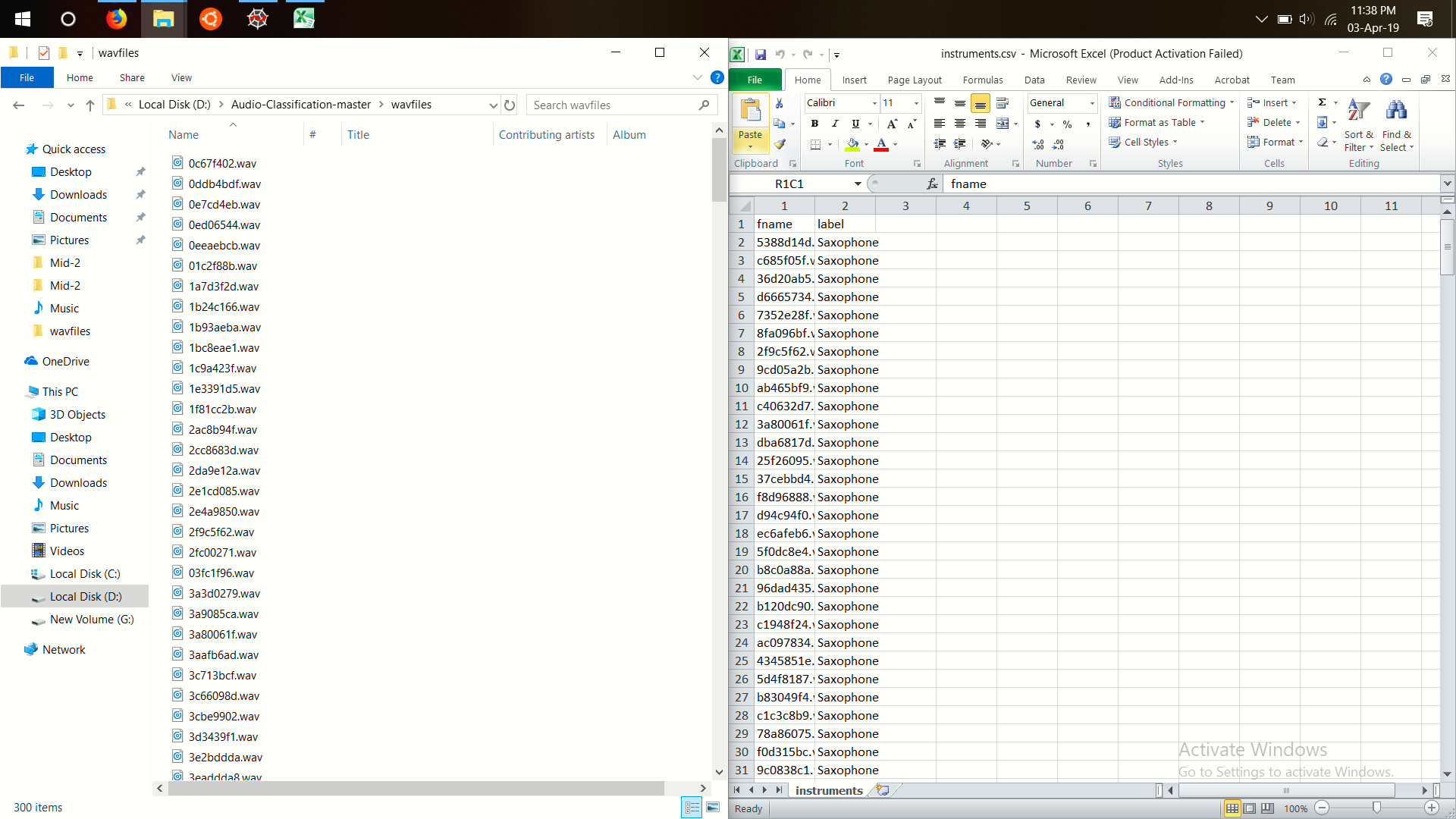
**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

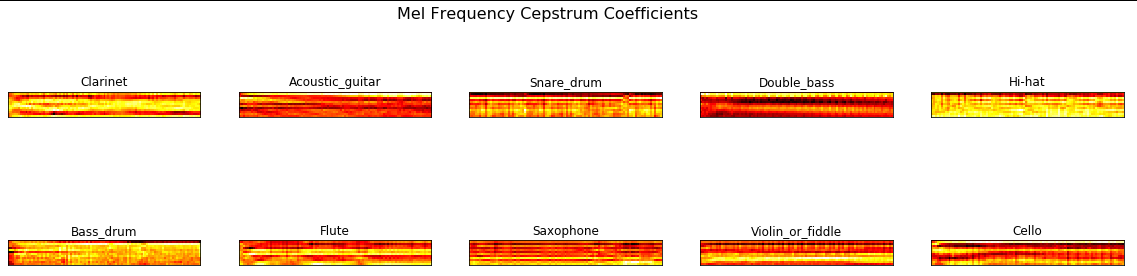
**7. SCREENSHOTS**

**7.1 Data Set**

****

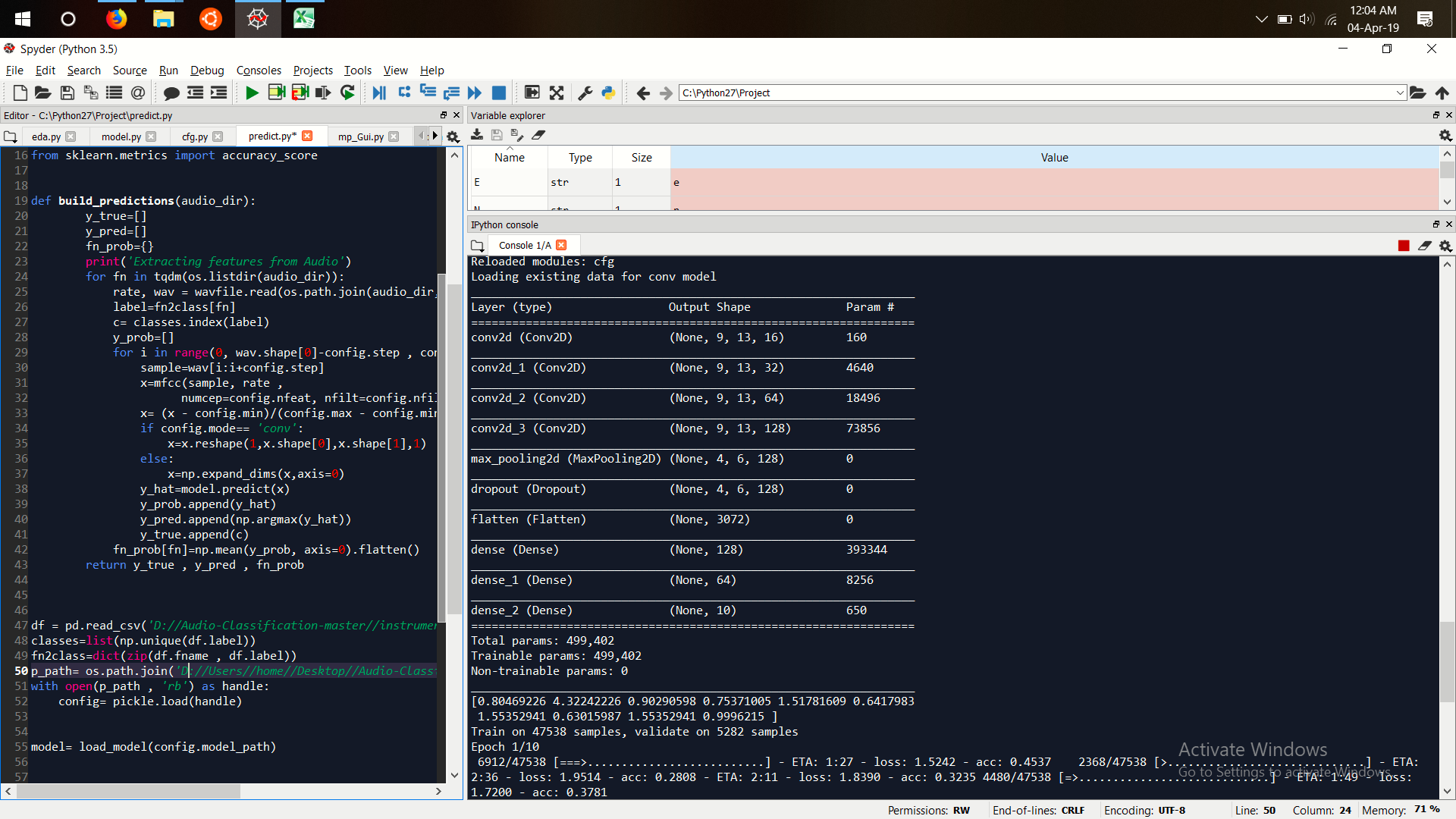
**Fig 7.1 :** Data Set

**7.2 Graph related to Pre-processing**

****

**Fig 7.2 :** Mel Frequency Cepstrum Coefficient

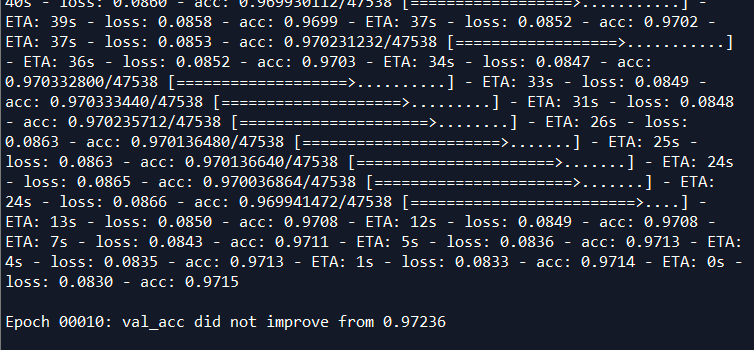
**7.3 Model**

****

**Fig 3.2.7 :** Model Building

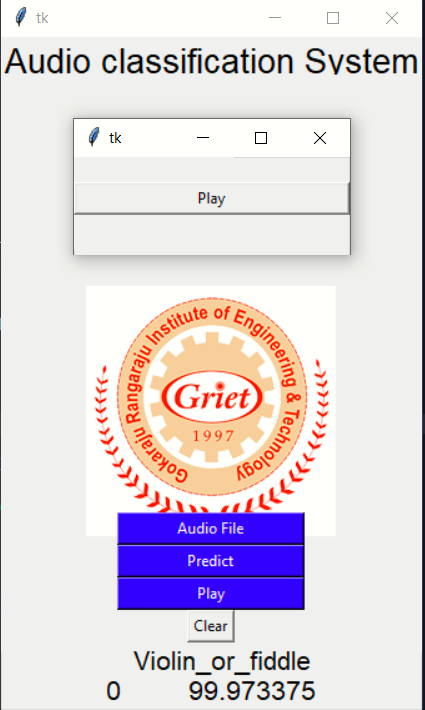
**7.4 Output**

**Model Accuracy**

****

**Fig 7.4 :** Model Accuracy

**7.5 GUI**

****

**Fig 7.5 :** GUI

**8. CONCLUSION**

An effective and fast audio classification model has been built, which provides the result with 95.68% accuracy on average. Main goal is to design a system which is capable of recognizing different sounds upon training.

Audio classification is complex and many problems have to be considered when a classifier is designed .First knowledge of what content classifier should classify between is essential for successful classification scheme to make limitations of possible content.In future this algorithm can be used to identifying human activities involving speech. Can also be used as a preprocessing step in automatic speech recognition, video conferencing.

**9. BIBLIOGRAPHY**

1. <https://www.hindawi.com/journals/mpe/2015/209814/>
2. <http://www.diva-portal.org/smash/get/diva2:1030847/FULLTEXT01.pdf>
3. <https://www.analyticsvidhya.com/blog/2017/08/audio-voice-processing-deep-learning/>
4. <https://www.pythonforengineers.com/audio-and-digital-signal-processingdsp-in-python/>
5. <https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8>
6. <https://en.wikipedia.org/wiki/Convolutional_neural_network>