

PROFESSIONAL TRAINING REPORT

AT

Sathyabama Institute of Science and Technology

Submitted in partial fulfillment of the requirements for the
award of
Bachelor of Technology degree in Information Technology

By

Reeya B Varghese
REG. NO. 40120074



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119, TAMIL NADU**

MARCH - 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

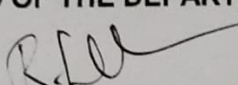
BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **Reeya B Varghese** (Reg. No: 40120074) who carried out the project entitled "**Voice Emotion Recognition using deep learning**" under my supervision from January 2023 to April 2023.

INTERNAL GUIDE

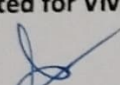

Mr. Amandeep Singh K B.E., M.E., (Ph.D.),

HEAD OF THE DEPARTMENT


Dr. R. Subhashini M.E., Ph.D.,

Submitted for Viva-voce Examination held on

29/4/23


Internal Examiner

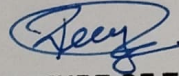

External Examiner

DECLARATION

I, **Reeya B Varghese** hereby declare that the Project Report entitled “ **Voice Emotion Recognition using Deep Learning** ” done by me under the guidance of **Mr. Amandeep Singh K B.E., M.E., (Ph.D.)**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 29/4/23

PLACE: CHENNAI



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **the Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean**, School of Computing **Dr.R.Subhashini M.E., Ph.D.**, Head of the Department of Information Technology for providing me with necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mr. Amandeep Singh K B.E., M.E.,(Ph.D.)**, for her valuable guidance, suggestions, and constant encouragement paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

ABSTRACT

As human beings speech is amongst the most natural way to express ourselves. We depend so much on it that we recognize its importance when resorting to other communication forms like emails and text messages where we often use emojis to express the emotions associated with the messages. As emotions play a vital role in communication, the detection and analysis of the same is of vital importance in today's digital world of remote communication. Emotion detection is a challenging task, because emotions are subjective. There is no common consensus on how to measure or categorize them. We define a VER – Voice emotion recognition system that process and classify speech signals to detect emotions embedded in them. Such a system can find use in a wide variety of application areas like interactive voice based-assistant or call center agent conversation analysis. In this project, we attempt to detect underlying emotions in recorded audio by analysing the acoustic features of the audio data of recordings. the representation of emotions can be done in two ways: Discrete Classification: Classifying emotions in discrete labels like anger, happiness, boredom, etc. this task has a lot of challenges that hinder the prediction of emotion. Emotions are subjective, people would interpret it differently. It is hard to define the notion of emotions. Annotating an audio recording is challenging. Should we label a single word, sentence or a whole conversation? How many emotions should we define to recognize? Collecting data is complex. There are lots of audio data can be achieved from films or news. However, both of them are biased since news reporting has to be neutral and actors' emotions are imitated. It is hard to look for neutral audio recording without any bias. Labelling data require high human and time cost. Unlike drawing a bounding box on an image, it requires trained personnel to listen to the whole audio recording, analysis it and give an annotation. The annotation result has to be evaluated by multiple individuals due to its subjectivity.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	viii
1.	INTRODUCTION	1
	1.1 AIM AND OBJECTIVES	1
	1.2 SCOPE OF THE PROJECT	2
2.	LITERATURE SURVEY	3
	2.1 SPEECH EMOTION RECOGNITION USING DEEP LEARNING TECHNIQUES: A REVIEW	3
	2.2 SPEECH EMOTION RECOGNITION FROM 3D LOG-MEL SPECTROGRAMS WITH DEEP LEARNING NETWORK	4
3.	METHODOLOGY	5
	3.1 DATA COLLECTION	5
	3.2 SOFTWARE SPECIFICATIONS	7

3.3	HARDWARE SPECIFICATIONS	7
3.4	DATA PREPARATION AND PRE – PROCESSING	8
3.5	DATA VISUALIZATION AND EXPLORATION	10
3.6	DATA AUGMENTATION AND FEATURE EXTRACTION	14
3.7	BUILDING THE MODEL	16
3.8	APPLICATION BUILDING	18
4.	RESULTS AND DISCUSSION	19
4.1	MODEL EVALUATION	19
4.2	RUNNING THE APP	22
5.	CONCLUSION AND FUTURE WORK	23
5.1	CONCLUSION	23
5.2	FUTURE WORK	23
	REFERENCES	24
	APPENDICES	26

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
3.1.1	<i>System Architecture</i>	7
3.2.1	<i>Dataset preview</i>	9
3.5.1	<i>Bar plot representing the count of the emotions</i>	11
3.5.2	<i>Wave plot and spectrogram for the audio with fear emotion</i>	12
3.5.3	<i>Wave plot and spectrogram for the audio with angry emotion</i>	12
3.5.4	<i>Wave plot and spectrogram for the emotion of happy</i>	13
3.5.5	<i>Wave plot and spectrogram for the emotion of sad</i>	13
3.6.1	<i>Shape of x train, y train, x test, and y test.</i>	15
3.6.2	<i>Shape after expansion.</i>	15

3.7.1	<i>Summary of the model</i>	17
3.8.1	<i>HTML index page</i>	18
4.1.1	<i>Test data accuracy</i>	19
4.1.2	<i>History loss and accuracy</i>	19
4.1.3	<i>Classification report of the model</i>	20
4.1.4	<i>Confusion matrix of each label</i>	21
4.2.1	<i>Running the flask application</i>	22
4.2.2	<i>Prediction page</i>	22

CHAPTER – 1

INTRODUCTION

1.1 AIM AND OBJECTIVES

Voice Emotion Recognition, abbreviated as VER, is the act of attempting to recognize human emotion and affective states from the voice or speech. This is capitalizing on the fact that voice often reflects underlying emotion through tone and pitch. This is also the phenomenon that animals like dogs and horses employ to be able to understand human emotion. There are various audio processing techniques to capture this hidden layer of information that can amplify and extract tonal and acoustic features from speech. In this model, we extract the audio features such as MFCC, chroma, Zero Crossing Rate, RMS (root mean square) value, and Mel spectrogram from a sound file. The features are then fed into a convolutional neural network model for training the model to detect the emotion of the audio. The model is trained to produce the discrete classification of the emotions into 8 categories such as anger, neutral, sad, happy, disgust, surprise, fear, and calm. The 1-dimensional convolution neural network model is saved as an HDF5 file and it is integrated with the flask application. A simple User Interface is created for the user to interact with the model and get the predicted emotion. The data used in this project was combined from four different data sources: the TESS (Toronto Emotional Speech Set): 2 female speakers (young and old), 2800 audio files. SAVEE (Surrey Audio-Visual Expressed Emotion): 4 male speakers, 480 audio files. RAVDESS: 2452 audio files, with 12 male speakers and 12 Female speakers. CREMA-D (Crowd-Sourced Emotional Multimodal Actors Dataset): 7442 audio files, 91 different speakers (48 male and 43 female between the ages of 20 and 74) of different races and ethnicities. Various Data augmentation techniques are also used to make our model invariant to the perturbations in the audio and enhance its ability to generalize. Finally, the model is deployed using the flask web application framework for the user where he or she can upload the audio .wav file and get the predicted emotion of the voice.

1.2 SCOPE OF THE PROJET

Applications of voice emotion recognition are more nuanced and add a newer dimension to the use of AI and how it can make our lives easier to improve them.

1. **Online learning:** A very recent application of VER has emerged from the sudden rise in online learning where educators can observe a student's response in class and highlight pointers that might help them assist the student's education.
2. **Interview:** Another upcoming use is to evaluate candidates applying for managerial positions by analysing their responses during audio or video interviews. Their confidence or trepidations can be quantitatively measured for the first time using VER, and thus hiring managers can select the candidate with the ideal fit.
3. **Call centre industry:** VER can help evaluate the performance of existing employees – especially in the call-centre industry where an improper conversation with a customer can be disastrous for the company's image. Such systems can also help address customer grievances more efficiently and even automatically.
4. **Heath care:** The employee's emotional health can also be tracked and cared for. We can understand the emotional state of the employee and provide better service to present depression and stress.

CHAPTER – 2

LITERATURE SURVEY

2.1 SPEECH EMOTION RECOGNITION USING DEEP LEARNING TECHNIQUES: A REVIEW

Deep learning is derived from the family of machine learning, which is a broader learning technique for data representation such as emotions [1], [2]. This deep learning can be un-supervised, semi-supervised, or fully supervised. Currently, deep learning is a fast-growing research area due to its multi-layered structure and efficient results delivery. These research areas include speech emotion recognition, speech and image recognition, natural language processing, and pattern recognition [3], [4]. CNN is another type of Deep learning technique based solely on feed-forward architecture [5] for classification. CNNs are commonly used for pattern recognition and provide better data classification. These networks have small size neurons present on every layer of the designed model architecture that process the input data in the form of receptive fields [6]. Filters are the base of local connections that are convolved with the input and share the same parameters (weight W_i and bias n_i) to generate i feature maps (z_i), each of size $a - b - 1$. The convolutional layers compute the dot product between the weights and provided inputs. So, the parameters for weight W_i and biasing n_i for generation of maps z_i for i features with sizes $a - b - 1$ can be given as $z_i = g(W_i * r + n_i)$ (7) An activation function f or a non-linear methodology needs to be applied to get the output of the convolution layers. It should be noted that inputs are very small regions of the original volumes. Down sampling is carried out at each subsampling layer to feature maps and decrease the parameters in the network. This, in turn, controls overfitting and boosts the training process. The pooling process is carried out over $p \times p$ elements (also known as filter size) for adjoining expanse of all the feature maps. In the final stage, the layers need to be fully connected as in other neural networks. These later layers take the previous low-level and mid-level features and generate high-level abstraction from the input speech data. The last layer also known as SVM or Softmax is utilized to further generate the score of classification in probabilistic terms to relate to a certain class.

2.2 SPEECH EMOTION RECOGNITION FROM 3D LOG-MEL SPECTROGRAMS WITH DEEP LEARNING NETWORK

Speech emotion recognition has attracted a great deal of attention in human-centered signal processing research. In recent years, many researchers divert their sight line on how to extract features from speech signal and take advantage of these to get the state-of-the-art recognition accuracy. The model is built increasingly that it plays an important role in the process of emotional state. In the last decade years, the model development of SER has undergone great changes from which based on the machine learning algorithm to deep learning framework. Among many classification methods, DNNs are widely used in speech emotion recognition. W. Zheng et al. constructed a CNN architecture to implement emotion recognition on labelled data, the ultimate experimental results showed that their proposed approach outperformed the SVM classification [8]. In recent years, CNN (convolution networks) and their different variants acquire the tremendous achievement in the image classification. And we will get some successful experience from the image classification to apply for speech emotion recognition. We will take the static of Log-Mel spectrums as the one of our inputs that means a feature map if it's in the area of image, and the others are the Log-Mel spectrums' deltas and delta-deltas respectively. These three feature maps comprise the whole model's input that can be utilized to acquire a better performance. The massive experiments show us the Mel spectrum is more suitable in human's auditory sense characteristic that presents the linear distribution under the 1000Hz and the logarithm growth above the 1000Hz, we utilize this point to obtain the Log-Mel spectrum static. The relationship between the Mel spectrum and the frequency is shown in (1). $f_{mel} = 2595 \cdot \lg(1 + f/700\text{Hz})$. In this SER mask, we adopt dilated convolution networks to instead of the tradition CNN, because the dilated CNN could acquire more comprehensive receptive field in the process of features extraction. The dilated CNN is applied for semantic segmentation to fetch a better performance. We use its characteristic of holding the interior information in the mask. In the area of the image classification, it's routine to add the pooling layer for reducing the size of image to raise the receptive field and to follow the upsampling to expand the size of image. However, in the process of changing the size of image, there is some info that could be ignored. Hence, the dilated CNN is developed to overcome this trouble.

CHAPTER – 3

METHODOLOGY

3.1 DATA COLLECTION

The data is collected from the Kaggle website. The data used in this project was combined from four different data sources as mentioned below:

1. TESS (Toronto Emotional Speech Set)
2. SAVEE (Surrey Audio-Visual Expressed Emotion):
3. RAVDESS
4. CREMA-D (Crowd-Sourced Emotional Multimodal Actors Dataset):

3.1.1 Dataset Information

1. RAVDESS dataset

2452 audio files, with 12 male speakers and 12 Female speakers, the lexical features (vocabulary) of the utterances are kept constant by speaking only 2 statements of equal lengths in 8 different emotions by all speakers. The filename identifiers as per the official RAVDESS website:

- Modality (01 = full-AV, 02 = video-only, 03 = audio-only).
- Vocal channel (01 = speech, 02 = song).
- Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
- Emotional intensity (01 = normal, 02 = strong). NOTE: There is no strong intensity for the 'neutral' emotion.
- Statement (01 = "Kids are talking by the door", 02 = "Dogs are sitting by the door").
- Repetition (01 = 1st repetition, 02 = 2nd repetition).
- Actor (01 to 24. Odd-numbered actors are male, even-numbered actors are female).

2. SAVEE dataset

4 male speakers, 480 audio files, same sentences were spoken in 7 different emotions. The audio files in this dataset are named in such a way that the prefix letters describe the emotion classes as follows:

- 'a' = 'anger'
- 'd' = 'disgust'
- 'f' = 'fear'
- 'h' = 'happiness'
- 'n' = 'neutral'
- 'sa' = 'sadness'
- 'su' = 'surprise'

3. TESS dataset

2 female speakers (young and old), 2800 audio files, and random words were spoken in 7 different emotions. The audio files in this dataset are named in such a way that the suffix letters describe the emotion classes as Angry, pleasant, surprise, fear, happy, sad, disgust.

4. CREMA – D dataset

7442 audio files, 91 different speakers (48 male and 43 female between the ages of 20 and 74) of different races and ethnicities, different statements are spoken in 6 different emotions and 4 emotional levels (low, mid, high, and unspecified). The audio files in this dataset are named in such a way that the third term in the filename separated by underscores (-) describes the emotion classes as Angry, pleasant, surprise, fear, happy, sad, disgust.

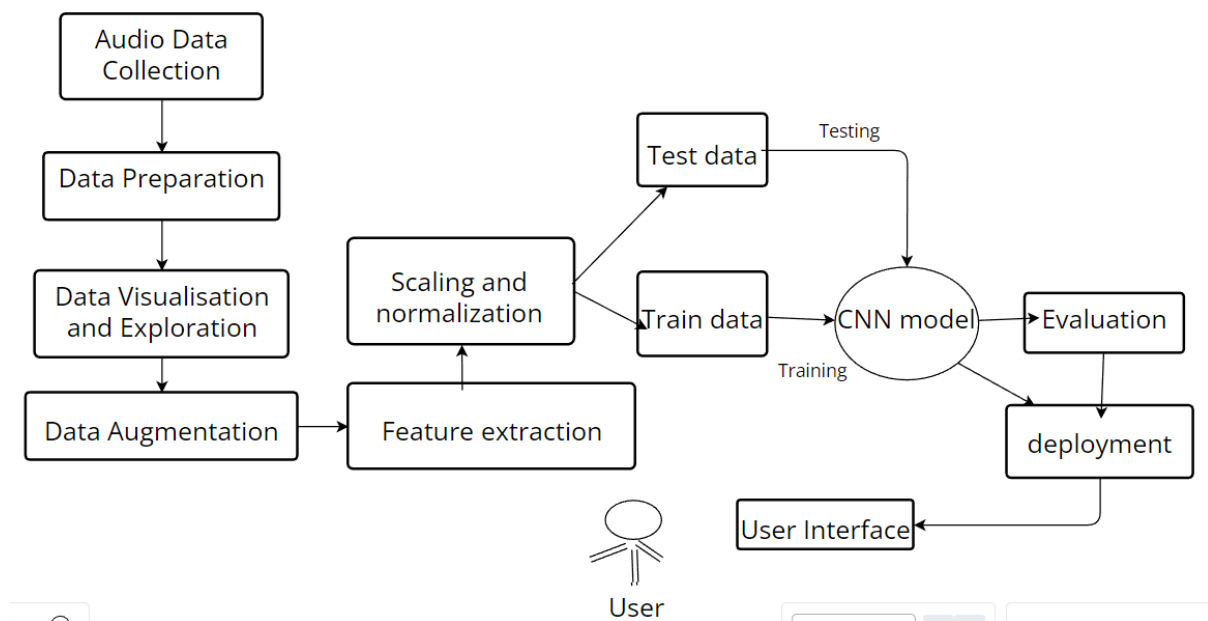


Fig 3.1.1 – System Architecture

3.2 SOFTWARE SPECIFICATION

- Packages: TensorFlow, Keras, Librosa
- Anaconda navigator
- Visual studio code editor
- Jupyter notebook
- Flask micro web Framework (python)
- Anaconda environment

3.3 HARDWARE SPECIFICATION

- Processor: Above Intel Corei7 7th Generation; GPU
- Ethernet connection: (LAN) or a wireless adapter (Wi-Fi)
- Hard Drive: Minimum 1TB HDD or 256 GB SSD
- Memory (RAM): Minimum 16 GB; Recommended 32 GB

3.4 DATA PREPARATION AND PRE – PROCESSING

Data pre-processing is the very first and crucial step in building the model. It prevents data inconsistencies from occurring due to redundancies and null values present in the dataset. As we are dealing with a large dataset that is used to perform complex operations, we preferred to go with google collab, which is a free jupyter notebook environment that runs entirely on the cloud.

3.4.1 *Importing Libraries*

In general, libraries are sets of prewritten code that users can use to streamline processes. In the project, as python is used as an implementation tool, it has the most libraries as compared to other programming languages.

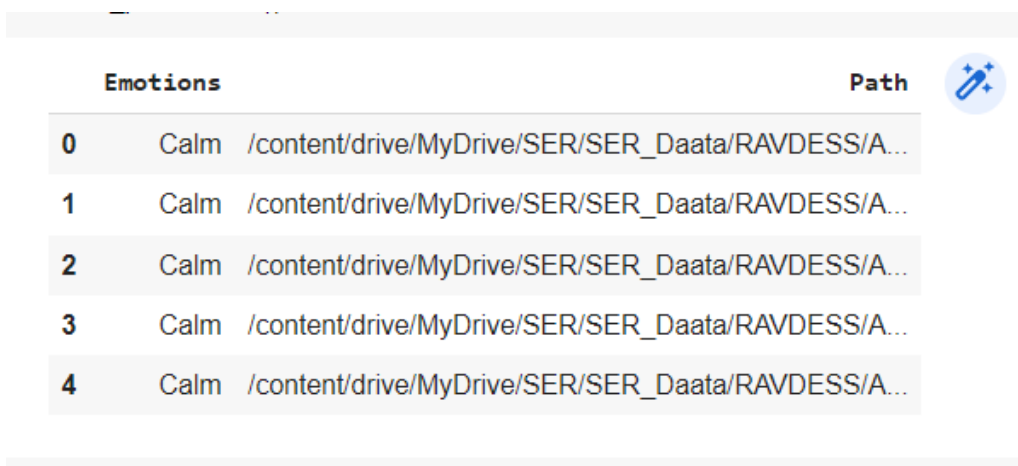
1. NumPy – It is an open-source numerical Python library. It contains a multi-dimensional array and matrix data structures. It can be used to perform mathematical operations on arrays such as trigonometric, statistical, and algebraic routines, etc.,
2. Pandas – It is a fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool. It is built on top of the Python programming language. It provides a lot of functions and methods that expedite the data analysis and preprocessing steps. Has powerful tools like DataFrame and Series for analyzing.
3. Seaborn – A Python data visualization library based on matplotlib is named Seaborn. It offers a sophisticated drawing tool for creating eye-catching and educational statistical visuals. To make statistical graphs more appealing, it offers a variety of default styles and color schemes, and it is also tightly integrated with the data structures from pandas.
4. Matplotlib – Visualisation with python. It is a comprehensive library that is primarily used for creating static, animated, and interactive visualizations in Python. It is a library that has been built on NumPy arrays and designed to work with the broader SciPy stack.
5. Sklearn – stands for Scikit-learn, a machine learning library. It is imported for various pre-processing, and metrics. It is designed

using libraries NumPy and SciPy. We can import various models from this library.

6. Keras - open-source high-level Neural Network library, which is written in Python and is capable enough to run on Theano, TensorFlow, or CNTK. It is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination.
7. Librosa - is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. It has a flatter package layout, standardizes interfaces and names, backward compatibility, modular functions, and readable code.

3.4.2 Importing The dataset

As we are working with four different datasets, a data frame storing all emotions of the data in the data frame with their paths is created. The paths of the audio along with their emotion labels which are categorized according to the file name are collected into a data frame. We will use this data frame to extract features for our model training. This data is saved as a CSV file in the device using `to_csv()` function.



	Emotions	Path
0	Calm	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/A...
1	Calm	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/A...
2	Calm	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/A...
3	Calm	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/A...
4	Calm	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/A...

Fig 3.2.1 – Dataset preview

Analyzing the data:

- i. `head()` is used to return the top n (5 by default) rows of a DataFrame or series.
- ii. The counter object from the collections is used to display the number of emotions and the files in each of the classes.
- iii. The shape attribute in Pandas enables us to obtain the shape of a DataFrame using the command `pd.shape`.

3.5 DATA VISUALIZATION AND EXPLORATION

Data visualization is where a given data set is presented in a graphical format. It helps the detection of patterns, trends, and correlations that might go undetected in text-based data. To visualize the dataset, we need libraries called Matplotlib and Seaborn. Basic properties used when plotting.

xlabel Set the label for the x-axis.

ylabel Set the label for the y-axis.

title: Set a title for the axes.

Legend: Place a legend on the axes

1. Bar plot: A bar plot (or bar chart) is one of the most common types of graphic. It shows the relationship between a numeric and a categorical variable. Each entity of the categorical variable is represented as a bar. The size of the bar represents its numeric value.

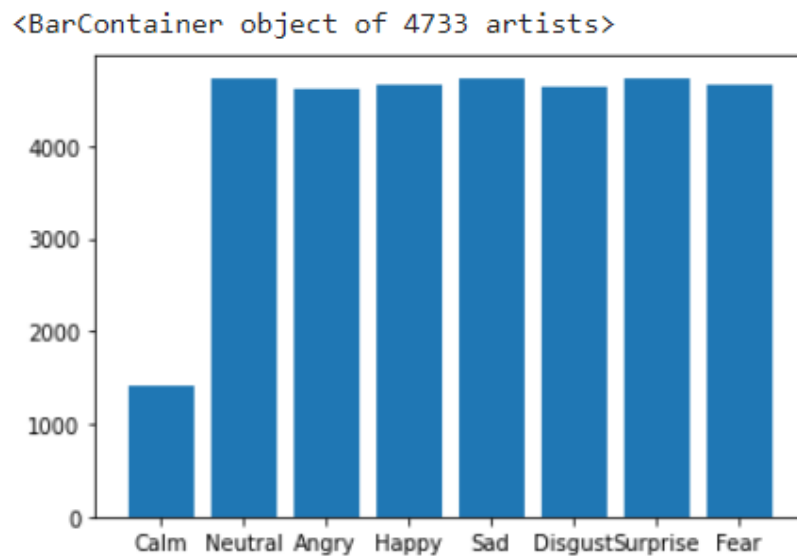
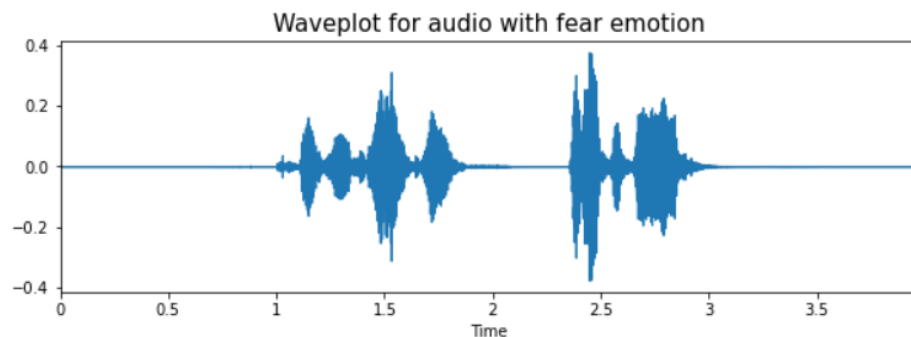


Fig 3.5.1 – Bar plot representing the count of emotions.

2. Wave plots - Wave plots let us know the loudness of the audio at a given time. We can construct the wave plot using the python library librosa. Using the `librosa.display.waveplot()` function.
3. Spectrograms - A spectrogram is a visual representation of the spectrum of frequencies of sound or other signals as they vary with time. It's a representation of frequencies changing with respect to time for given audio/music signals. We can construct the spectrogram using the python library librosa. Using the `librosa.display.specshow()` function.

We have plotted the wave plots and spectrogram for various emotions by taking a random sample to analyse how they differ from each other.



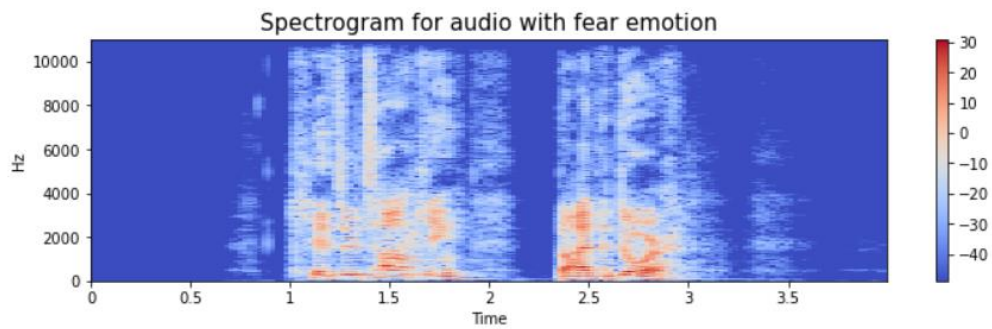
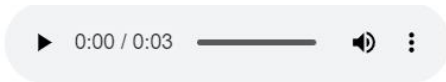


Fig 3.5.2 – Wave plot and Spectrogram for the audio with fear emotion

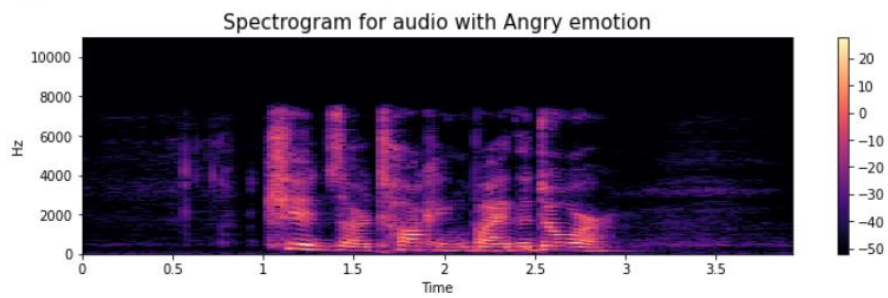
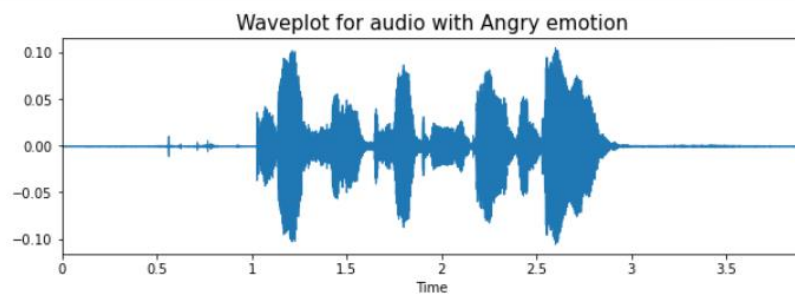
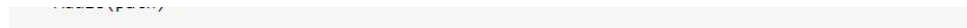


Fig 3.5.3 – Wave plot and Spectrogram for the audio with Angry emotion

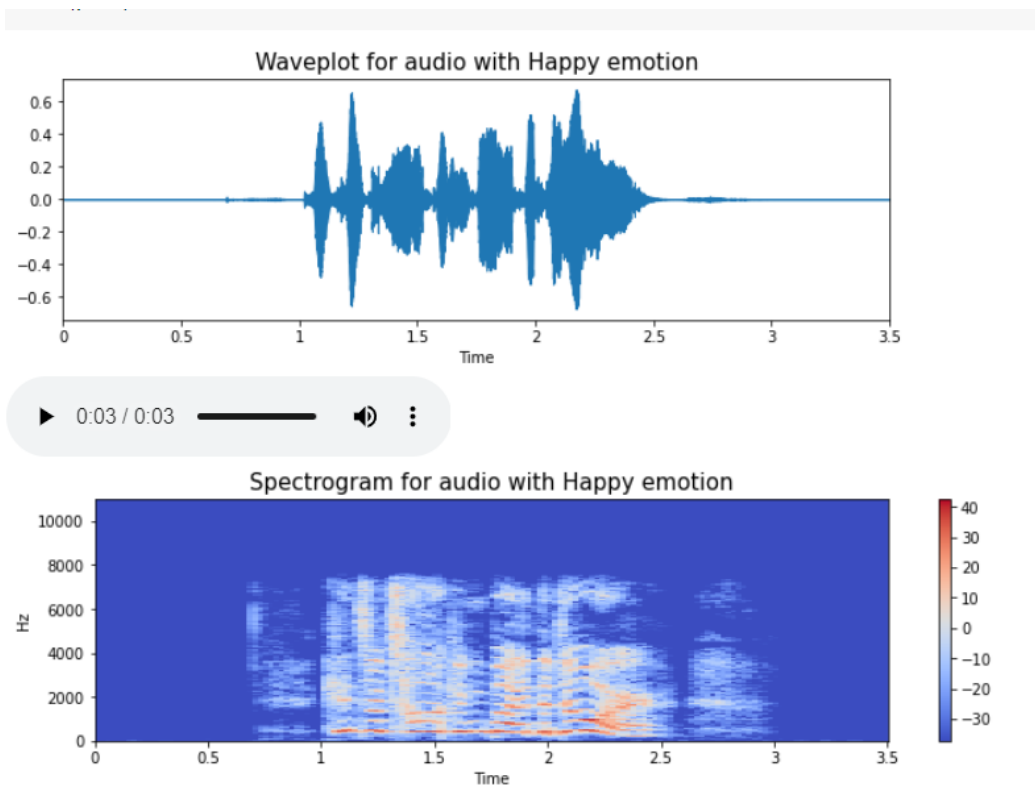


Fig 3.5.4 – Wave plot and Spectrogram for the emotion happy

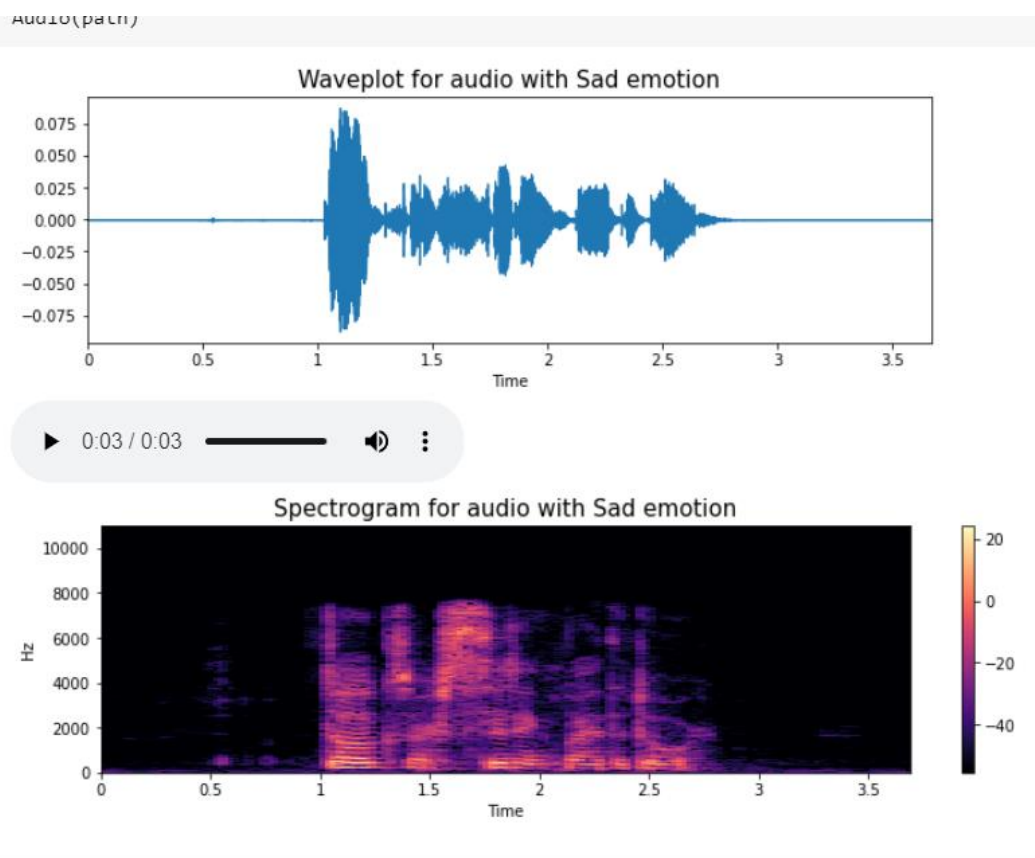


Fig 3.5.5 – Wave plot and Spectrogram for the emotion Sad.

3.6 DATA AUGMENTATION AND FEATURE EXTRACTION

3.6.1 Data Augmentation

Data augmentation is the process by which we create new synthetic data samples by adding small perturbations to our initial training set. To generate syntactic data for audio, we can apply noise injection, shifting time, and changing pitch, and speed. The objective is to make our model invariant to those perturbations and enhances the ability to generalize. In order for this to work adding the perturbations must conserve the same label as the original training sample. We add the augmented audio data to our dataset thus increasing its size. After carefully analysing sample audio after adding the different augmentation techniques, we consider going with noise, stretching (i.e. changing speed), and some pitching to our dataset.

3.6.2 Feature extraction

As the machine learning models can only understand inputs in the form of numbers. we have to find a way to represent the audio file as numbers and this can be done by signal processing techniques. We extract the significant features from the waveform of the audio that can easily help to distinguish emotions embedded. The audio signal is a three-dimensional signal in which three axes represent time, amplitude, and frequency. In this project, I've decided to go with the five audio feature extraction techniques to train our model.

1. Zero Crossing Rate: The rate of sign changes of the signal during the duration of a particular frame.
2. Chroma_stft; A Chroma vector is typically a 12-element feature vector indicating how much energy of each pitch class is present in the signal in a standard chromatic scale.
3. MFCC: Any cepstrum feature is obtained by applying Fourier Transform on a spectrogram. The special characteristic of MFCC is that it is taken on a Mel scale which is a scale that relates the

perceived frequency of a tone to the actual measured frequency. It scales the frequency in order to match more closely what the human ear can hear.

4. RMS(root mean square) value: The RMS (Root-Mean-Square) value is the effective value of the total waveform.
5. MelSpectrogram: It shows how the frequency and amplitude evolve over time. The x-axis represents time, the y-axis represents frequency, and the colour represents amplitude. The brighter the colour, the higher the amplitude. This spectrogram uses a special scale for frequency called the Mel scale.

These features obtained from extraction from audio clips are in a matrix format. To model them on traditional 1-D CNN, we convert the matrices into the 1-D format by taking column means. An audio file represented by the column mean values of MFCC is stacked horizontally with column mean values of all the other 4 features horizontally. These stacked features are added as a new column named features. These features are saved as a CSV file into our drive.

3.6.3 Test and Train splitting

The Labels are taken in the variable Y and fitted using the one Hot Encoder which converts the categorical variables as numerical values. The features are taken in the variable X . The Data is split into test and train using the train_test_split function and their shapes are checked.

```

# split x and y into the training and testing sets
x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state=0, shuffle=True)
x_train.shape, y_train.shape, x_test.shape, y_test.shape

```

↳ ((17849, 162), (17849, 8), (5950, 162), (5950, 8))

Fig 3.6.1 – Shape of xtrain, ytrain, xtest and ytest.

The X_train and X_test are scaled using the Standard Scaler and converted into a shape that is compatible for the CNN model.

```

x_train.shape, y_train.shape, x_test.shape, y_test.shape

```

((17849, 162, 1), (17849, 8), (5950, 162, 1), (5950, 8))

Fig 3.6.2 – Shape after expansion.

3.7 BUILDING THE MODEL

First, The traditional 1 – D CNN model is built using the sequential function of the tensorflow keras library. The model takes in the input of the shape $=(\text{x_train.shape}[1],1)$. The model has four convolutional layers and Max pooling layers stacked and the output is flattened and given to two dense layers. Appropriate dropout is added whenever needed. The convolutional first layer has 256 output channels, 5 kernel sizes, and, 1 stride, padding is given the same, and activation relu, is followed by max pooling. The second convolutional layer is also constructed in the same way. The third convolutional layer consists of 128 output channels and it consists of a dropout of 0.2. the fourth layer consists of 64 output channels. Then the output of these layers is flattened into a 1 – D array using the flatten layer and it is given into the fully connected layers which give the output of size 8 produced by the activation softmax which converts a vector into a probability distribution of n possible outcomes. The model is compiled using the optimizer adam which is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems. It also gives the parameters categorical cross entropy as we dealing the categories of multiple classifications.

Model: "sequential_3"		
Layer (type)	Output Shape	Param #
=====		
conv1d_19 (Conv1D)	(None, 162, 256)	1536
max_pooling1d_17 (MaxPooling1D)	(None, 81, 256)	0
conv1d_20 (Conv1D)	(None, 81, 256)	327936
max_pooling1d_18 (MaxPooling1D)	(None, 41, 256)	0
conv1d_21 (Conv1D)	(None, 41, 128)	163968
max_pooling1d_19 (MaxPooling1D)	(None, 21, 128)	0
dropout_8 (Dropout)	(None, 21, 128)	0
conv1d_22 (Conv1D)	(None, 21, 64)	41024
max_pooling1d_20 (MaxPooling1D)	(None, 11, 64)	0
flatten_4 (Flatten)	(None, 704)	0
dense_9 (Dense)	(None, 32)	22560
dropout_9 (Dropout)	(None, 32)	0
dense_10 (Dense)	(None, 8)	264
=====		
Total params: 557,288		
Trainable params: 557,288		
Non-trainable params: 0		

Fig 3.7.1 – Summary of the model.

We fit the model with x – train, and y – train data and for validation we use x – test and y – test. The model is run for 50 epochs of batch size 64. In the end, we obtain an accuracy of 0.8403 on train data and a validation accuracy of 0.7002. The model is then saved in the HDF5 file format.

3.8 APPLICATION BUILDING

The saved model is deployed using the flask web application framework. A UI is provided for the users where they must upload audio for the predictions. The audio is sent to the saved model and the prediction is displayed on the UI. Firstly, we need to create HTML pages. front–end part of the web page is created which will accept the input from the user and predict the output values. As the starting page we have index.html. This page will redirect to the predict page and display based on the values predicted.

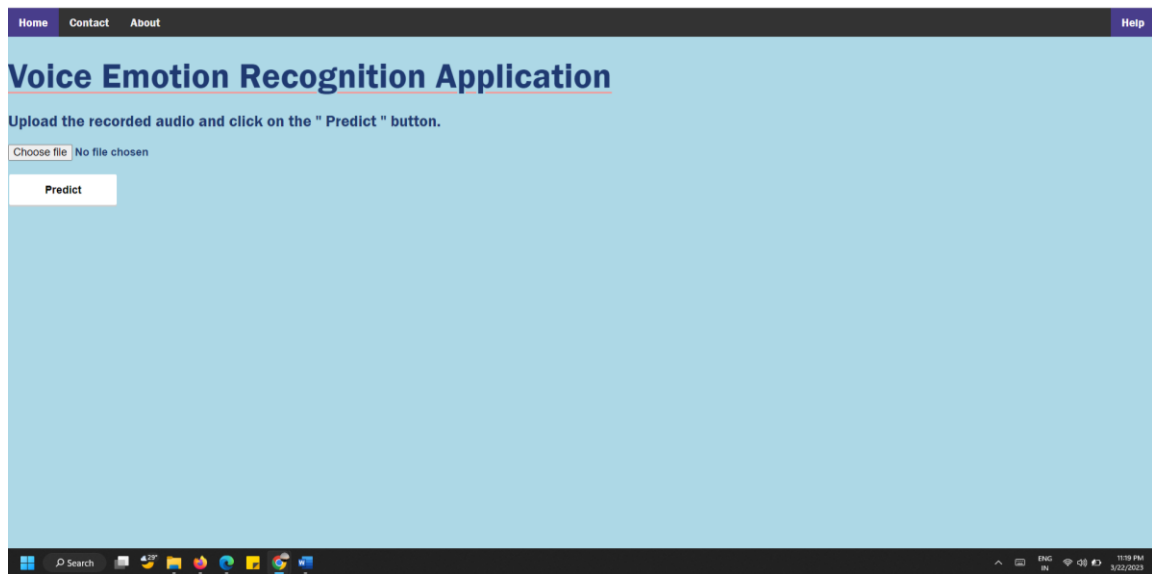


Fig 3.8.1 – HTML index page

The features are extracted from the audio file and encoded using one hot encoder and passed to the model for prediction. The model produces the prediction. The prediction is in the form of probabilities of each class and the maximum probability class is obtained through the argmax function. The class is obtained using the inverse transform function. Then the right classification label is presented on the prediction page.

CHAPTER – 4

RESULTS AND DISCUSSIONS

4.1 Model Evaluation

We perform Accuracy testing using the train data. The y_{test} values and predicted y values are used and the accuracy is calculated. This can be done by importing the metrics module from sklearn.

```
186/186 [=====] - 20s 105ms/step - loss: 1.0040 - accuracy: 0.7002  
Accuracy of our model on test data : 70.01680731773376 %
```

Fig 4.1.1 test data accuracy

The model metrics are visualized using the training and validation data. The loss and accuracy are plotted.

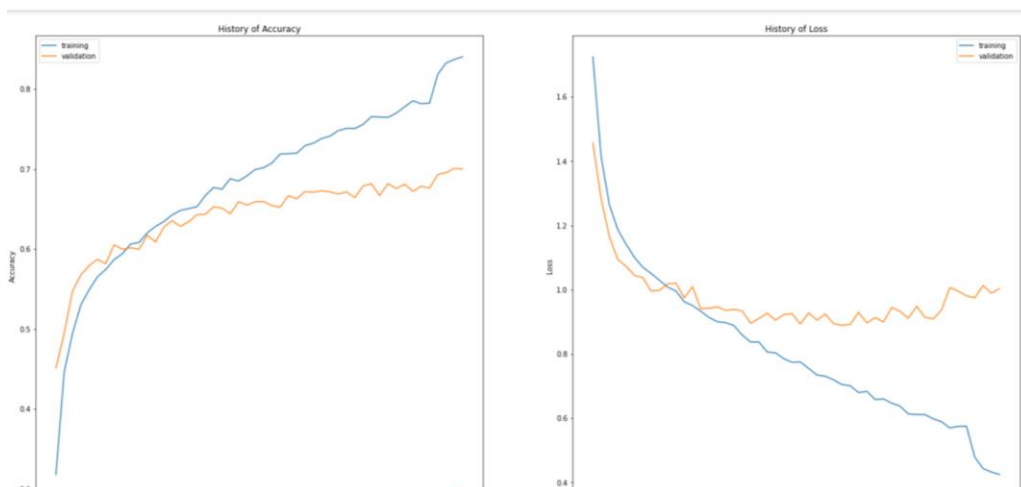


Fig 4.1.2 – History loss and accuracy

Then we print the classification report of each model using:

```
sklearn.metrics.classifiaction_report(y_pred,y_test)
```

	precision	recall	f1-score	support
Angry	0.86	0.71	0.77	936
Calm	0.63	0.82	0.71	164
Disgust	0.64	0.61	0.63	889
Fear	0.70	0.66	0.68	852
Happy	0.67	0.66	0.67	921
Neutral	0.66	0.68	0.67	795
Sad	0.62	0.76	0.68	896
Surprise	0.86	0.85	0.85	497
accuracy			0.70	5950
macro avg	0.71	0.72	0.71	5950
weighted avg	0.71	0.70	0.70	5950

Fig 4.1.3 – Classification report of the model.

Classification report is used to measure the quality of predictions from a classification algorithm used to build the model. This report shows the main classification metrics which are precision, recall and f1-score for each class. Some of the terms used are:

- a. TP – True Positives:** when an instance was negative and predicted negative
- b. FP – False Positives:** when an instance was positive and predicted positive
- c. FN – False Negatives:** when an instance was positive but predicted negative
- d. FP – False Positives:** when an instance was negative but predicted positive

- I. *Precision* is the capacity of a classifier to ignore instances that are negative when classifying them as positive. It is described for each class as the proportion of true positives to the total of true and false positives.

Precision – Accuracy of positive predictions.

$$\text{Precision} = TP / (TP + FP)$$

- II. *Recall* is defined as the ability of a classifier algorithm to find all positive instances. It is described as the proportion of true positives to the total of true positives and false negatives for each class.

$$\text{Recall} = TP / (TP + FN)$$

- III. *F₁ score* The *F₁ score*, typically ranges from 0.0 to 1.0 depending on recall and precision, is a weighted harmonic mean of those two metrics. 0.0 is considered as lowest score and 1.0 is considered as the highest score. Since *F₁* scores incorporate precision and recall into their computation, they are less accurate than accuracy measures.

$$\text{F1 Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

These scores give a detailed overview of the accuracy rate of each class labels.

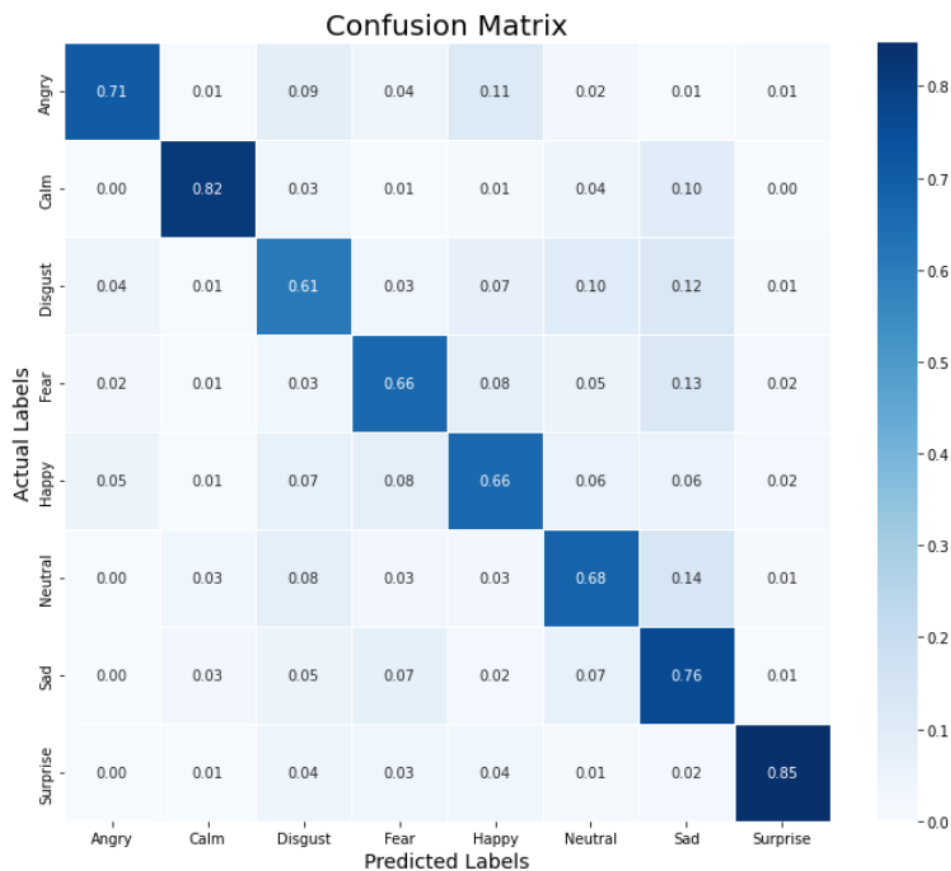


Fig 4.1.4 – Confusion Matrix of each label

4.2 RUNNING THE APP

1. To run the flask application, anaconda prompt is opened from the start menu.
2. Install the packages required using the pip install *packagename* command.
3. The path of the folder where the program is saved is navigated.
4. Python app.py command is used to run the flask application.
5. The link is copied and pasted onto the web browser.

```
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [23/Mar/2023 00:45:27] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [23/Mar/2023 00:45:27] "GET /static/style/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [23/Mar/2023 00:45:27] "GET /static/style/navigation%20bar.css HTTP/1.1" 304 -
[43804:0323/004533.012:ERROR:gpu_init.cc(481)] Passthrough is not supported, GL is disabled, ANGLE is
file uploaded successfully
1/1 [=====] - 1s 508ms/step
127.0.0.1 - - [23/Mar/2023 00:45:47] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [23/Mar/2023 00:45:48] "GET /static/style/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [23/Mar/2023 00:45:48] "GET /static/style/navigation%20bar.css HTTP/1.1" 304 -
```

Fig 4.2.1 – Running the flask application

After pasting the link in the browser, it will display the form index page. The user is supposed to upload an audio file and click on predict button. It will redirect to the prediction page based on the prediction output. The emotion of the audio will be displayed on the web page screen.

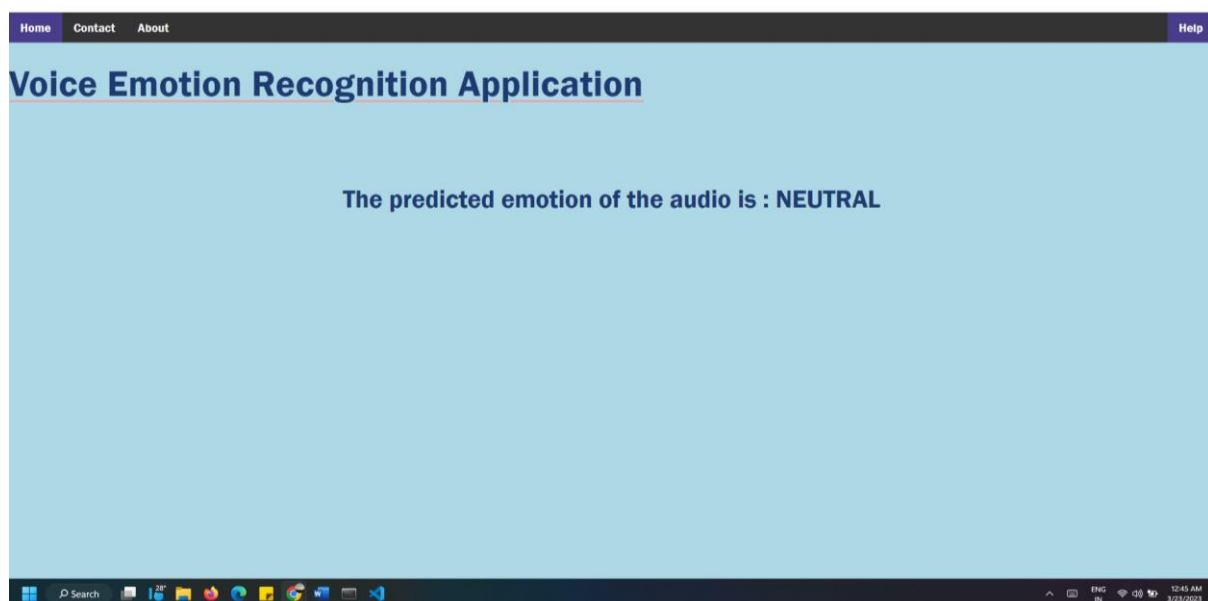


Fig 4.2.2 – Prediction page.

CHAPTER – 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

A deep learning model has been created for recognizing and predicting the emotion of the audio file which is in .wav format. This has been developed using the Convolutional neural network model with an accuracy score of 70.002 indicating a 70% accurate classification of emotions. The model is then integrated with the flask application. This model can be used to get some insights into the human expression of emotion through voice. This system can be employed in a variety of setups like Call centers for complaints or marketing, voice-based virtual assistants or chatbots, linguistic research, etc.

5.2 FUTURE WORK

The accuracy can be improved by applying more augmentation techniques and using other feature extraction methods. An accurate implementation of the pace of the speaking can be explored to check if it can resolve some of the deficiencies of the model. Figuring out a way to clear random silence from the audio clip. Following a lexical features-based approach towards SER and using an ensemble of the lexical and acoustic models. This will improve the accuracy of the system because in some cases the expression of emotion is contextual rather than vocal. Adding more data volume either by finding more annotated audio clips. This model can be further integrated with recommendation systems or chatbots to improve their services and achieve customer satisfaction.

REFERENCES

RESEARCH PAPERS

- [1] M. Wöllmer, A. Metallinou, F. Eyben, B. Schuller, and S. Narayanan, "Context-sensitive multimodal emotion recognition from speech and facial expression using bidirectional LSTM modeling," in Proc. INTERSPEECH, Makuhari, Japan, 2010, pp. 2362–2365.
- [2] Y. Niu, D. Zou, Y. Niu, Z. He, and H. Tan, "A breakthrough in speech emotion recognition using deep retinal convolution neural networks," 2017, arXiv:1707.09917. [Online]. Available: <https://arxiv.org/abs/1707.09917>
- [3] Y. Bengio, "Learning deep architectures for AI," Found. Trends Mach. Learn., vol. 2, no. 1, pp. 1–127, 2009.
- [4] M. W. Bhatti, Y. Wang, and L. Guan, "A neural network approach for human emotion recognition in speech," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), vol. 2, May 2004, p. II-81.
- [5] W. Q. Zheng, J. S. Yu, and Y. X. Zou, "An experimental study of speech emotion recognition based on deep convolutional neural networks," in Proc. Int. Conf. Affect. Comput. Intell. Interact. (ACII), Sep. 2015, pp. 827–831.
- [6] Y. Kim, "Convolutional neural networks for sentence classification," 2014, arXiv:1408.5882. [Online]. Available: <https://arxiv.org/abs/1408.5882>
- [7] J. G. Rázuri, D. Sundgren, R. Rahmani, A. Moran, I. Bonet, and A. Larsson, "Speech emotion recognition in emotional feedback for human-robot interaction," Int. J. Adv. Res. Artif. Intell., vol. 4, no. 2, pp. 20–27, 2015.
- [8] W. Q. Zheng, J. S. Yu, and Y. X. Zou, "An experimental study of speech emotion recognition based on deep convolutional neural networks," in Proc. IEEE Int. Conf. Affect. Comput. Intell. Interact., Sep. 2015, pp. 827–831.

INTERNET ARTICLES

- <https://www.projectpro.io/article/speech-emotion-recognition-project-using-machine-learning/573>
- <https://medium.com/@draj0718/deploying-deep-learning-model-using-flask-api-810047f090ac>
- <https://analyticsarora.com/how-to-make-a-flask-web-app-for-keras-model/>
- <https://data-flair.training/blogs/python-mini-project-speech-emotion-recognition/>
- https://www.researchgate.net/publication/335360469_Speech_Emotion_Recognition_Using_Deep_Learning_Techniques_A_Review#:~:text=In%20the%20literature%20of%20speech,to%20traditional%20techniques%20in%20SER.
- [https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9#:~:text=1\),%2D%20to%20high%2Dlevel%20patterns.](https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9#:~:text=1),%2D%20to%20high%2Dlevel%20patterns.)

APPENDICES

A. SAMPLE CODE

```
B. from tensorflow.keras.models import Sequential
C. from tensorflow.keras.losses import categorical_crossentropy
D. from tensorflow.keras.losses import sparse_categorical_crossentropy
E.
F. model=Sequential()
G. model.add(Conv1D(256, kernel_size=5, strides=1, padding='same',
    activation='relu', input_shape=(x_train.shape[1], 1)))
H. model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
I.
J. model.add(Conv1D(256, kernel_size=5, strides=1, padding='same',
    activation='relu'))
K. model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
L.
M. model.add(Conv1D(128, kernel_size=5, strides=1, padding='same',
    activation='relu'))
N. model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
O. model.add(Dropout(0.2))
P.
Q. model.add(Conv1D(64, kernel_size=5, strides=1, padding='same',
    activation='relu'))
R. model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
S.
T. model.add(Flatten())
U. model.add(Dense(units=32, activation='relu'))
V. model.add(Dropout(0.3))
W.
X. model.add(Dense(units=8, activation='softmax'))
Y. model.compile(optimizer = 'adam' , loss = 'categorical_crossentropy' , metrics =
    ['accuracy'])
Z.
AA. model.summary()
```

FLASK APPLICATION CODE

```
app = Flask(__name__)
config = tf.compat.v1.ConfigProto(
    intra_op_parallelism_threads=1,
    allow_soft_placement=True
)
session = tf.compat.v1.Session(config=config)

#files=os.listdir(os.curdir)
#print(files)
```

```

tf.compat.v1.keras.backend.set_session(session)

model=load_model('./CNN-Model.h5', compile =False)
model.make_predict_function()

e=OneHotEncoder()
with open('encoder.pickle', 'rb') as f:
    e= pickle.load(f)

@app.route('/Prediction')
def Prediction():
    return render_template('Prediction.html')

@app.route('/')
def index_view():
    return render_template('index.html')

def extract_features_and_predict(path):
    filepath='tmp' + path
    features=np.empty((0,162))
    x,sr =librosa.load(filepath)
    zcr = np.mean(librosa.feature.zero_crossing_rate(x).T,axis=0)

    stft=np.abs(librosa.stft(x))
    chroma_stft = np.mean(librosa.feature.chroma_stft(S=stft, sr=sr).T,axis=0)

    mfcc = np.mean(librosa.feature.mfcc(y=x, sr=sr).T, axis=0)

    rmse = np.mean(librosa.feature.rms(y=x).T, axis=0)

    mel = np.mean(librosa.feature.melspectrogram(y=x,sr=sr).T, axis=0)
    ext_features=np.hstack([zcr,chroma_stft,mfcc,rmse,mel])
    features=np.vstack([features,ext_features])
    x=features
    x=x.reshape(x.shape[1],x.shape[0])
    res_x=x
    preds = model_predict(res_x, model)

    os.remove(filepath)
    return preds

def model_predict(vector, model):
    try:
        with session.as_default():
            #with session.graph.as_default():
                preds = model.predict(np.array([vector]))

```

```

        pred= e.inverse_transform(preds)
        return pred

except Exception as ex:
    print(ex)

@app.route('/predict',methods=["POST","GET"])
def predict():
    if request.method == "POST":
        f = request.files['audio_data']
        random_number = random.randint(00000, 99999)
        filepath='./tmp'+str(random_number)+ '.wav'
        f.save(filepath)
        print('file uploaded successfully')

        filename=str(random_number) +'.wav'
        res = extract_features_and_predict(filename)

        if(res=='Calm'):
            return render_template("Prediction.html", prediction_text="The predicted
emotion of the audio is : CALM ")
        elif(res=='Neutral'):
            return render_template("Prediction.html", prediction_text="The predicted
emotion of the audio is : NEUTRAL ")
        elif(res=='Angry'):
            return render_template("Prediction.html", prediction_text="The predicted
emotion of the audio is : ANGRY ")
        elif(res=='Happy'):
            return render_template("Prediction.html", prediction_text="The predicted
emotion of the audio is : HAPPY ")
        elif(res=='Sad'):
            return render_template("Prediction.html", prediction_text="The predicted
emotion of the audio is : SAD ")
        elif(res=='Disgust'):
            return render_template("Prediction.html", prediction_text="The predicted
emotion of the audio is : DISGUST ")
        elif(res=='Surprise'):
            return render_template("Prediction.html", prediction_text="The predicted
emotion of the audio is : SURPRISE ")
        elif(res=='Fear'):
            return render_template("Prediction.html", prediction_text="The predicted
emotion of the audio is : FEAR ")
        else:
            return render_template("index.html")
if __name__ == "__main__":
    port=int(os.environ.get('PORT',5000))
    app.run(port=port,debug=True,use_reloader=False)

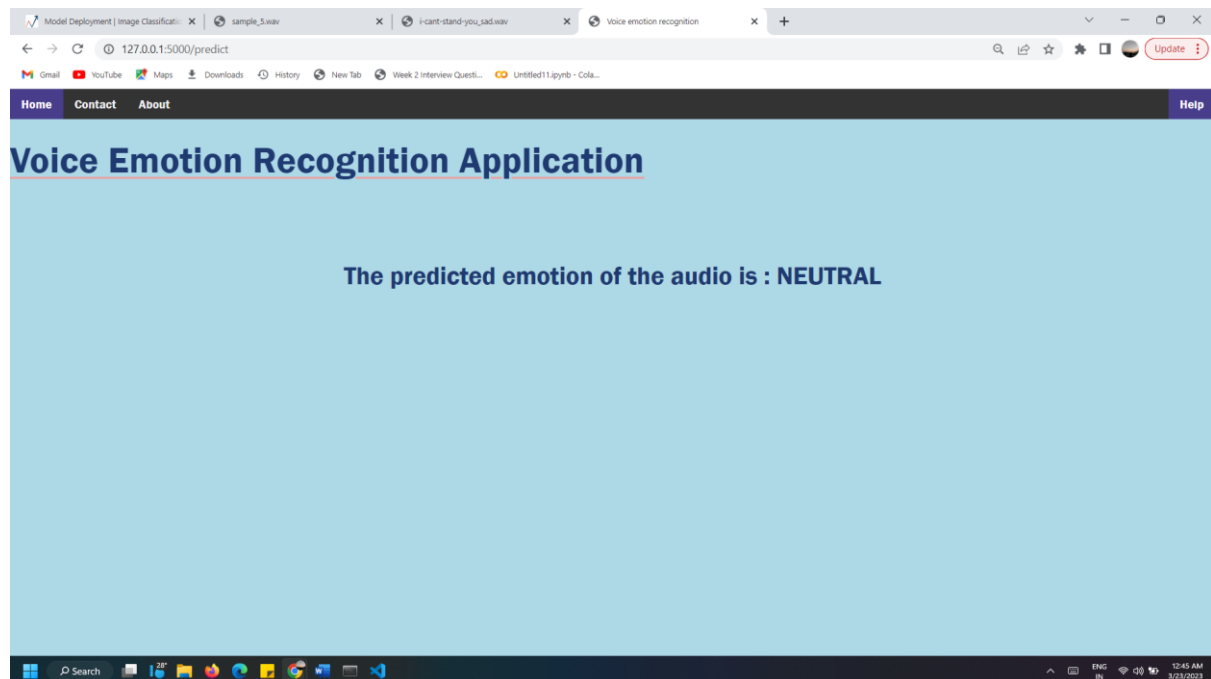
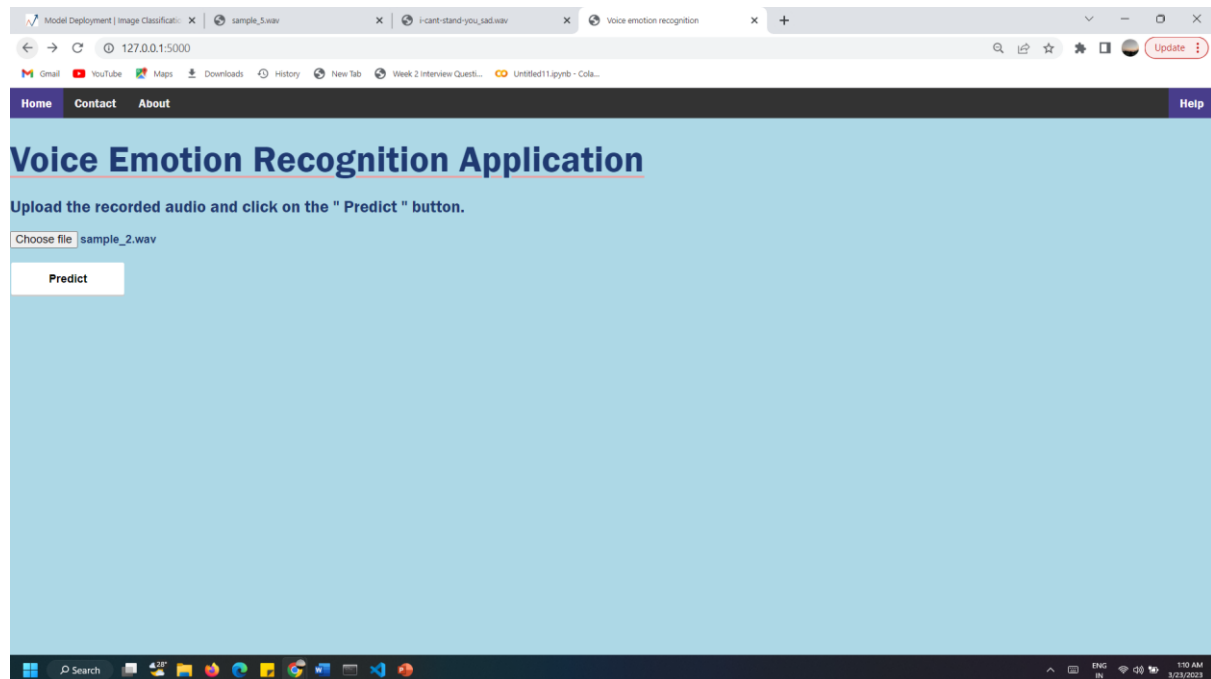
```

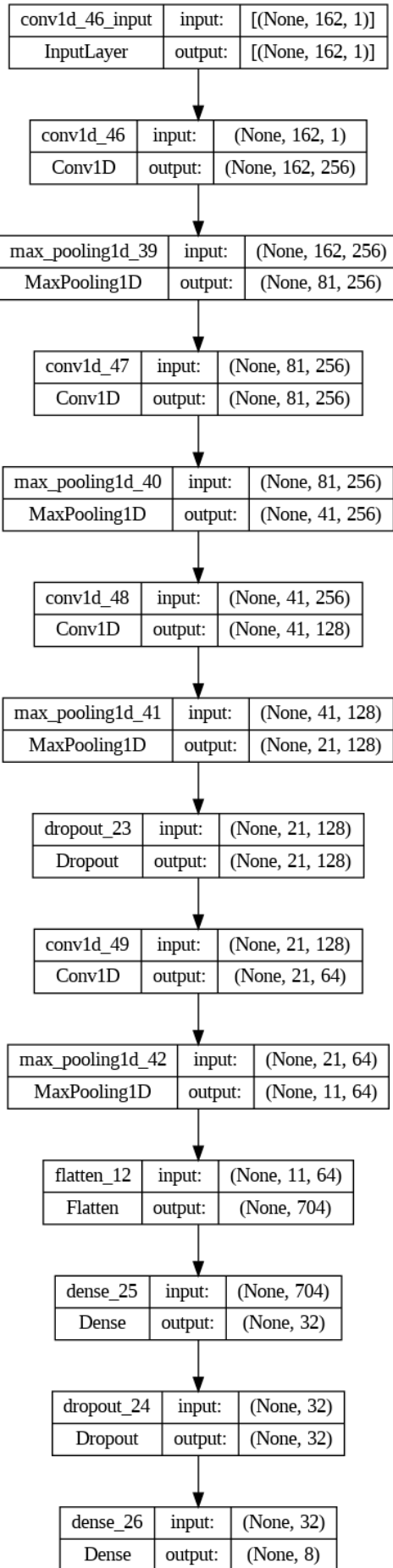
B SCREEN SHOTS

```

Anaconda Powershell Prompt (anaconda)
(base) PS C:\Users\reeya\PI2\SE8_cop2> app.py
(base) PS C:\Users\reeya\PI2\SE8_cop2>
(base) PS C:\Users\reeya\PI2\SE8_cop2> ^tate idle
(base) PS C:\Users\reeya\PI2\SE8_cop2> ^cess id: 1, type: extensionhost; pid: [none]; creating new...
(base) PS C:\Users\reeya\PI2\SE8_cop2> ^canda activate tf_tensionhost; pid: 1564; successfully created
(tf) PS C:\Users\reeya\PI2\SE8_cop2> app.py
(tf) PS C:\Users\reeya\PI2\SE8_cop2> ^tate idle
(tf) PS C:\Users\reeya\PI2\SE8_cop2> python app.py
2023-03-23 00:45:14.395840: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:
AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-03-23 00:45:16.000297: I tensorflow/core/platform/process_util.cc:146] Creating new thread pool with default inter op setting: 8. Tune using inter_op_parallelism_threads for best performance.
WARNING:tensorflow: From C:\Users\reeya\PI2\SE8_cop2\app.py:29: The name tf.keras.backend.set_session is deprecated. Please use tf.compat.v1.keras.backend.set_session instead.
2023-03-23 00:45:16.104497: I tensorflow/core/common_runtime/process_util.cc:146] Creating new thread pool with default inter op setting: 2. Tune using inter_op_parallelism_threads for best performance.
* Serving Flask app "app"
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production process. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - [23/Mar/2023 00:45:27] "GET / HTTP/1.1" 200 -
127.0.0.1 - [23/Mar/2023 00:45:27] "GET /static/style/styles.css HTTP/1.1" 304 -
127.0.0.1 - [23/Mar/2023 00:45:27] "GET /static/style/navigation300bar.css HTTP/1.1" 304 -
[43084.0322/00453].012[ERROR:log_int.cc(481)] Passthrough is not supported, GL is disabled, ANGLE is
file uploaded successfully
127.0.0.1 - [23/Mar/2023 00:45:27] "POST / HTTP/1.1" 200 -
127.0.0.1 - [23/Mar/2023 00:45:43] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - [23/Mar/2023 00:45:48] "GET /static/style/styles.css HTTP/1.1" 304 -
127.0.0.1 - [23/Mar/2023 00:45:48] "GET /static/style/navigation300bar.css HTTP/1.1" 304 -

```





File Home Insert Draw Page Layout Formulas Data Review View Help Te										
Clipboard		Font		Alignment						
A1		Emotions								
A	B	C	D	E	F	G	H	I	J	
1	Emotions	Path								
2	Calm	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-02-02-01-01-10.wav								
3	Calm	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-02-01-01-02-10.wav								
4	Calm	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-02-02-01-02-10.wav								
5	Calm	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-02-01-01-01-10.wav								
6	Calm	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-02-02-02-01-10.wav								
7	Neutral	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-01-01-02-02-10.wav								
8	Neutral	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-01-01-01-02-10.wav								
9	Neutral	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-01-01-01-01-10.wav								
10	Neutral	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-01-01-02-01-10.wav								
11	Calm	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-02-01-02-02-10.wav								
12	Calm	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-02-01-02-01-10.wav								
13	Angry	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-05-01-02-01-10.wav								
14	Happy	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-03-02-02-01-10.wav								
15	Happy	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-03-02-01-02-10.wav								
16	Angry	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-05-01-01-01-10.wav								
17	Sad	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-04-02-02-02-10.wav								
18	Angry	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-05-01-01-02-10.wav								
19	Sad	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-04-02-02-01-10.wav								
20	Sad	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-04-02-01-01-10.wav								
21	Happy	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-03-01-01-01-10.wav								
22	Sad	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-04-02-01-02-10.wav								
23	Calm	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-02-02-02-02-10.wav								
24	Sad	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-04-01-01-01-10.wav								
25	Sad	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-04-01-02-01-10.wav								
26	Happy	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-03-01-02-01-10.wav								
27	Sad	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-04-01-01-02-10.wav								
28	Happy	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-03-01-02-02-10.wav								
29	Sad	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-04-01-02-02-10.wav								
30	Happy	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-03-02-01-01-10.wav								
31	Happy	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-03-01-01-02-10.wav								
32	Happy	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-03-02-02-02-10.wav								
33	Disgust	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-07-01-02-01-10.wav								
34	Disgust	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-07-02-01-02-10.wav								
35	Surprise	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-08-01-01-01-10.wav								
36	Fear	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-06-01-02-01-10.wav								
37	Fear	/content/drive/MyDrive/SER/SER_Daata/RAVDESS/Actor_10/03-01-06-02-01-01-10.wav								