2 Introduction

2.1 Aim

The goal of this project is to build MLP, LSTM and CNN Model for speech emotion recognition using Ravdess Emotional Speech Audio dataset.

2.2 Abstract

Speech Emotion Recognition, abbreviated as SER, is the act of attempting to recognize human emotion and the associated affective states from speech. This is capitalizing on the fact that voice often reflects underlying emotion through tone and pitch. Emotion recognition is a rapidly growing research domain in recent years. Unlike humans, machines lack the abilities to perceive and show emotions. But human-computer interaction can be improved by implementing automated emotion recognition, thereby reducing the need for human intervention. In this project, basic emotions like calm, happy, fearful, disgust etc. are analyzed from emotional speech signals.

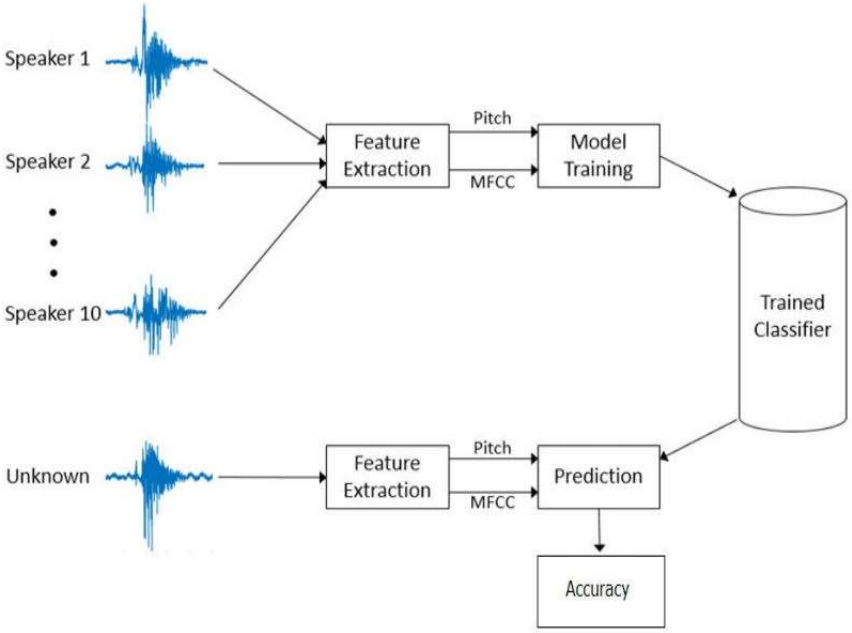


Figure 1: Speech Emotion Recognition System Flowchart

3 Brief Approach

Using RAVDESS dataset which contains around 1440 audio file inputs from 24 different actors (12 male and 12 female),First, we analyzed our dataset by plotting the spectrogram and waveforms of a sample audio file from the dataset. Then we extracted the features of the data using Mel Frequency Cepstral Coefficents (MFCCs), chroma frequencies, MeI spectrogram and Spectral centroid. Then we build a Multi-layer perceptron (MLP) model to predict the emotions of the audio data as well as the gender of the speaker. We plan to improve upon our performance by using LSTM model and CNN model, compare the performances of the three and establish which model is the best for Speech Emotion Recognition (SER).

4 Dataset

This section gives a detailed description of RAVDESS dataset used in this project.

Files
This portion of the RAVDESS contains 1440 files: 60 trials per actor x 24 actors = 1440. The RAVDESS contains 24 professional actors (12 female, 12 male), vocalizing two lexically-matched statements in a neutral North American accent. Speech emotions includes calm, happy, sad, angry, fearful, surprise, and disgust expressions. Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression.

File naming convention
Each of the 1440 files has a unique filename. The filename consists of a 7-part numerical identifier (e.g., 03-01-06-01-02-01-12.wav). These identifiers define the stimulus characteristics:

Filename identifiers

- 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).

Filename example: 03-01-06-01-02-01-12.wav

- Audio-only (03)
- Speech (01)
- Fearful (06)
- Normal intensity (01)
- Statement "dogs" (02)
- 1st Repetition (01)
- 12th Actor (12)

Link for RAVDESS dataset is here

5 Feature Extraction

Feature extraction is very effective in increasing the accuracy and efficiency of machine learning algorithms in emotional speech recognition. The input audio file is reduced to a set of features which represent or summarise the original input. The extracted features are then fed into the neural network to identify the respective emotion. The features which have been used in the project are discussed below.

5.1 Mel Spectrogram.

An audio signal can be broken down into sine and cosine waves that form the original signal. The frequencies and amplitudes of these representative waves can be used to convert the input signal from the time to the frequency domain. The fast Fourier transform (FFT) is an algorithm that can be used to perform this conversion. The FFT is however performed on a single time window of the input signal. If the frequencies of the representative signals change over time (non-periodic), the change cannot be captured through a single time window conversion. Using the FFT over multiple overlapping windows can be used to construct a spectrogram representing the amplitude of the representative frequencies as they change over time.

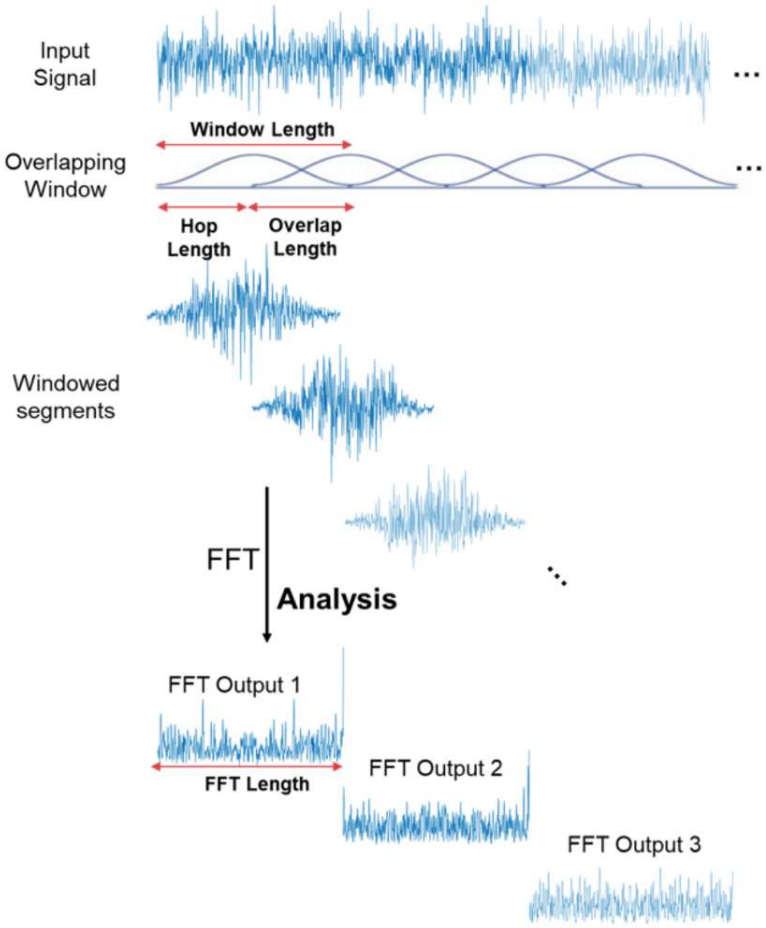


Figure 2: Performing FFT on multiple windows of the input signal.

5.2 Mel-frequency cepstral coefficients (MFCCs).

The Fourier spectrum is obtained by using the FFT on an audio signal. Taking the log of the Fourier spectrum's magnitude and then taking the spectrum of this log through a cosine transformation allows us to calculate the cepstrum of the signal (ceps is the reverse of spec, called so due to being a non-linear 'spectrum of a spectrum'). The amplitudes of this resulting cepstrum are the cepstral coefficients. Representing the frequencies in terms of the mel scale (discussed above) instead of a linear scale converts the cepstrum to a mel-frequency cepstrum and the cepstral coefficients to mel-frequency cepstral coefficients (MFCCs). The cepstrum represents the rate of change in the different spectrum bands, and the coefficients are widely used in machine learning algorithms for sound processing.

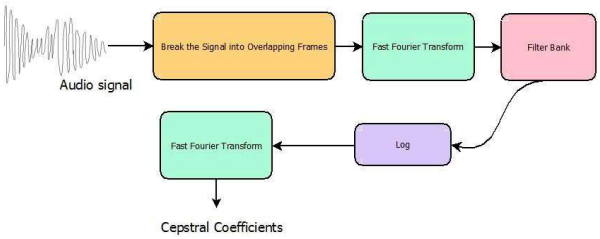


Figure 3: Process of extracting MFCCs. The filter bank represents the mel-scale conversion.

5.3 Chroma.

An octave is a musical interval or the distance between one note and another, which is twice its frequency: A3 (220 Hz) - A4 (440 Hz) – A5 (880 Hz). An octave is divided into 12 equal intervals, and each interval is a different note. The intervals are called chromas or semitones and are powerful representations of audio signals. The calculation of the chromagram is similar to the spectrogram using short-time Fourier transforms, but semitones/pitch classes are used instead of absolute frequencies to represent the signal. The unique representation can show musical properties of the signal which is not picked up by the spectrogram.

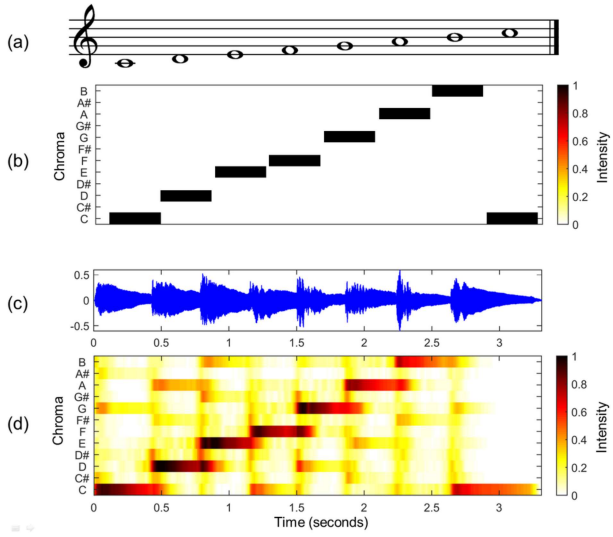


Figure 4: (a) Musical score of a C-major scale. (b) Chromagram obtained from the score. (c) Audio recording of the C-major scale played on a piano. (d) Chromagram obtained from the audio recording.

6 Implementation Details

The objective of the project was to classify the sound files into gender and emotion labels. Hence two separate models were used. One for classifying gender and another for classifying the emotion. Both the models have identical implementation besides the fact that in case of emotion there are 8 labels ranging from 0-7 and in case of gender classification only 0-1 labels are present.Below is the description of the emotion classifier model.

6.1 Taking inputs

There are 1440 files available in the dataset. Each file is approximately of 3 sec length. But while doing sampling at the rate of 22100 hz of each file, the number of samples in each file differs. In short, the length of each file is different by few micro or miliseconds and hence when they are loaded using librosa library in the form of a numpy array, the arrays have different lengths. To resolve this issue, padding is needed at ths end in case of shorter files. While taking inputs maximum file size was observed to be around 111k and hence a round off value of 120k was chosen. The padding was done with zeros i.e. zeros were appended at the end of each file to make them equal in length. The value zero is specifically chosen because it would mean that as if no voice is there. All the files are stored in a numpy array of shape (24*60,120000). Now coming to the y_train array, the emotion label of a file is denoted by the 6th and the 7th character of the file name itself. The labels vary from 1-8 but we store them after subtracting one from each of the labels. This is done because softmax function in mlp classifier predicts values ranging from 0-7.

```
path='../dataset/voices'
k=0
x_data=np.ndarray(shape=(24*60,120000))
y_data=np.ndarray(shape=(24*60,1))
for i,actor in enumerate(os.listdir(path)):
    path1=os.path.join(path,actor)
    for j,file in enumerate(os.listdir(path1)):
        b=file[6:8]
        y_data[k,0]=b
        y_data[k,0]=y_data[k,0]-1
        temp_path=os.path.join(path1,file)
        # print(temp_path)
        var,_=librosa.load(temp_path)
        x_data[k,:var.shape[0]]=var[:]
        x_data[k,var.shape[0]:]=0
        k=k+1
```

6.2 Feature extraction

The sound files cannot be directly fed into the mlp model and hence features are extracted from the sound files and then they are fed into the model to predict the label. 4 features were extracted for each of the 1440 files in the datset. Initially the four features namely mfcc, chroma frequency, mel frequency and spectral rolloff were used. Mfcc gave 40 features for each frame of a file while chroma frequency gave 12, spectral rollofg gave 1 feature and mel frequency gave 128 features for each frame of the file. Mean was taken for each of the features over all the

frames of a single file and then appended in an array. Spectral rolloff was found to have no significant influence on the accuracy, hence it was discarded in the final computation.

```
def extract_feature(file):
    result=np.array([])
    mfcc=np.mean(librosa.feature.mfcc(file,sr=sr,n_mfcc=40).T,axis=0)
    result=np.hstack((result,mfcc))
    chroma=np.mean(librosa.feature.chroma_stft(file,sr=sr).T,axis=0)
    result=np.hstack((result,chroma))
    mel=np.mean(librosa.feature.melspectrogram(file,sr=sr).T,axis=0)
    result=np.hstack((result,mel))
    spectral_rolloff = np.mean(librosa.feature.spectral_rolloff(file, sr=sr).T,axis=0)
    result=np.hstack((result,spectral_rolloff))
    return result
```

6.3 Model Architecture

A fully connected neural network was deployed to predict the emotion from the features of the sound files.The model consists of 4 hidden layers with a dropout of 0.1 in the first three layers. The first and the second hidden layers consists of 512 neurons while the third layer contained 128 neurons and the fourth hidden layer contained 64 neurons.The model was defined using the keras in-built module Sequential which takes input in the form of layers. 'Relu' activation function was used to calculate the values of parameters from one layer to another layer. A dropout of 0.1 was also added in case of first three hidden layers to avoid overfitting by dropping a node with a 10% probability.

```
def build_fc_model(input_shape=(180,1), num_class=8):


    model=tf.keras.models.Sequential([tf.keras.layers.Dropout(0.1),
                                      tf.keras.layers.Dense(512,activation='relu',input_shape=input_shape,kernel_regularizer=tf.keras.regul
arizers.l2(1e-2)),
                                      tf.keras.layers.Dropout(0.1),
                                      tf.keras.layers.Dense(512,activation='relu',kernel_regularizer=tf.keras.regularizers.l2(1e-3)),
                                      tf.keras.layers.Dropout(0.1),
                                      tf.keras.layers.Dense(128,activation='relu'),
                                      tf.keras.layers.Dense(64,activation='relu'),
                                      tf.keras.layers.Dense(num_class,activation='softmax')
                                      ])

    return model
```

In the first two layers 12 regularization was also deployed to overcome the case of overfitting and to effectively tune the parameters.The input layer is of the shape (180,1) and the ouput layer predicts the emotion using the 'softmax' function which gives the probability of each label and the label with the highest probability is selected and given as the answer.

6.4 Model training and prediction

After the feature extraction process, the data was divided into training and testing data with a test size of 0.2. The data was scaled between 0 and 1 for better performance of the model and then the data was fed into the model architecture. Sparse_categorical_crossentropy was used to calculate the loss and adam optimizer was used in the final compilation of the trained model. Then the model was tested on test dataset.

```
x=[]
for i in range(24*60):
    feature=extract_feature(x_data[i,:])
    x.append(feature)
x_train, x_test,y_train,y_test = train_test_split(np.array(x), y_data, test_size=0.2, random_state=9)
```

```
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

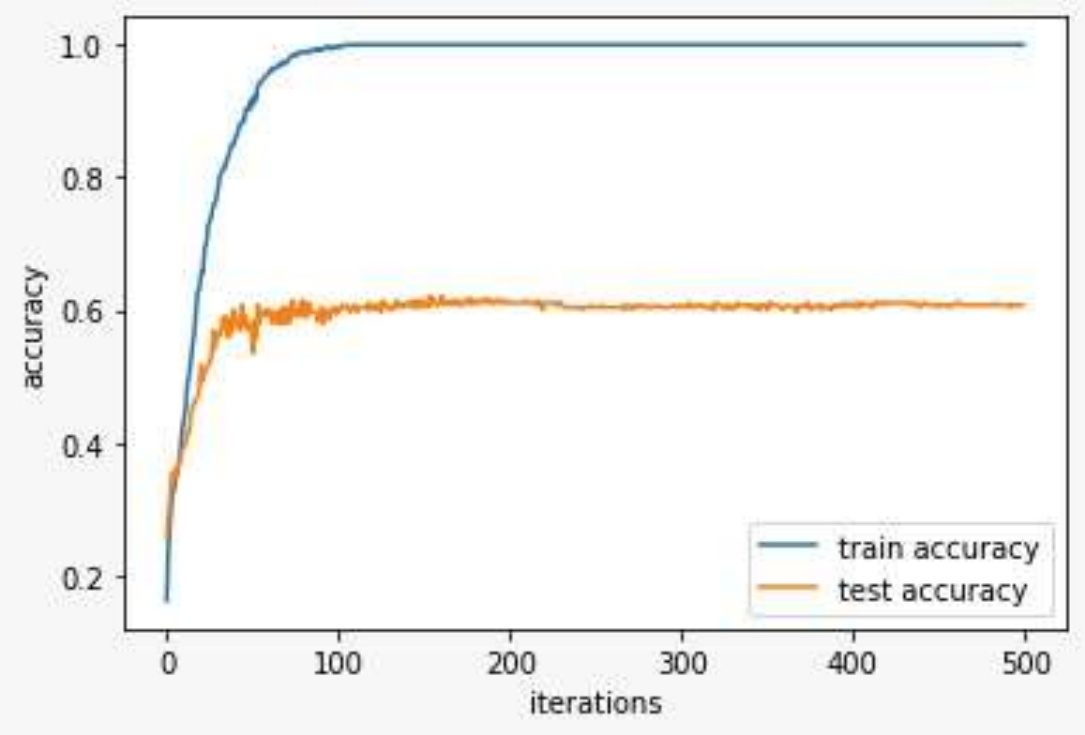
Github link for Team 1 is here.
Github link for Team 2 is here
Github link for Team 3 is here.
Github link for Team 4 is here.
Github link for Team 5 is here.

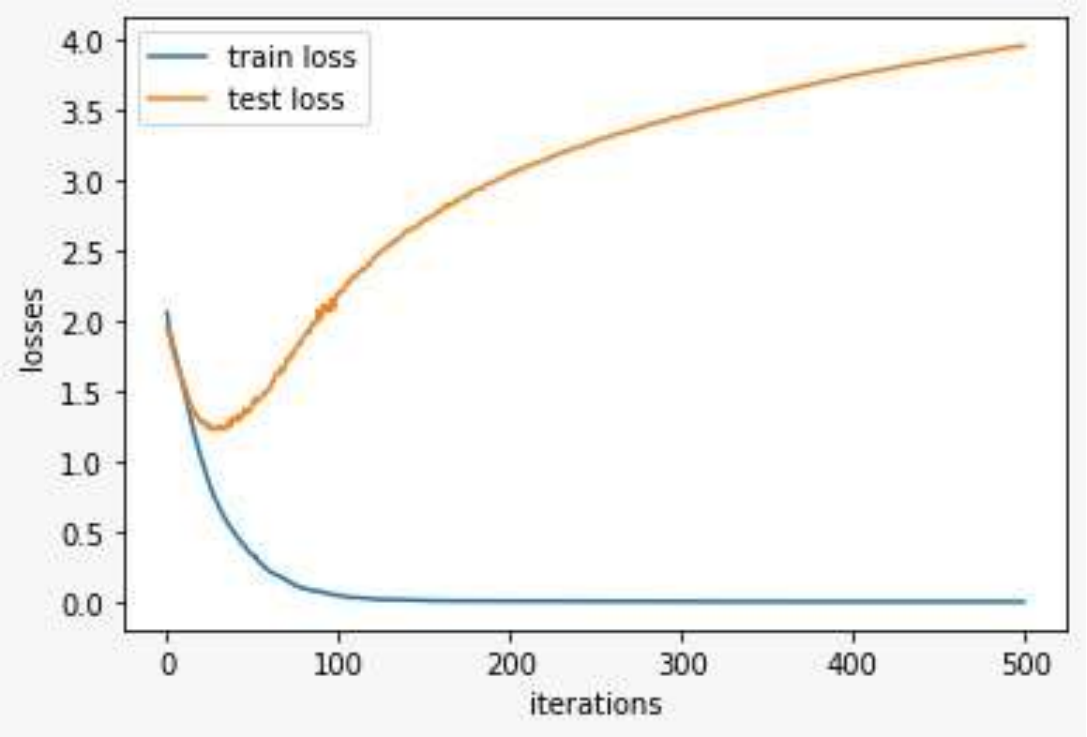
7 Results

7.1 TEAM - 1 Model Results

7.1.1 Emotion Recognition

Accuracy obtained on training data is 100 %
Accuracy obtained on test data is 62.15 %

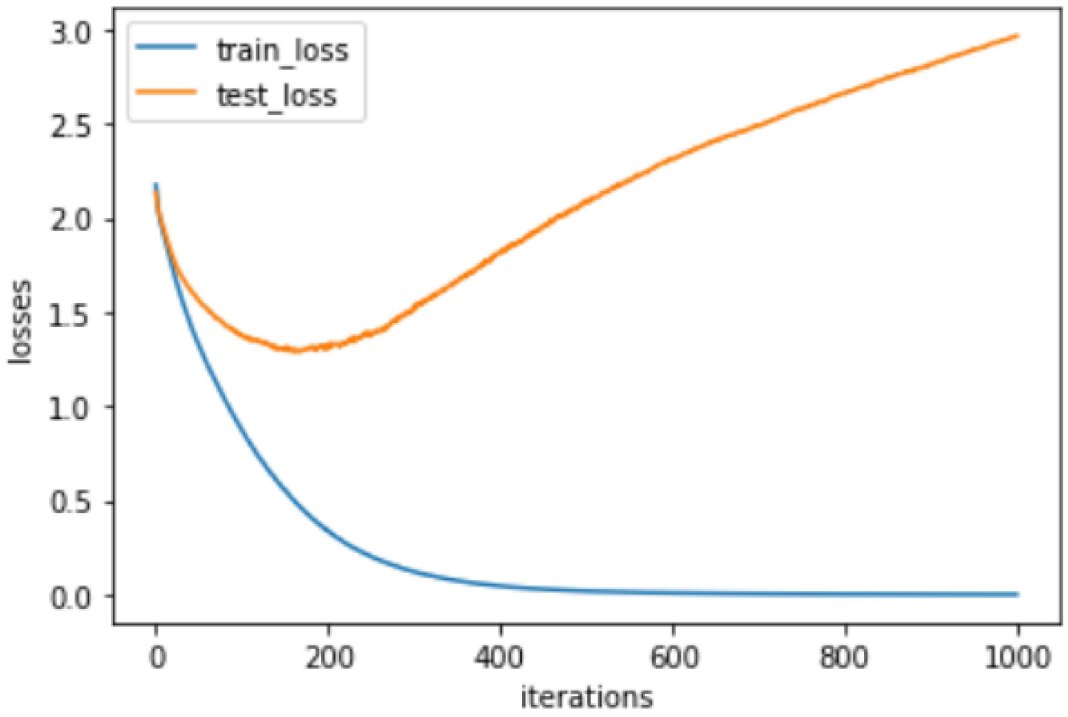


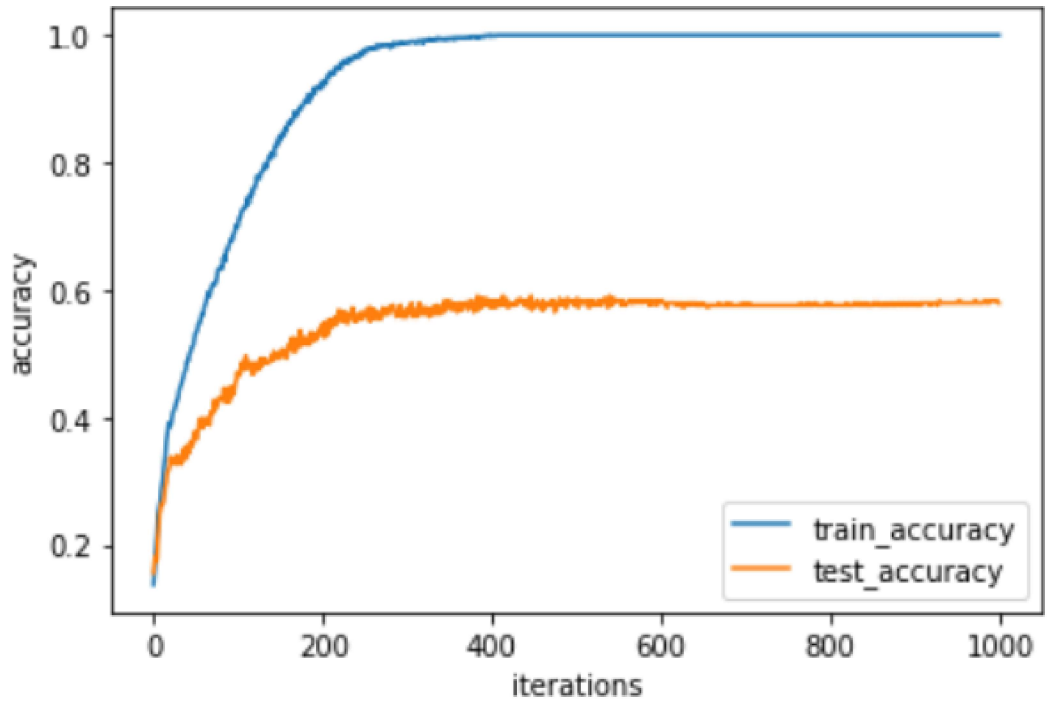

train loss
test loss
losses
iterations
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
100
200
300
400
500

7.2 TEAM - 2 Model Results

7.2.1 Emotion Recognition

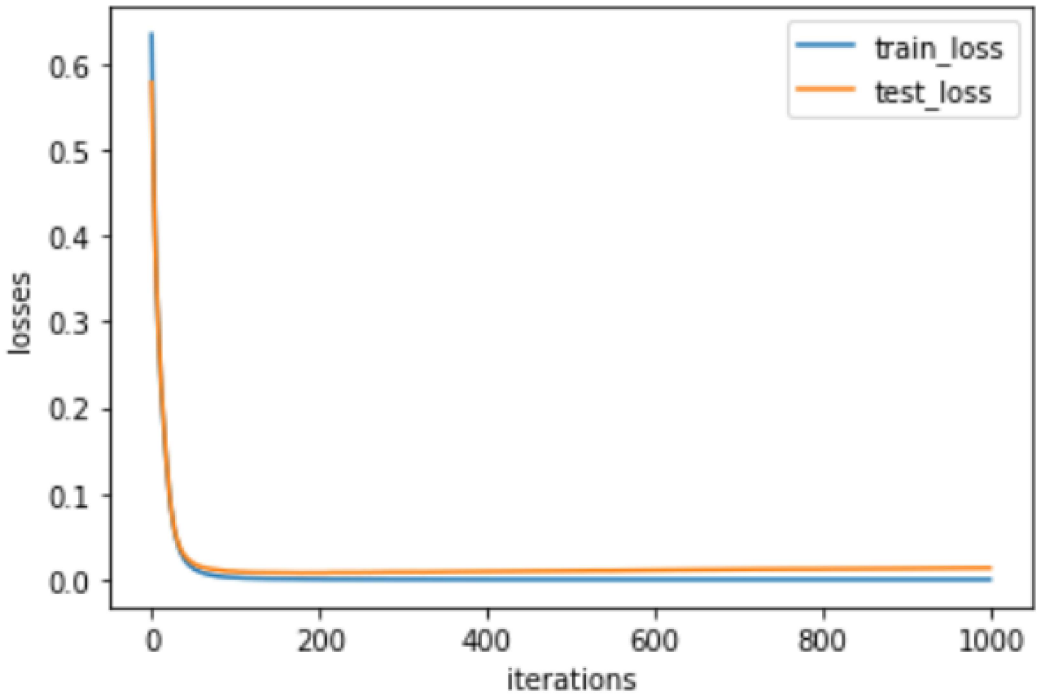
Accuracy obtained on training data is 100 %
Accuracy obtained on test data is 57.97 %

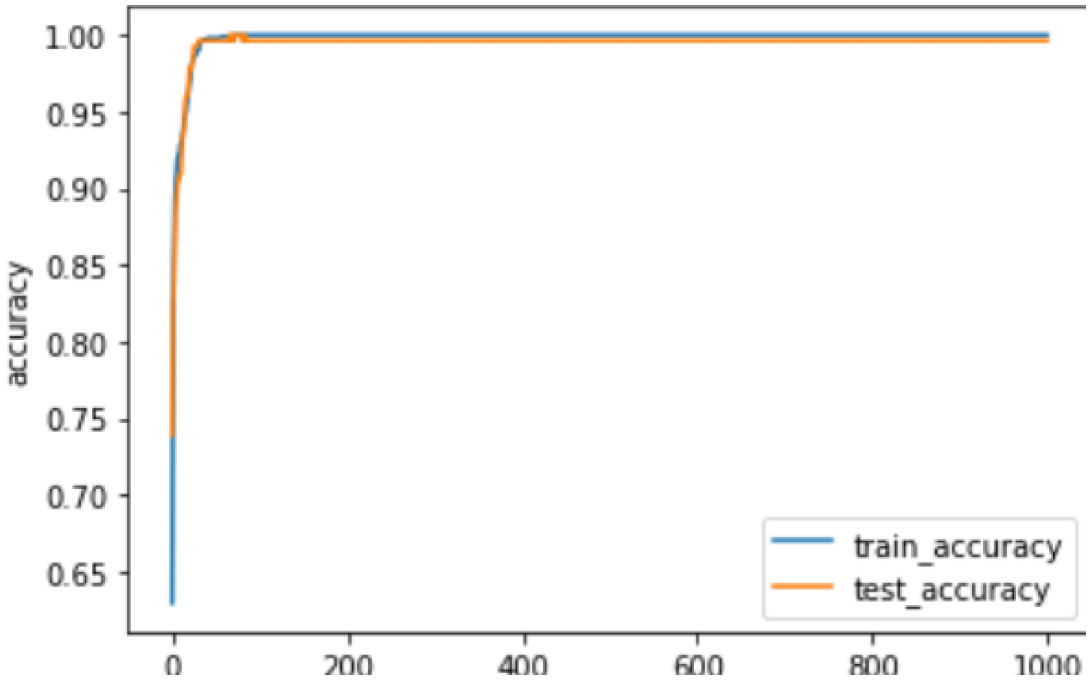




7.2.2 Gender Recognition

Accuracy obtained on training data is 100 %
Accuracy obtained on test data is 99.64 %





7.3 TEAM - 3 Model Results

7.3.1 Emotion Recognition

Accuracy obtained on training data is 93.24 %
Accuracy obtained on test data is 65.83 %

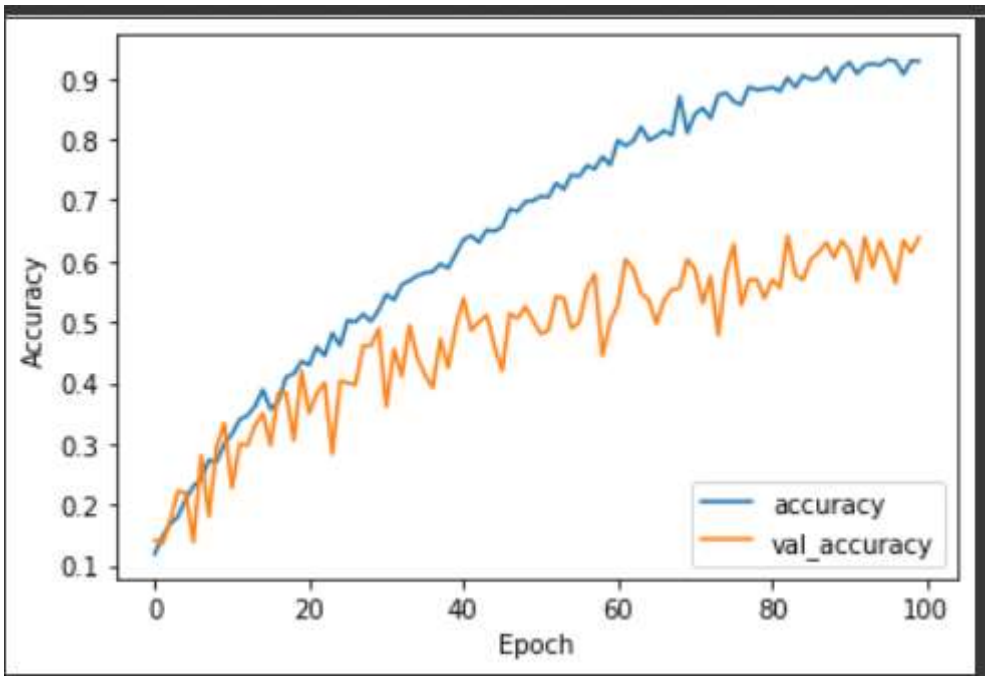


Figure 5: Accuracy vs Epoch

7.3.2 Confusion Matrix

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
y_pred=model.predict(x_test)
y_pred=np.argmax(y_pred,axis=1)
labels=('neutral','calm','happy','sad','angry','fear','disgust','suprise')
cm=confusion_matrix(y_test,y_pred,normalize='true')
fig, axis = plt.subplots(figsize=(10, 10))
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=labels)
disp.plot(ax=axis)
```

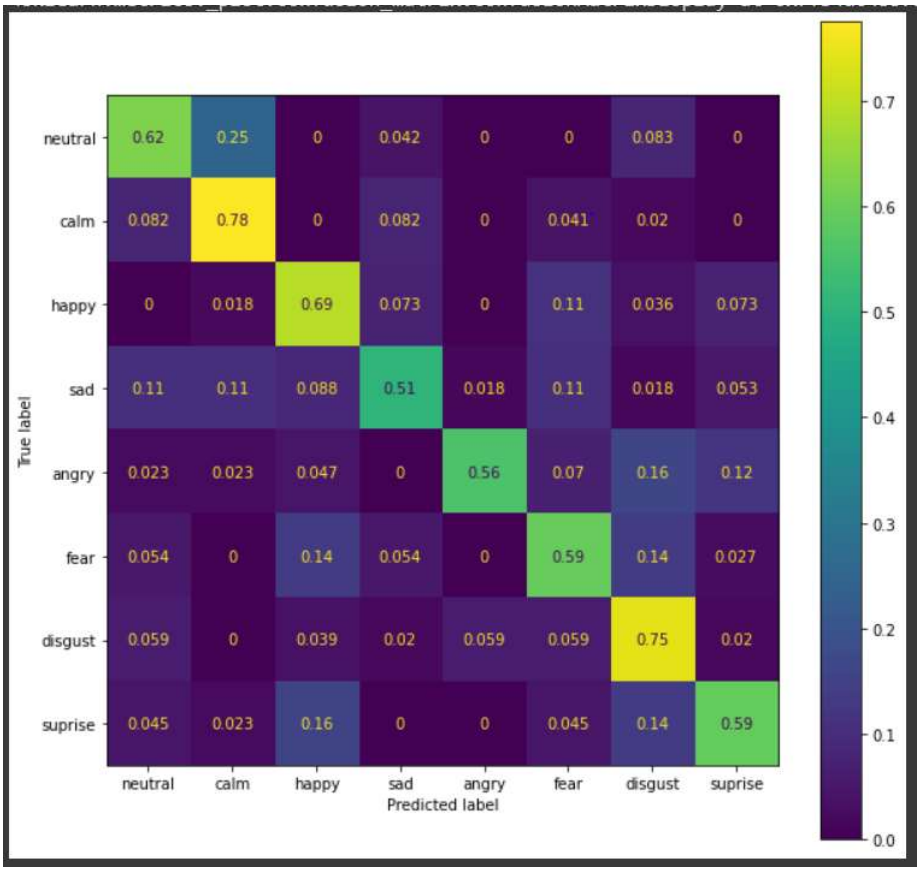
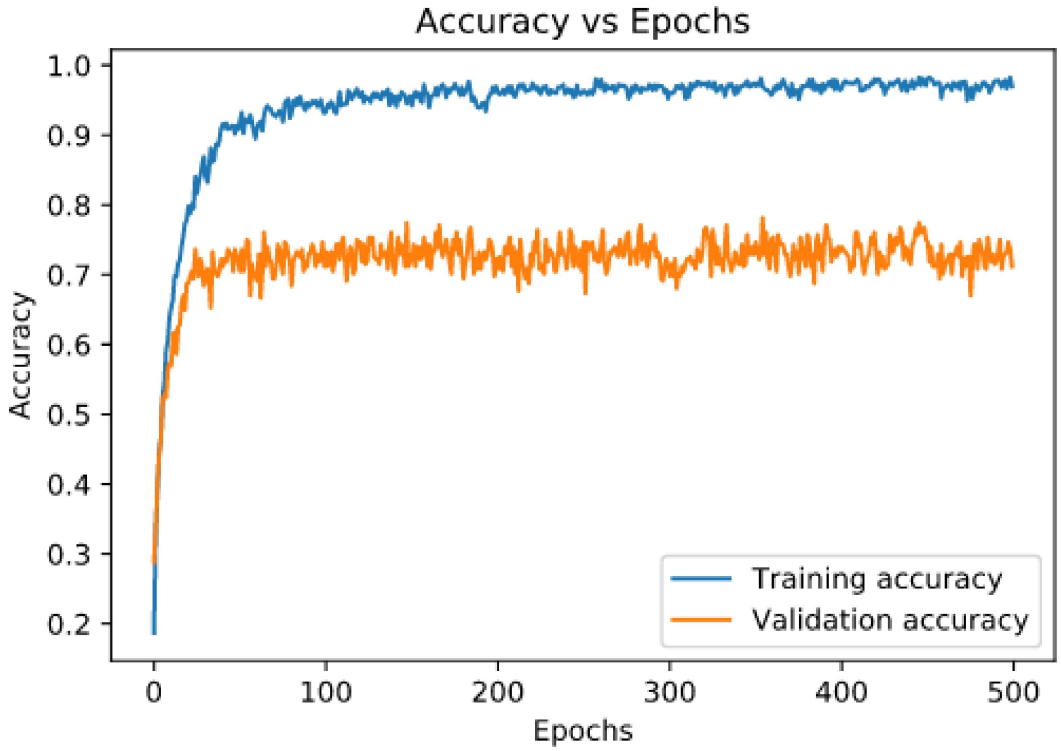


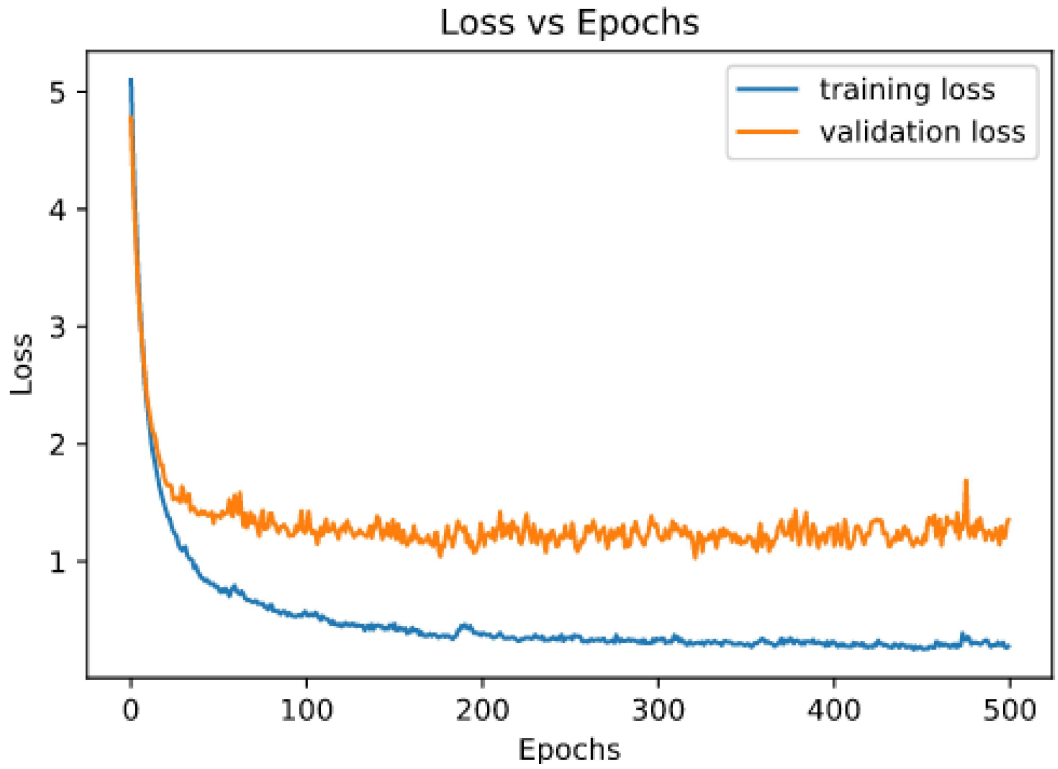
Figure 6: Confusion Matrix Result

7.4 TEAM - 4 Model Results

7.4.1 Emotion Recognition

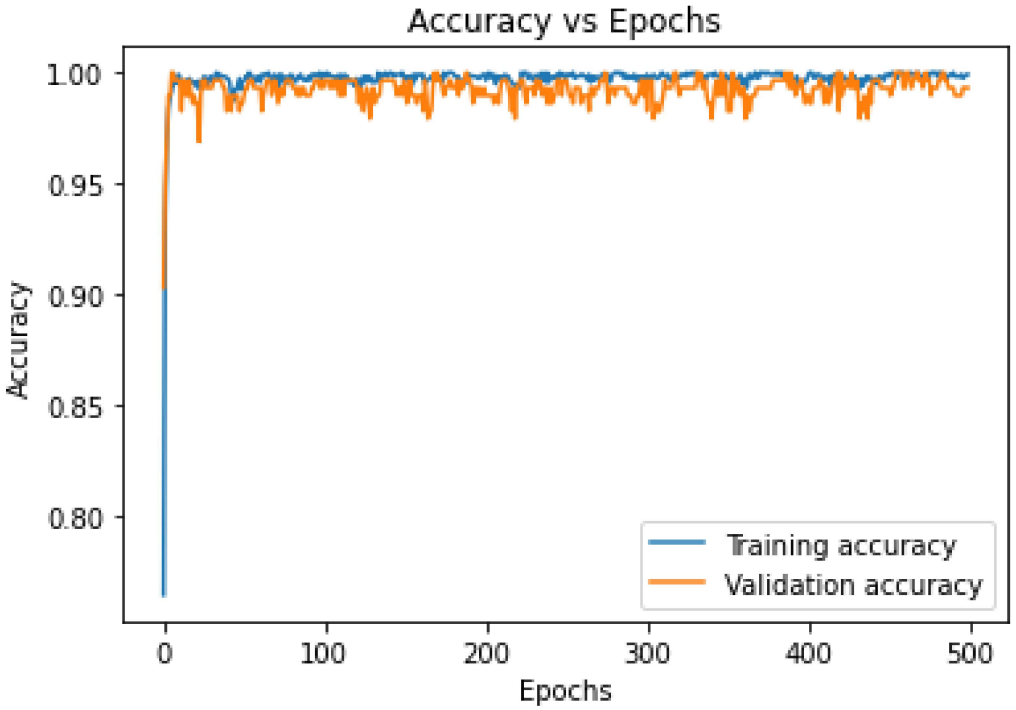
Accuracy obtained on training data is 96.96 %
Accuracy obtained on test data is 71.18 %

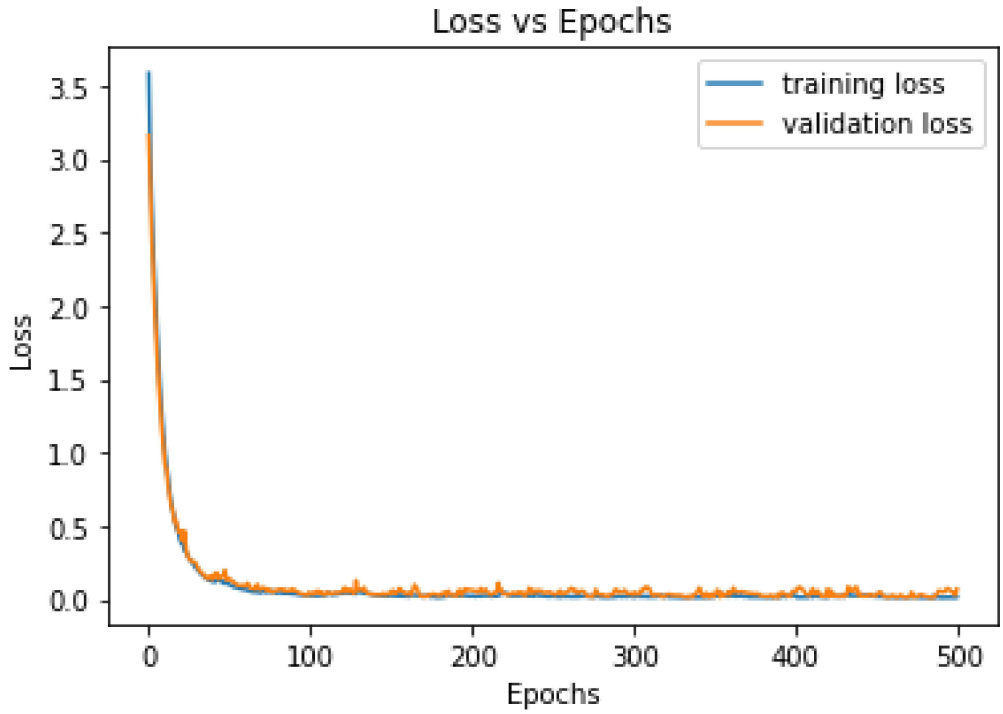




7.4.2 Gender Recognition

Accuracy obtained on training data is 99.91 %
Accuracy obtained on test data is 99.30 %

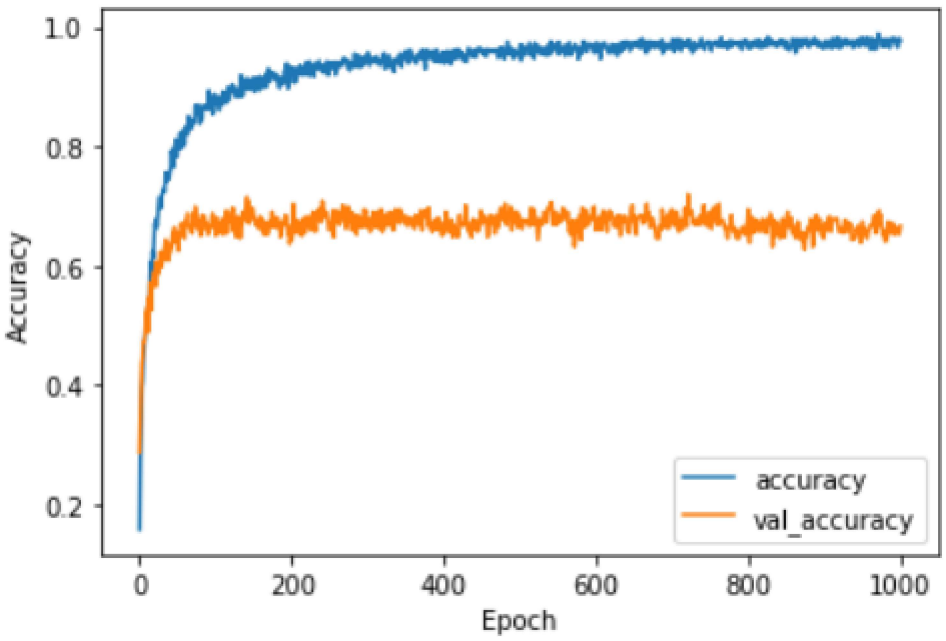




7.5 TEAM - 5 Model Results

7.5.1 Emotion Recognition

Accuracy obtained on training data is 97.72 %
Accuracy obtained on test data is 66.677 %



7.5.2 Gender Recognition

Accuracy obtained on training data is 99.6 %
Accuracy obtained on test data is 99.07 %