Real-Time Speech Emotion Recognition

Group Name: **Valar Morghulis**
Group Members:

First name	Last Name	Student number
Harshkumar	Patel	C0848343
Kanan	Trivedi	C0851492
Kishan	Lakhatariya	C0849341
Mohammaddin	Masbi	C0850500

Submission date: ***18-08-2022***
Professor: **Dr. Parisa Naraei**

Table of Contents

1. Abstract

- We have trained the LSTM model with two audio datasets with different sample rates and TESS and RAVDESS, which contain a total of 4240 audio samples with appropriate labels with extracting three unique features which are RMSE, MFCCs and Mel Spectrogram that can recognize seven different human emotions which a categorical accuracy of 96% with gender classification.
- To check the model on real-time audio, we have programmed an application that continuously takes the audio as input and every 2.6 seconds, it performs emotion and gender detection on audio and shows the output.

2. Introduction

- Emotion plays a significant role in daily interpersonal human interactions. Speech Emotion Recognition is the act of attempting to recognize human emotion and affective states from speech. This capitalizes on the fact that voice often reflects underlying feelings through tone and pitch. The idea behind creating this project is to build a machine learning model that could detect emotions from the human voice. It will analyze different emotions like Calm, Happy, Fearful, Disgust, Neutral, Sad, and Angry.
- Classifying the emotion based on the voice and speech is much crucial task to complete. Because the tone, energy and frequency of the voice determine the feelings to extract these features from audio files is not an easy task.
- The purpose of building this project is to make the online health care sector more advance by interpreting the voice of the patient in voice-over meetings. The second purpose of the model is used in chatbots and personal AI assistance to recognize emotion and react or reply based on the user's feelings. The third use case is in the call center or marketing sector to determine the clients feeling on the call so the call center representative can understand the need of the client, and also the call center organization can able to check which representative makes the customer happy most over the period of time.

3. Methods

i. Libraries

- We have used the Pandas library for handling the dataset. For audio handling, pre-processing and feature extraction, we have used Librosa, Pydub, and Pyaudio libraries. And for feature scaling and label encoding, we have used the Sklearn library. To visualize the data, we have used python libraries called matplotlib and seaborn. And for model training and evaluation, we have used TensorFlow and Keras.

ii. Dataset

- In the model training, we have used two different datasets, which are,
  a. The RAVDESS (Reyorson Audio-Visual Database of Emotional Speech and Songs) dataset. It consists voices of 24 professional actors. 12 are male voices, and 12 are female voices vocalizing two lexically matched statements in a neutral North American accent. The total size of this dataset is 24.8 GB. It contains in total 1440 audio samples with 8 different emotions.
  b. And the TESS (Toronto emotional speech set) dataset, A set of 200 target words, was spoken by two actors and actresses aged 26 and 64 years with seven emotions (anger, disgust, fear, happiness, pleasant surprise, sadness, and neutral). There are 2800 stimuli in total.

iii. Dataset Pre-Processing

- The first set of pre-processing is creating the Pandas data frame with all paths of audio files with its emotion and gender from two different datasets.
- For extracting appropriate emotion and gender from the audio file, the name of the file contains everything, so we separated the elements of the audio name and mapped appropriate labels.
- The RAVDESS filename consists of a 7-part numerical identifier (e.g., 03-01-06-01-02-01-12.wav).

- 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).

○ For the TESS filename OAF/YAF_actorename_emotion.wav, it contains direct emotions rather than numbers.
- OAF = Male
- YAF = Female

○ But there is a difference in the count of the emotions as RAVDESS has 8 emotions, and TESS has 7 emotions. The one different emotion is Neutral, so we combine the neutral emotion with calm emotion because both are kindly similar.

○ After processing all the datasets, we stored all required information into a data frame which is Path, Emotion, and Gender, to further processing.

```
dataset = pd.DataFrame({
    'path': file_paths,
    'gender': file_gender,
    'emotion': file_emotions
})
# dataset['emotion'] = dataset['emotion'].map(EMOTIONS)
print('Shape=>', dataset.shape)
dataset["emotion"] = dataset["emotion"]
dataset.head(50)
```

```
Shape=> (4240, 3)
```

	path	gender	emotion
0	Data/RAVDESS/audio_speech_actors_01-24/Actor_0...	male	surprise
1	Data/RAVDESS/audio_speech_actors_01-24/Actor_0...	male	surprise
2	Data/RAVDESS/audio_speech_actors_01-24/Actor_0...	male	angry
3	Data/RAVDESS/audio_speech_actors_01-24/Actor_0...	male	fear
4	Data/RAVDESS/audio_speech_actors_01-24/Actor_0...	male	fear
5	Data/RAVDESS/audio_speech_actors_01-24/Actor_0...	male	angry

○

iv. Exploratory Data Analysis (EDA)

○ In the dataset, the frequency of male and female samples is identical by the count of 2120.

```
dataset['gender'].value_counts()
```

```
male      2120
female    2120
Name: gender, dtype: int64
```

○

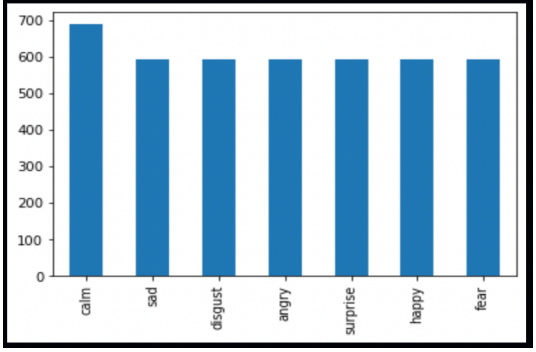
○ We discovered that the frequency of the 6 emotions is identical except for the emotion calm as we combine the neutral and calm from both datasets.

```
dataset['emotion'].value_counts()
```

```
calm        688
sad         592
disgust     592
angry       592
surprise    592
happy       592
fear        592
Name: emotion, dtype: int64
```

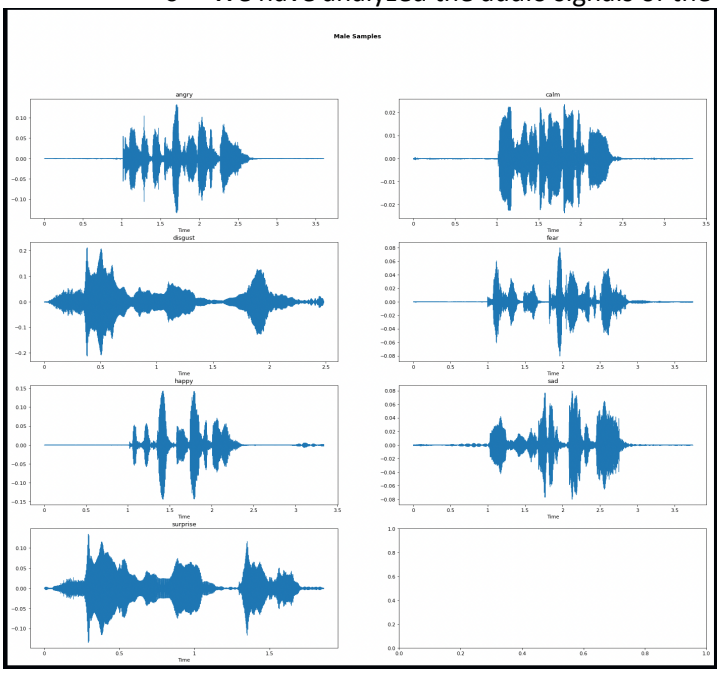
```
dataset.emotion.value_counts().plot.bar()
```

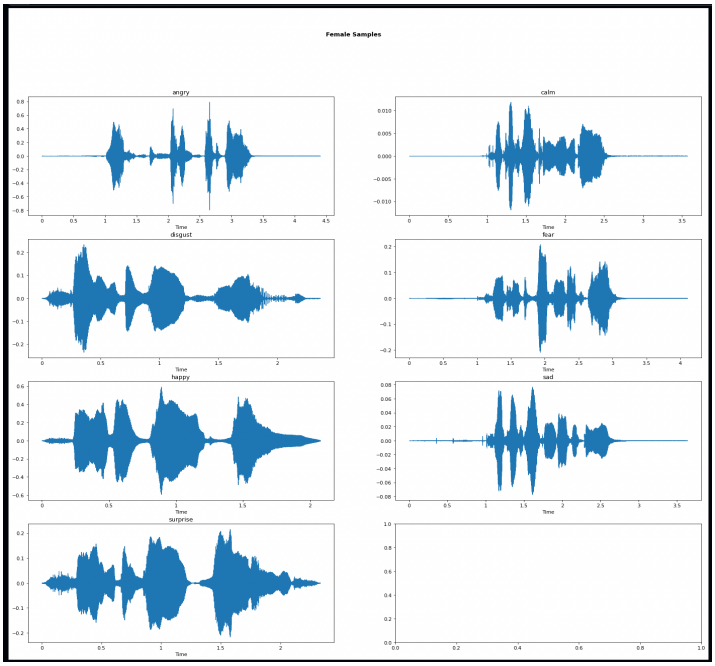
```
<AxesSubplot:>
```



○

- We have analyzed the audio signals of the different emotions and gender.





- Here, we can see that all emotion has its own shape in the form of energy and frequency with gender as female has a high pitch.
- We need to convert these audio signals into numeric vectors to make the computer understand the audio for model training.

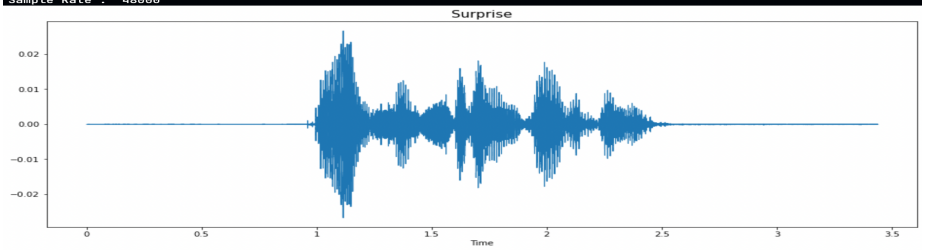
v. Pre-processing

- We have processed a single file for the initial pre-processing steps.
- We have taken a random file from the dataset and displayed the initial audio signal using Librosa and Matplotlib library.

```
#Taking an audio file form male_sample to perform preprocessing to know the audio data more
path = male_sample[-1]

audio, sr = librosa.load(path, sr=None)
print("Sample Rate : ", sr)
create_waveplot(audio, sr, "Surprise")
ipd.display(ipd.Audio(audio, rate=sr))
```

```
Sample Rate :  48000
```

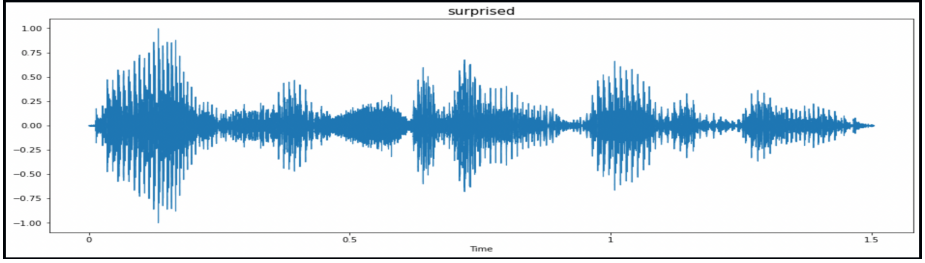


- Then we trim the audio file with the silent part lower than 30 DB

```
# Trimming the audio to remover silence before and after speech
trimmed_audio, index = librosa.effects.trim(normalizedsound, top_db=30)

create_waveplot(trimmed_audio, sr, "surprised")

ipd.display(ipd.Audio(data=trimmed_audio, rate=sr))
```



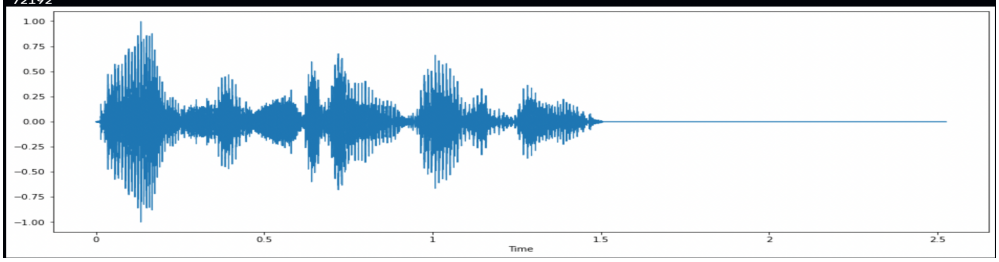
- We have discovered the uneven length of the audio file; to overcome this problem, we fixed the length of the audio file to 2.6 seconds. To make all audio lengths equal, we padded the zeros to the file; if the audio is smaller than 2.6 sec and if the length is longer than 2.6 sec, we cut the audio file from the end.

```
# Padding some values to right side to make all data length equal
# so we have maximum lenght audio file with 173056
print(len(trimmed_audio))
final_audio = np.pad(trimmed_audio, (0, 121212 - len(trimmed_audio)),
                     'constant')

create_waveplot(final_audio, sr, "")

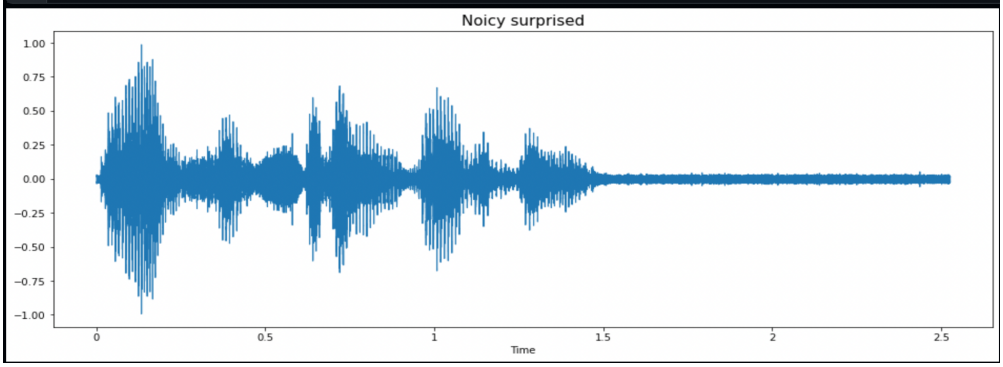
ipd.display(ipd.Audio(data=final_audio, rate=sr))
```

```
72192
```



- We observed that the dataset is too low to fit the model. Also, to make emotion recognition possible in noisy audio, we doubled up the dataset by adding the noise to every file during preprocessing.
- So, before adding noise, we have 4240 audio samples, and after adding noise, we have a total sample of audio is 8480.

```
noisy_audio = noise(final_audio)
create_waveplot(noisy_audio, sr, "Noicy surprised")
ipd.display(ipd.Audio(data=noisy_audio, rate=sr))
```

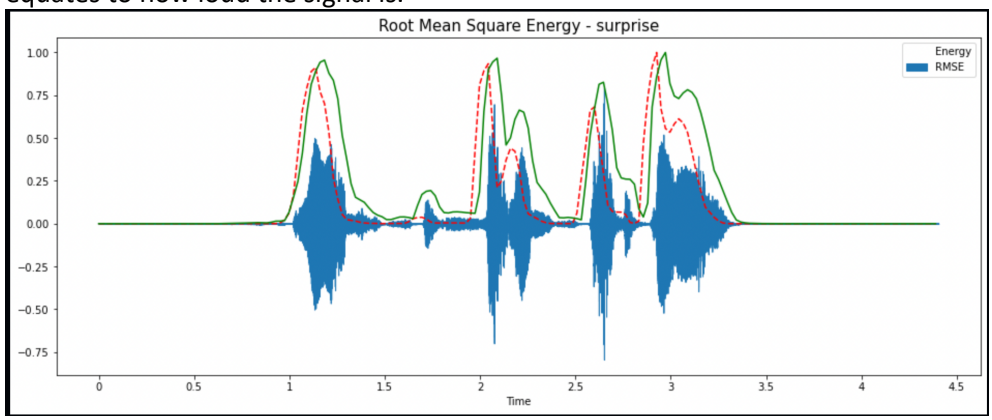


vi. Feature Extraction and Selection

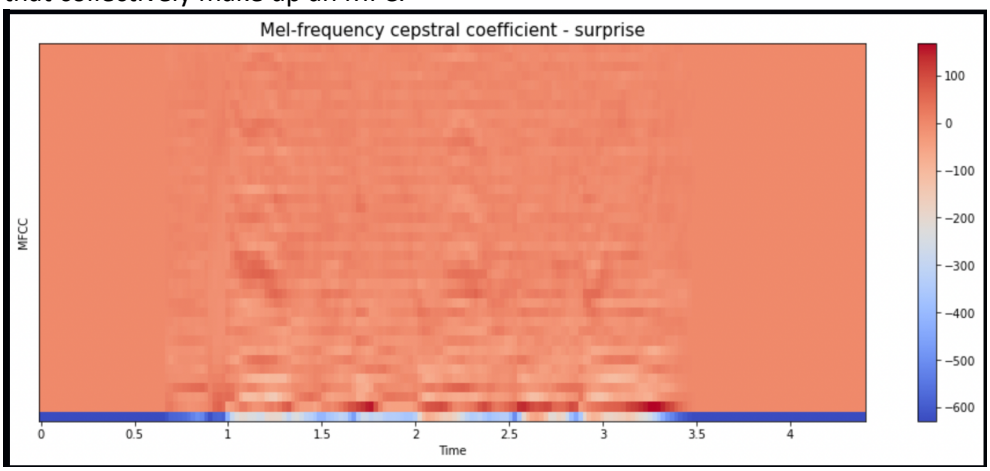
- We have extracted and selected these three features from the audio signal.

a. Root Mean Square Energy (RMSE)

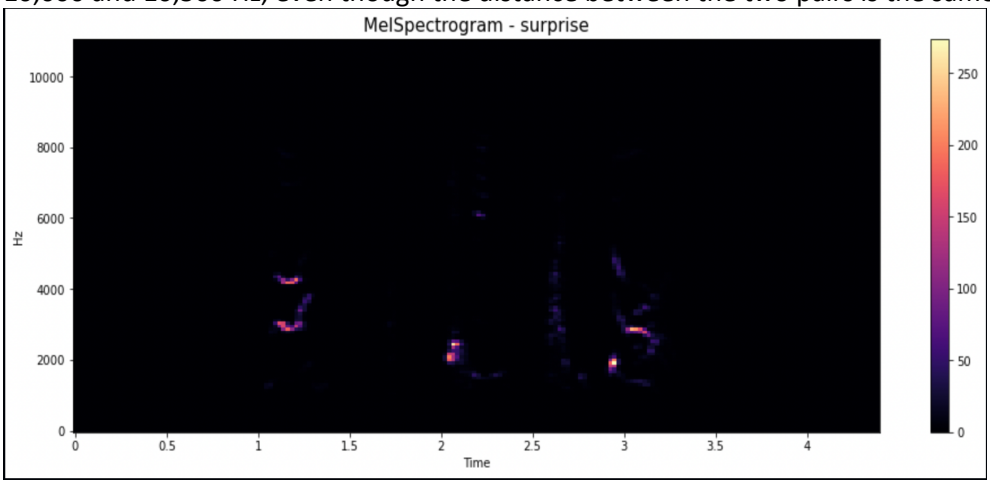
- The overall magnitude of a signal corresponds to its energy. For audio signals, this generally equates to how loud the signal is.



b. Mel-frequency cepstrum (MFC)

- In sound processing, the Mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound based on a linear cosine transform of a log power spectrum on a nonlinear Mel scale of frequency. Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC.
- 

c. Mel Spectrogram

- The Mel Spectrogram remaps the values in hertz to the Mel scale as we are better at detecting differences in lower frequencies than in higher frequencies. For example, we can easily tell the difference between 500 and 1000 Hz, but we will hardly be able to tell the difference between 10,000 and 10,500 Hz, even though the distance between the two pairs is the same.
- 

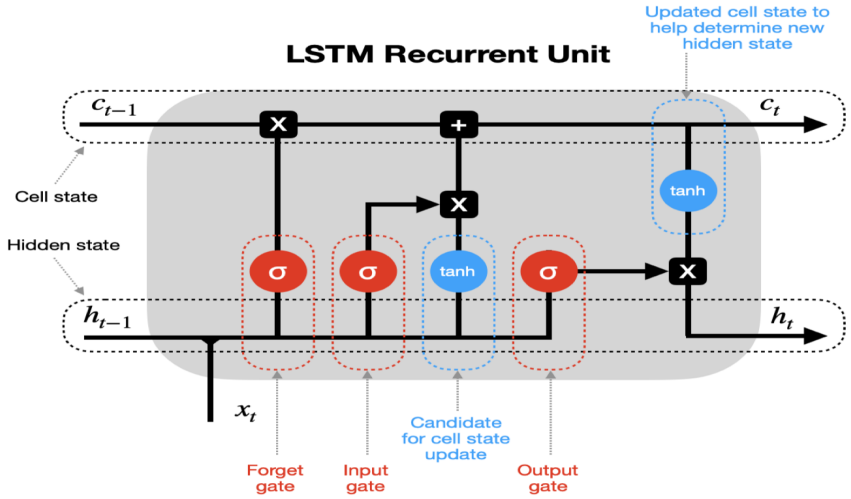
- From this, all pre-processing and feature extraction was performed on one file; we created two functions for pre-processing and feature extraction to perform this step on all files.
- After extracting all features, we scaled the feature with the Standard Scaler of the Sklearn library and converted it into a NumPy array.
- We encoded all 7 labels with One Hot Encoding.
- Now, we have everything to train the LSTM model.

vii. Model Training and Evaluating

- We have the following data as features:
- ```
  print('RMS shape:', f_rms.shape)
  print('MFCCs shape:', f_mfccs.shape)
  print('mel shape:', f_mel.shape)

  print('X shape:', X.shape)
  print('Y shape:', Y.shape)
  ```
  ```
  RMS shape: (8480, 237, 1)
  MFCCs shape: (8480, 237, 40)
  mel shape: (8480, 237, 128)
  X shape: (8480, 237, 169)
  Y shape: (8480, 7)
  ```
  ```

- We split the dataset into train and validation sets with the ratio of 80/20 using the train_test_split function of the Sklearn library.
- For this problem, we have selected the Long Short-Term Memory (LSTM) algorithm to build the neural network.
- LSTM is a sub-category of RNN (Recurrent Neural Network), which can be able to decide which information is kept or which information should be deleted.



- We build a neural network with the following configuration.

```
Metal device set to: Apple M1 Pro

systemMemory: 16.00 GB
maxCacheSize: 5.33 GB

2022-07-31 07:55:47.822666: I tensorflow/core/common_runtime/pluggat
ting to 0. Your kernel may not have been built with NUMA support.
2022-07-31 07:55:47.823177: I tensorflow/core/common_runtime/pluggat
0/device:GPU:0 with 0 MB memory) -> physical PluggableDevice (device
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 237, 520)          1435200

 lstm_1 (LSTM)               (None, 237, 347)          1204784

 lstm_2 (LSTM)               (None, 237, 213)          477972

 lstm_3 (LSTM)               (None, 192)               311808

 dense (Dense)               (None, 64)                12352

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 7)                 231

=================================================================
Total params: 3,444,427
Trainable params: 3,444,427
Non-trainable params: 0
_________________________________________________________________
None
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_12 (LSTM)              (None, 237, 128)          152576

 lstm_13 (LSTM)              (None, 93)                82584

 dense_15 (Dense)            (None, 32)                3008

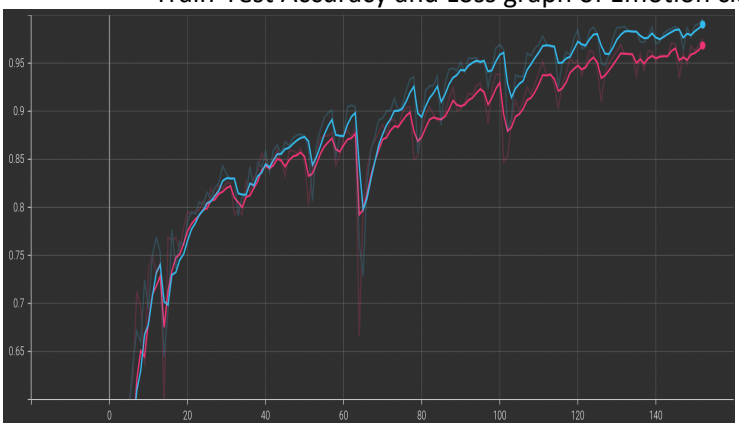
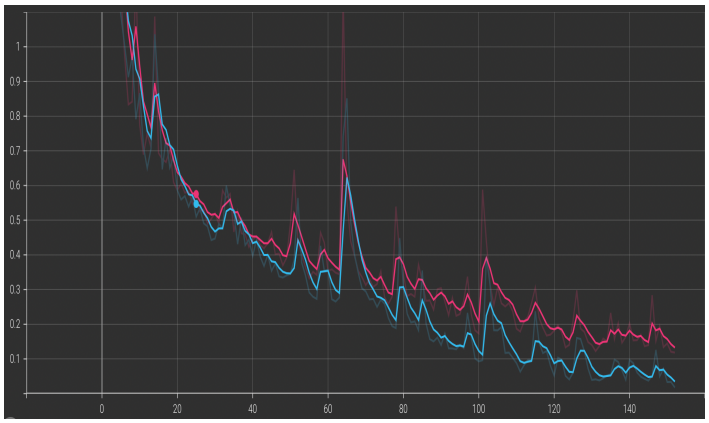
 dense_16 (Dense)            (None, 1)                 33

=================================================================
Total params: 238,201
Trainable params: 238,201
Non-trainable params: 0
_________________________________________________________________
None
```

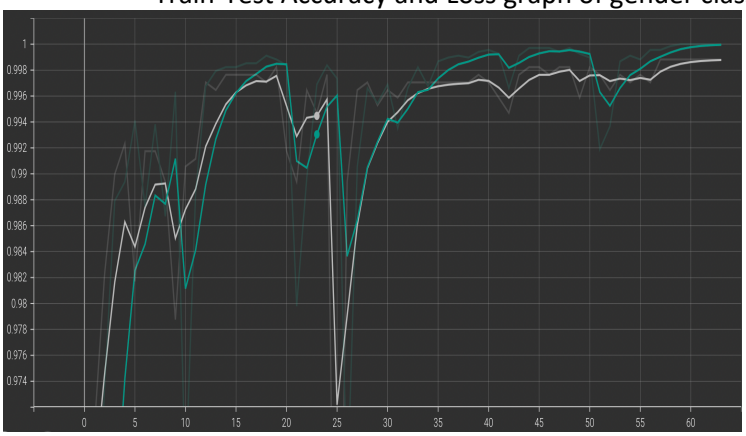
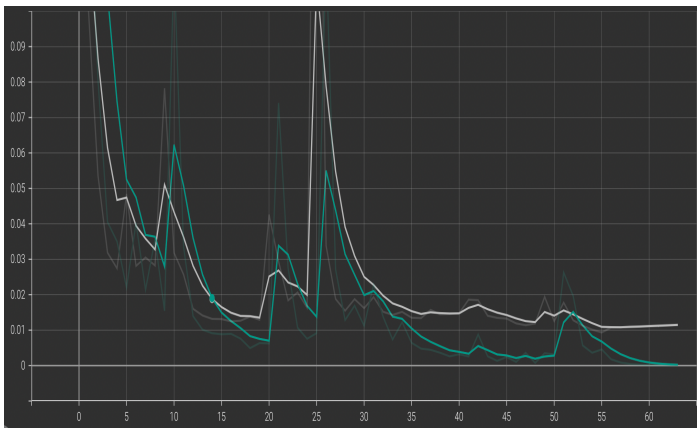
- We have used the Softmax activation function for the last layer of the neural network for Emotion Classifier as multiple labels need to predict, so this is the problem of multiclass classification.
- We compile the model with categorical_crossentropy as a Loss with Adam optimizer. And perform 160 epochs with batch size 256.
- To classify the gender, we used all the same features with the male-female label and performed the binary classification using the neural network shown in the second image above.
- For gender classification, we have used the sigmoid activation function for the last layer of the neural network, as this problem is a binary classification problem.

4. Results

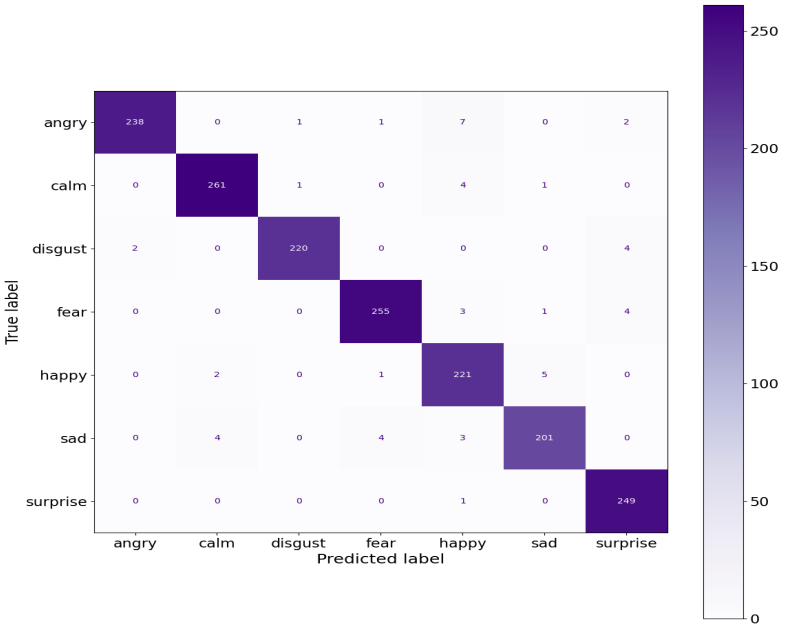
- We monitored all training performance and processes and visualized all graphs using the TensorBoard.
- Train-Test Accuracy and Loss graph of Emotion classifier:

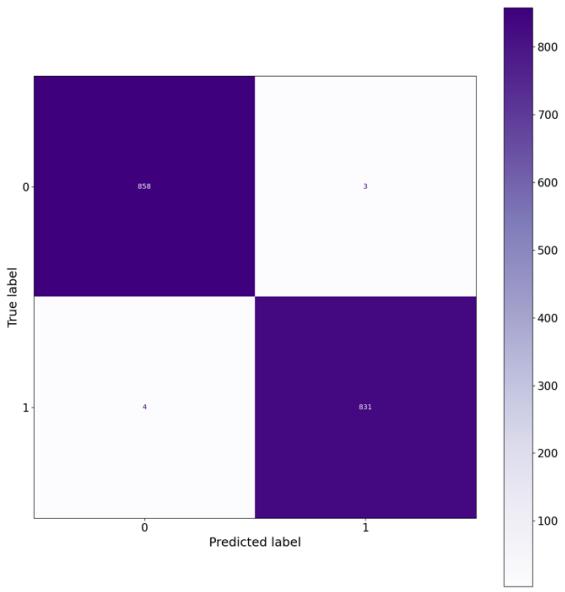



- We have achieved the categorical training accuracy of 0.9906 with a loss of 0.035 and the validation accuracy of 0.9684 with a loss of 0.1184.
- Train-Test Accuracy and Loss graph of gender classifier:

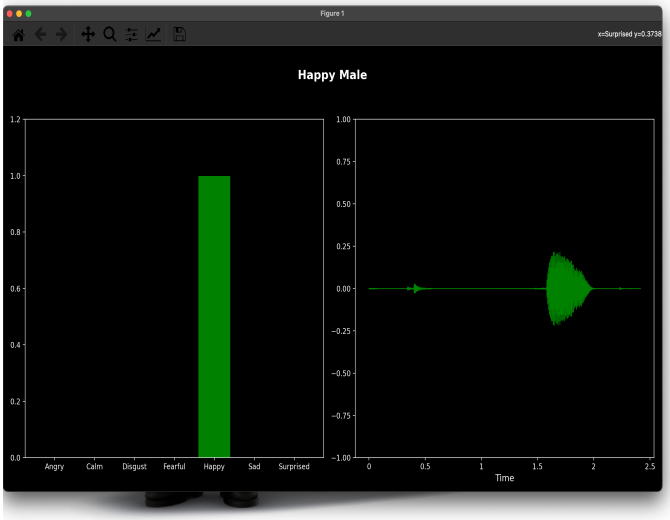



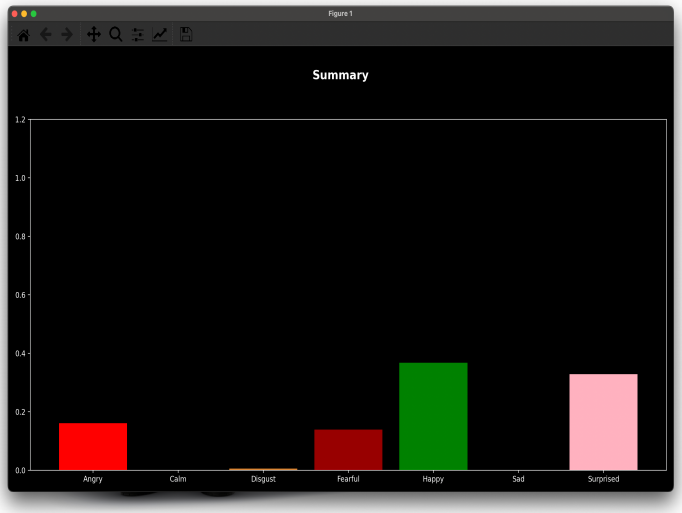
- The gender classifier is 99% accurate on the validation data, with a loss of 0.011.
- Confusion Matrix for Emotion and Gender Classifiers.





- We plot the confusion of the model performance on the validation dataset for the emotion and gender classifier.
- We have also built an application that can perform detection in real-time and display the detected emotion with gender, and at the end of the stream, it shows the overall summary of the session.





5. Conclusion

- To conclude, we have built an application that streams real-time audio using a computer's microphone and shows the emotion and gender every 2.6 Seconds of the audio using the model we have built on the LSTM neural network by feeding the extracted features RMSE, MFCCs, and Mel Spectrogram from 8480 audio samples with preprocessing steps and achieved the accuracy of 96% on the validation dataset.

6. Reference:

- TSpace. (n.d.). Retrieved August 18, 2022, from https://tspace.library.utoronto.ca/handle/1807/24487
- Livingstone, S. R., & Russo, F. A. (2018, April 5). The Ryerson Audio-Visual Database of emotional speech and Song (RAVDESS). Zenodo. Retrieved August 18, 2022, from https://zenodo.org/record/1188976#.Yv5fKy8r35g
- MeidanGR. (n.d.). MeidanGR/speechemotionrecognition_realtime: Speech emotion recognition (SER) in real-time, using Deep Neural Networks (DNN) of Long short memory term (LSTM). GitHub. Retrieved August 18, 2022, from https://github.com/MeidanGR/SpeechEmotionRecognition_Realtime
- Chauhan, Nagesh Singh. "Audio Data Analysis Using Deep Learning with Python (Part 1) - KDnuggets." KDnuggets, www.kdnuggets.com, https://www.kdnuggets.com/2020/02/audio-data-analysis-deep-learning-python-part-1.html. Accessed 18 Aug. 2022.