INDIVIDUAL REPORT

Akshat Saini

EMOTION ANALYSIS USING SPEECH

Introduction and Project Goal:

The objective of the final project is to classify emotions after listening to a speech or a short voice note and do a comparative analysis of the different dynamic models like RNN or LSTM and a convolutional model like CNN.

Individual contribution and Code Implementation:

The final project involved usage of the RAVDESS speech audio dataset. The dataset included 1440 .wav audio clips, equally recorded by men and female actors. The dataset was perfectly balanced with men actors recording the same number of clips as the female actors. The clips consisted of 2 statements ‘Kids are talking by the door’ and ‘Dogs are sitting by the door’, with each actor saying the two statements in different emotions or tones.

The audio files existed in the format of ‘03-01-01-01-01-01-01.wav’ and required fetching feature information from the name itself.

Here 03 stands for the modality of the note, suggesting where the clips are in speech, song or video format. All clips have ‘03’, suggesting speech. The third 01 suggests the emotion class, with 01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised. The last 01 suggests whether the actor is a male or a female, with even numbers being female actors and odd numbers being male actors.

I began collecting feature information by extracting the target emotion values and the gender information from the audio file names using the python package pandas. Once I had collected all the information in a list, I made a dataframe called the df_main containing all the information along with the file path of the audio clips.
Since the data was originally in audio format, I had to convert it into something that could be understood and used by the model for analysis and future prediction. For this task, I used the python library Librosa which is commonly used for audio analysis and processing. Audio data, especially with varying pitch or frequency is usually converted to their mel-spectrogram values.

The mel-spectrogram values when plotted against the time on X-axis can show the distribution of frequency or pitch in the audio file. This was used for my analysis of each of the audio clips. The Librosa library allowed me to convert each audio file into their respective mel-spectrogram values using the below functions:

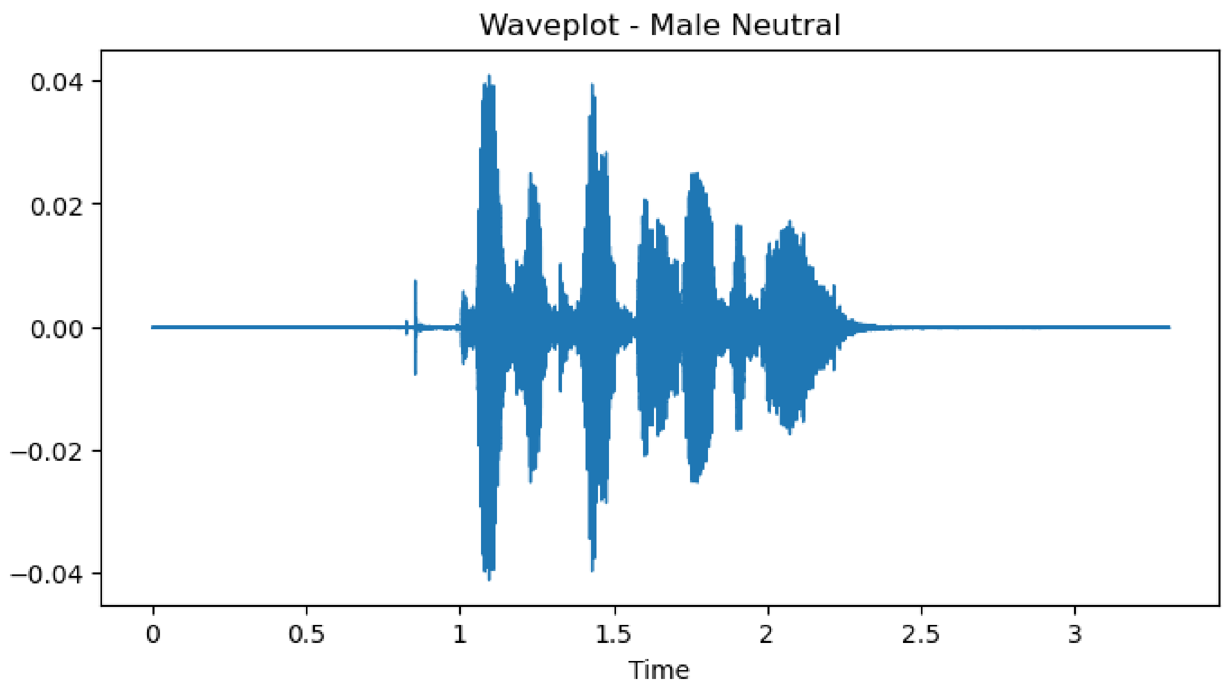
```
x, sr = librosa.load('03-01-01-01-01-01-01.wav')
spectrogram = librosa.feature.melspectrogram(y=x, sr=sr, n_mels=128,fmax=8000)
spectrogram = librosa.power_to_db(spectrogram)
```

Here the x value is the audio time series of the respective clip and sr is the sampling rate of the clip. The feature.melspectrogram(x, sr) function allowed me to extract the mel-spectrogram values of the audio file. The sampling rate of the audio clip was the same across all audio files with 44100 or 44k hz.

A waveplot of speech using the display.waveshow(x, sr) function:

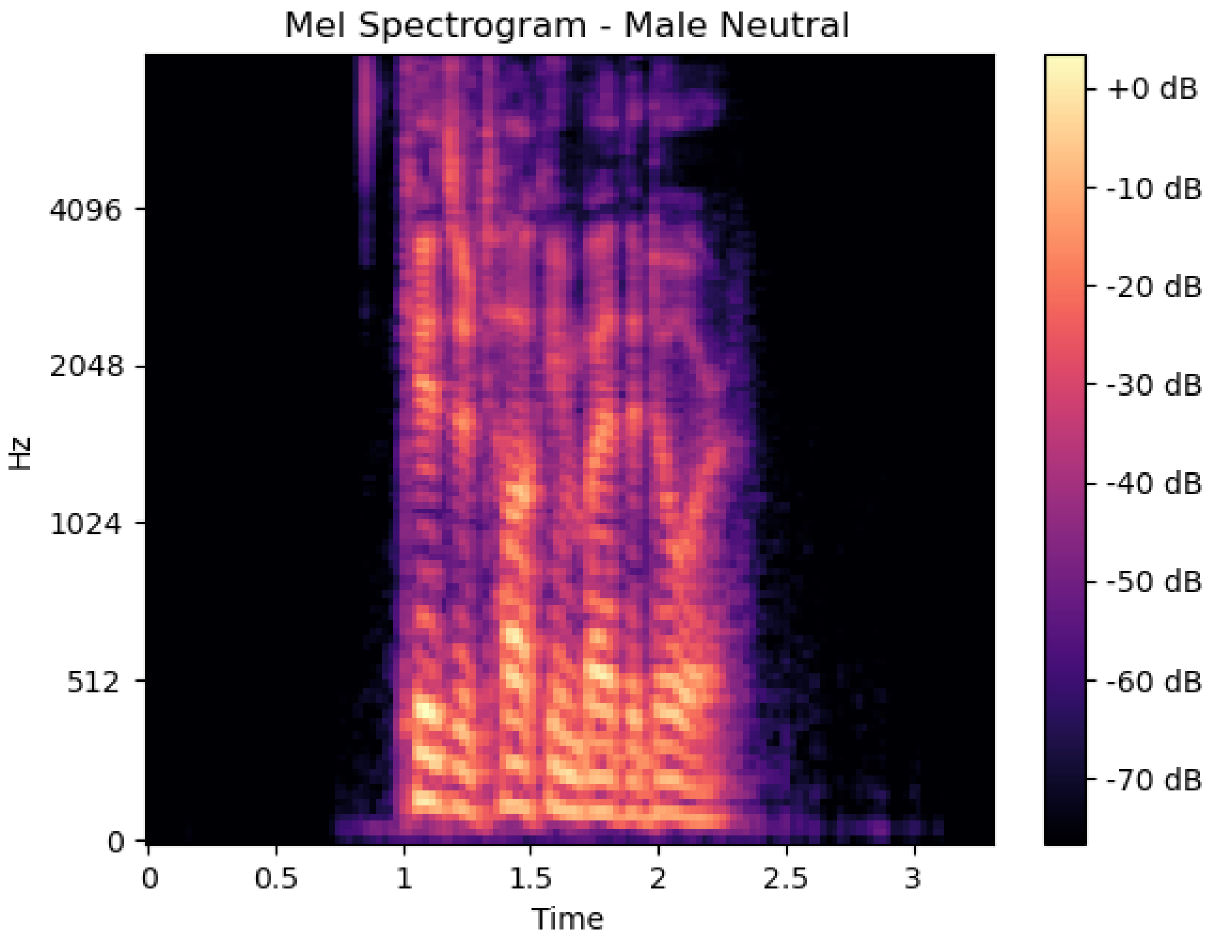
```
plt.figure(figsize=(8, 4))
librosa.display.waveshow(x, sr = sr)
plt.title('Waveplot - Male Neutral')
plt.show()
```

Here the emotion of the male actor is ‘Neutral’



When the ‘spectrogram’ value extracted above from the feature.melspectrogram(x, sr) is plotted against time using the function display.specshow(), the below plot is generated:

```
librosa.display.specshow(spectrogram, y_axis='mel', fmax=8000, x_axis='time');
plt.title('Mel Spectrogram - Male Neutral')
plt.colorbar(format='%+2.0f dB');
```



Once I had collected the mel-spectrogram values for all the audio clips I created a df_main pandas dataframe which included the mel values, the target emotion class and the actor gender.

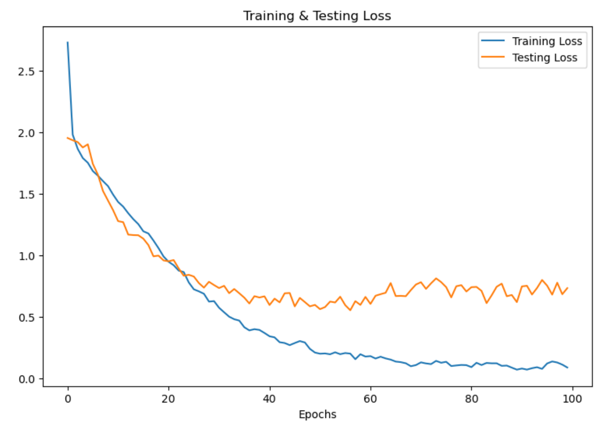
Model:

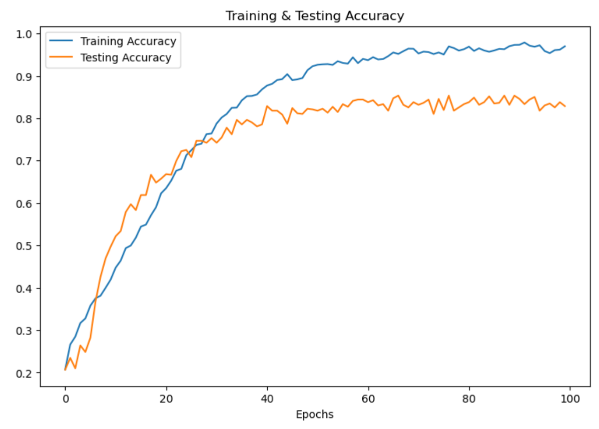
Once I had all the data in the format that could be used for model creation, I began by first creating a baseline convolution model or a CNN model. CNN models have historically performed best on audio datasets due to their ability to detect patterns. The baseline CNN model was run on the original data which included the 1440 audio files. The model architecture used was a base CNN model with 2 hidden layers and an input followed by the softmax function in the output layer. The softmax function results in 8 different probabilities for the target classes.

The baseline CNN model finished with an accuracy score of 45% on the test data set.

Following this, I did audio data augmentation such as adding noise to the data, changing pitch of the file and adding time shift among many. The data augmentation was applied randomly to the original audio clips and the target emotion and the gender was still noted to be the same as the original clip after augmentation.
Once the data was augmented, which added both variability and increased size of the dataset, I ran the CNN model again. Executing the CNN model again, I received an accuracy of 85% on the test set.

The accuracy and loss plots were generated for the final CNN model which used audio augmentation. These can be seen below:





Final Notes:

CNN models have usually performed better than other dynamic models like LSTM or GRU for audio clips. This seems to be the case here as well. I also realized that increasing the size of the data set using audio augmentation increased the test set accuracy by almost 40%.
Another reason the LSTM model did not perform the best could be because the model did not require the usage or the benefit of a feedback loop, which the LSTM model provides over CNN. Predicting patterns seems to be more important in this case as compared to having the full temporal data.
Finally, I would like to add that I could have tried creating a third dataset just for conducting a comparative analysis between the different models. This approach could have helped me to avoid cherry picking the best scores even within the CNN model, as there can be cases where models that have a high test accuracy can perform low on unseen data compared to models that had a low accuracy on test set, which can end up performing better on unseen data points.

Percentage of Code Borrowed:

Total lines written in ‘main_cnn.py’: 262
Lines borrowed: 21
Lines edited in borrowed lines: 4

Calculation: [(21-4)/262] x 100 = 6.48

References:

librosa/librosa: Python library for audio and music analysis
https://github.com/librosa/librosa

Dataset:
https://zenodo.org/record/1188976#.YFZuJ0j7SL8

https://paperswithcode.com/dataset/ravdess

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0196391