SPEECH EMOTION RECOGNITION

By

- Sai Mounica Pothuru
- Harshini Bonam
- Chaitanya Sharma

Contents

What Is Speech Emotion Recognition (SER) ?

SER is the act of attempting to recognize human and affective states from speech.

SER is **tough** because emotions are subjective, people would interpret it differently. This makes it **hard to define** the **notion of emotions**.

Objectives

- Develop various classification models.
- Evaluate the performance of the generated models.
- Analyze the approach and results.

Data Set

- **Source:**
  [The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS)]

- **Description:**
  - Speech files (215 MB)
  - Contains 1440 files: 60 trials per actor x 24 actors

- **File naming convention:**
  - Each of the RAVDESS files has a **unique** filename.
  - Consisting a 7-part numerical identifier as labels
  - E.g., 02-01-06-01-02-01-12.wav
    These identifiers define the stimulus characteristics.

Data Set

Source : The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS)

- **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
  - Statement
  - Repetition
  - Actor

Libraries

- For Data wrangling in Python:
  - Soundfile - Read and write audio files
  - Librosa - Audio and video analysis
  - Wavfilehelper - Read and write audio files
  - Numpy
  - Pandas
  - Scipy
- For Data Visualization in Python:
  - IPython - To play audio files
  - Plotly - Plots
  - Matplotlib
  - Seaborn
  - Tqdm

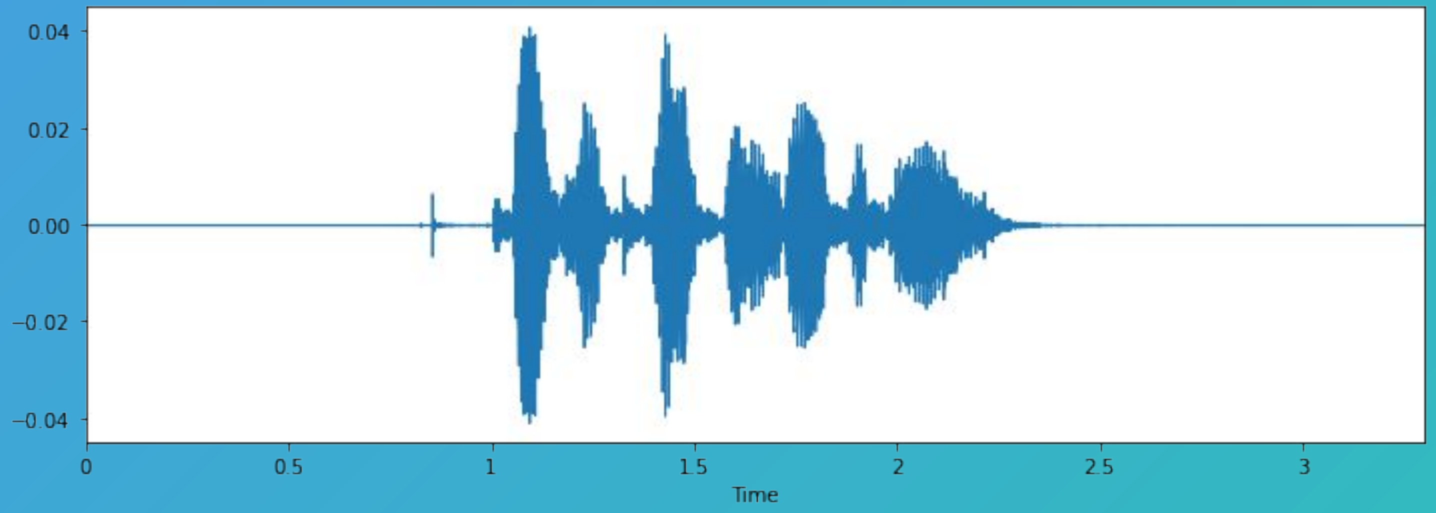
Libraries

- For machine learning
  - Keras - CNN and other metrics
  - Sklearn - MLP and other metrics

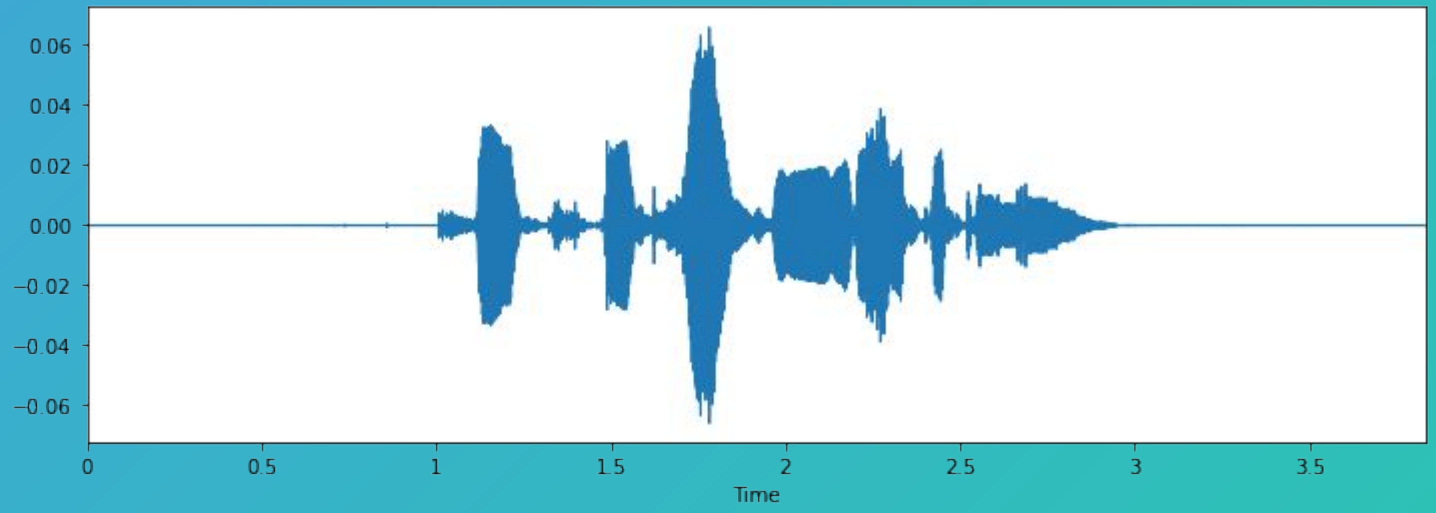
Exploration

To visualise the speech files, Waveforms of different emotions are plotted & are as follows :

1. Actor 1 - Emotion Neutral

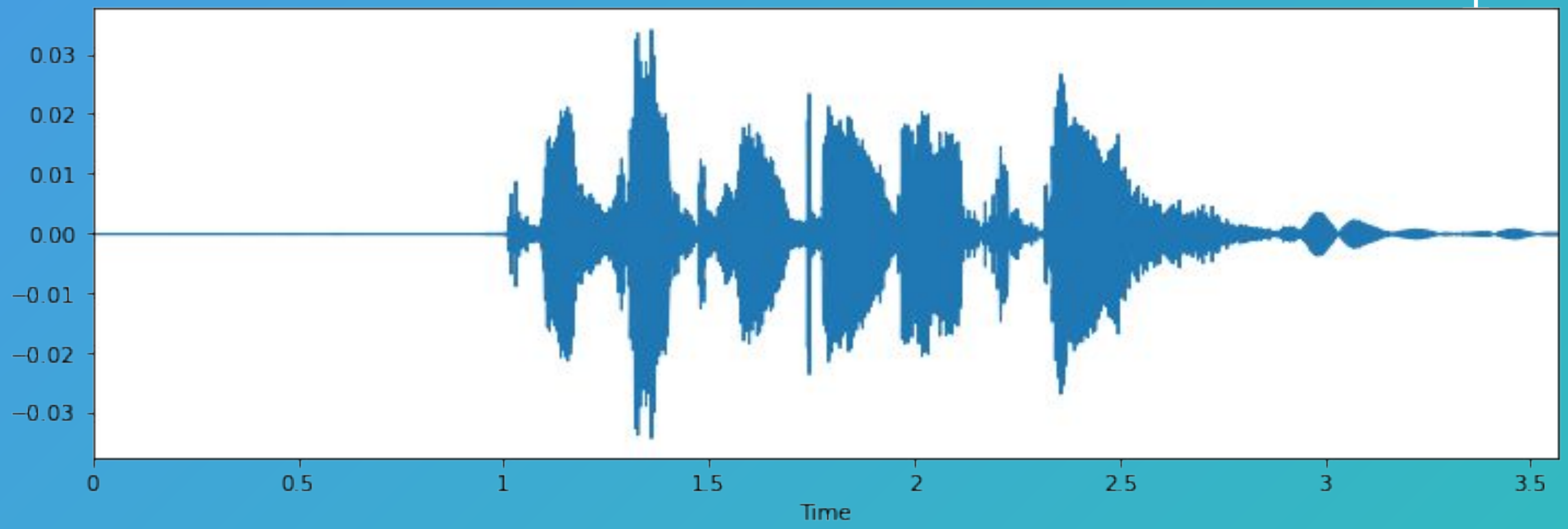


2. Actor 2 - Emotion Calm

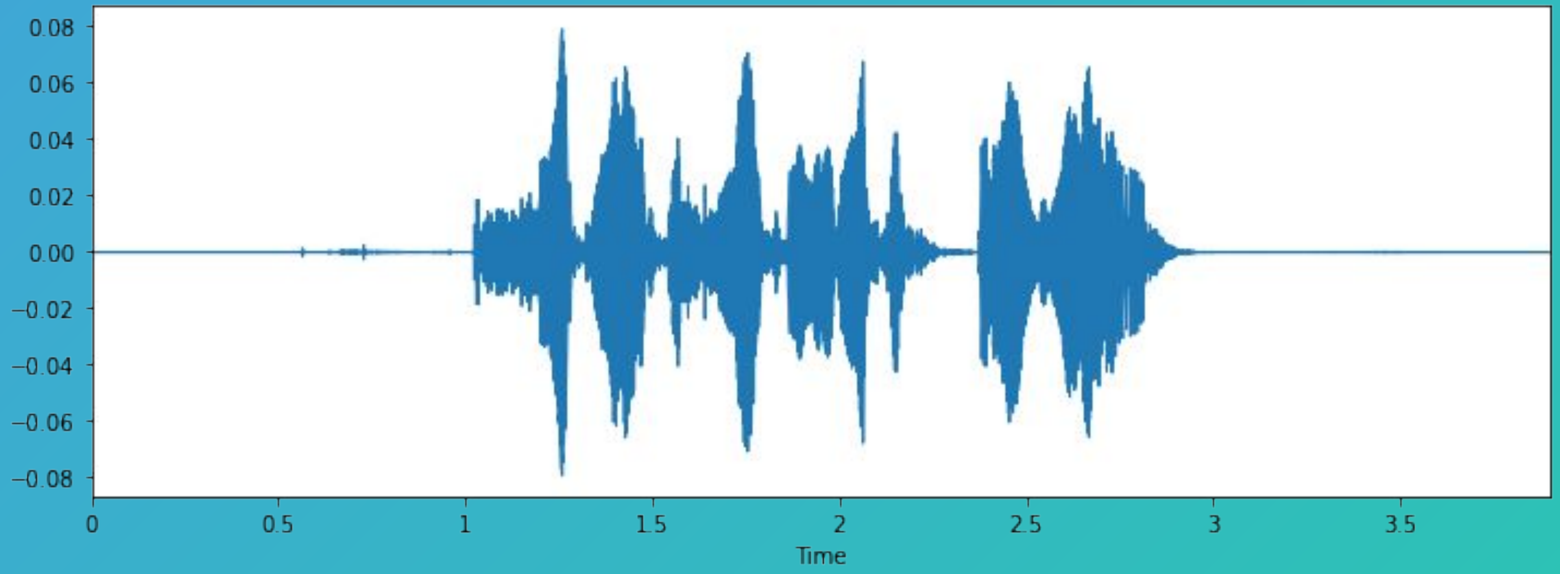


Exploration

3. **Actor 3 - Emotion Sad**



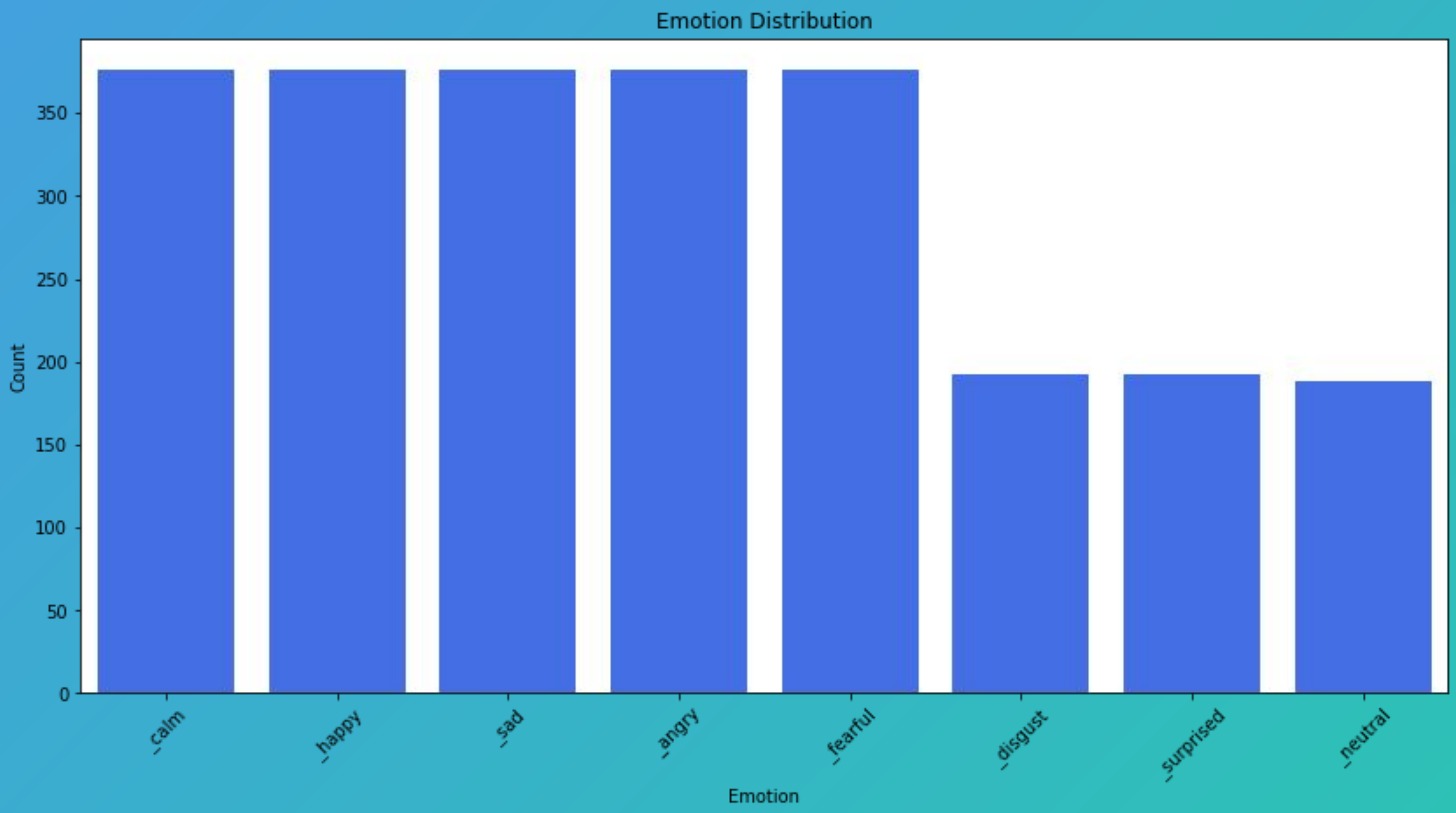
4. **Actor 4 - Emotion Disgust**



Exploration

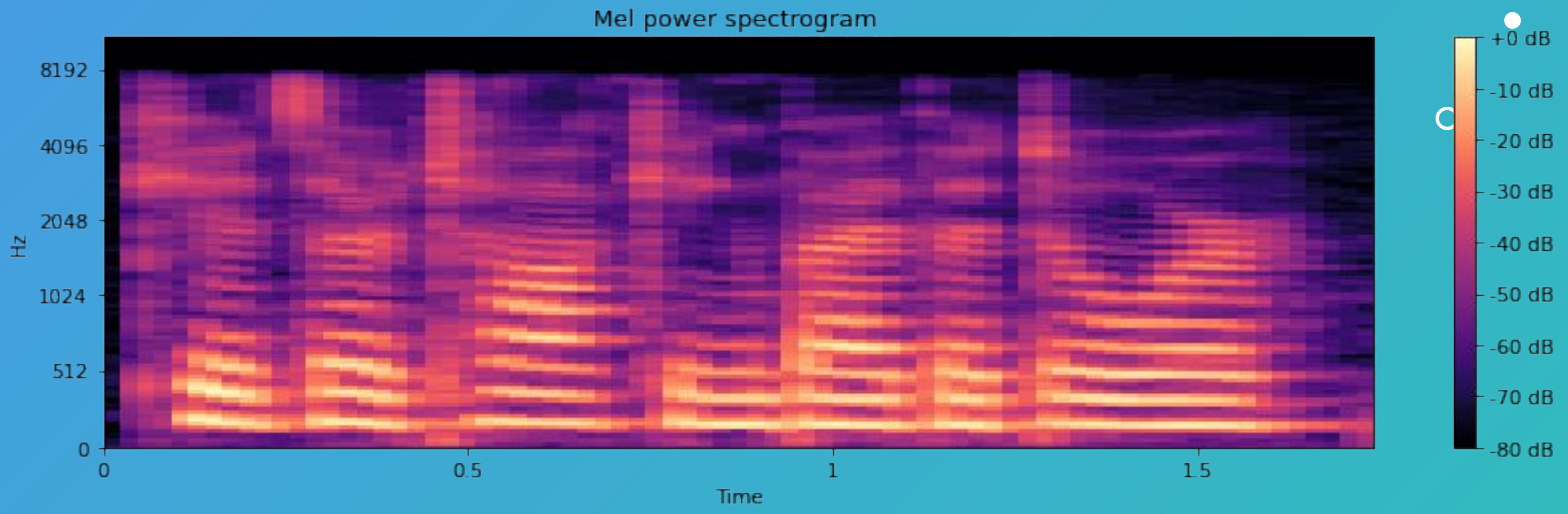
5. A Distribution Bar chart of Emotions Class:

The below graph shows the various emotions distribution in the dataset

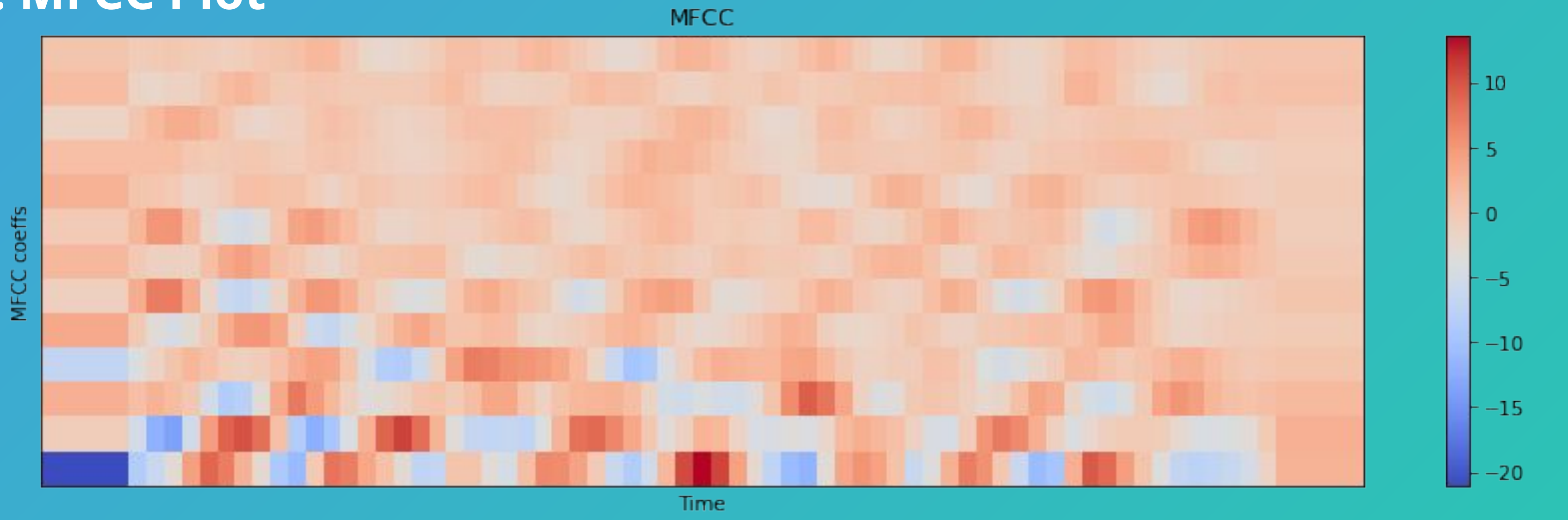


Exploration

6. MEL Spectrogram



7. MFCC Plot



Exploration Synopsis

Based on the Explorations, we concluded that out of all emotions, five of the below are are quite significant .

calm
happy
sad
angry
fearful

For all our Models, we mainly **focused** on these **five**.

Feature Extraction

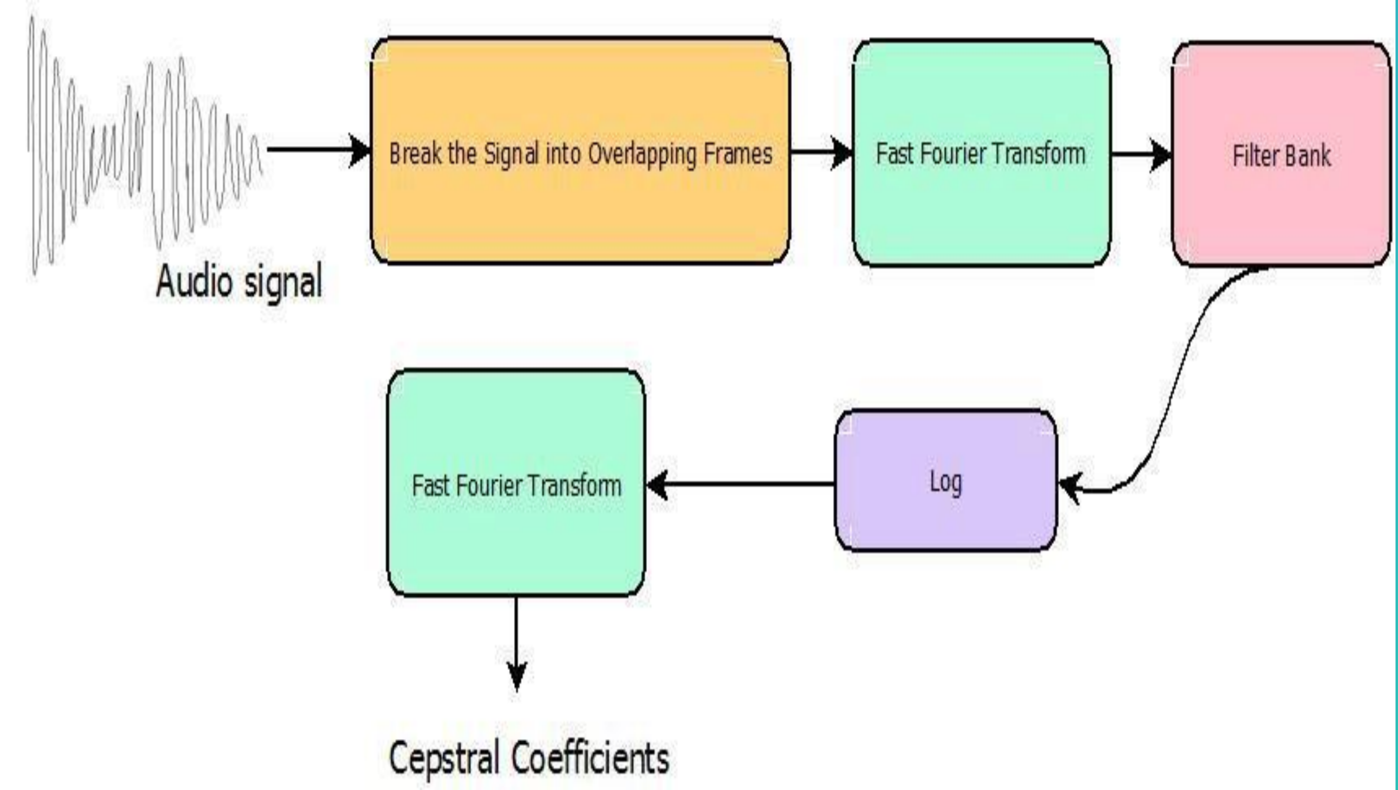
1. MFCC (Mel-Frequency Cepstral Coefficients)

- Summarises **frequency distribution with the time** characteristics of audio input.
- Also called **short term power spectrum of sound**
- State-of-the-art feature since it was invented in the 1980s.
- Speech is the sounds that are generated by a human filtered by the shape of vocal tract including tongue, teeth. This shape if can be determined correctly, one can accurately represent what sound comes out.
- The envelope of the time power spectrum of the speech signal is representative of the vocal tract and MFCC accurately represents this envelope.

Feature Extraction

1. MFCC (Mel-Frequency Cepstral Coefficients) (cont.)

- Our dataset has 40 MFCC features.



Feature Extraction

2. Chroma

- A descriptor which represents the tonal content of a musical audio signal in a condensed form.
- They capture harmonic and melodic characteristics of music.
- Our dataset has 12 Chroma features

Feature Extraction

3. Mel Spectrogram

- The Mel Scale is a logarithmic transformation of a signal's frequency.
- Core idea: Sounds of equal distance on the Mel Scale are perceived to be of equal distance to humans.
- Mel Spectrograms are spectrograms that visualize sounds on the Mel scale as opposed to the frequency domain.
- Our dataset has 128 Mel Spectogram frequencies

Model Architecture for MLP

1. Data is loaded and split into train, and test with 0.2 ratio split.

2. Fed the training data to Sklearn library's MLPClassifier.

3. This Multi-layer Perceptron Classifier, a feed-forward ANN optimizes the log-loss function using L-BFGS or SGD.

4. We tuned a better model with the following hyperparameters
    - alpha = 0.5, batch_size = 256
    - epsilon = 1 epsilon=5e-07,
    - hidden_layer = 300 max_iter=1000
    - learning_rate='adaptive'

```
[242] #DataFlair - Initialize the Multi Layer Perceptron Classifier
      model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-07, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=1000)
```

```
#DataFlair - Train the model
model.fit(x_train,y_train)
```

```
MLPClassifier(activation='relu', alpha=0.01, batch_size=256, beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-07,
              hidden_layer_sizes=(300,), learning_rate='adaptive',
              learning_rate_init=0.001, max_fun=15000, max_iter=1000,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

```
[244] #DataFlair - Predict for the test set
      y_pred=model.predict(x_test)
```

```
#DataFlair - Calculate the accuracy of our model
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)

#DataFlair - Print the accuracy
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
Accuracy: 64.24%
```

Model Architecture for CNN

1. CNN model with Keras.

2. Constructed with 4 Dense layers - 3 RELU activators and softmax activator, and Stochastic Gradient Descent optimizer, with a 0.1 dropout between each layer.

3. Compiled with 'categorical_crossentropy' loss method, accuracy as basis , and adam optimization algorithm for evaluation

4. Trained on a batch_size of 20 for 80 epochs, (increased to 500)

```python
num_epochs = 80
num_batch_size = 20

checkpointer = ModelCheckpoint(filepath='saved_models/weights.best.basic_mlp.hdf5',  verbose=1,   save_best_only=True, monitor='val_loss', mode='min')
start = datetime.now()

lr_reduce = ReduceLROnPlateau(monitor='val_loss', factor=0.9, patience=20, min_lr=0.0007)

model_cnn.fit(xc_train,  yc_train,
          batch_size=num_batch_size,
          epochs=num_epochs,
          validation_data=(xc_test, yc_test),
          callbacks=[checkpointer, lr_reduce],
           verbose=1)


duration = datetime.now() - start
print("Training completed in time: ", duration)
```

[182]
```python
# Evaluating the model on the training and testing set
score_cnn = model_cnn.evaluate(xc_train, yc_train, verbose=0)
print("Training Accuracy: ", score_cnn[1])

score_cnn = model_cnn.evaluate(xc_test, yc_test, verbose=0)
print("Testing Accuracy: ", score_cnn[1])
```

```
Training Accuracy:  0.6145493984222412
Testing Accuracy:  0.5411255359649658
```

Model Architecture for 2D-CNN with padding

1. CNN model with Keras.

2. Constructed with 5 layers - with 4 2D-ConvNNs each with RELU activators and softmax activator, with a 0.3 - 0.2 dropout between each layer.

3. Compiled with 'categorical_crossentropy' loss method, accuracy as basis , and adam optimization algorithm for evaluation

4. Trained on a batch_size of 15 for 60 epochs, (increased to 500)

```python
num_epochs = 60
num_batch_size = 15

checkpointer = ModelCheckpoint(filepath='saved_models/weights.best.basic_cnn.hdf5',
                               verbose=1, save_best_only=True)
start = datetime.now()

model_cp.fit(xcp_train,
             ycp_train,
             batch_size=num_batch_size,
             epochs=num_epochs,
             validation_data=(xcp_test, ycp_test),
             callbacks=[checkpointer], verbose=1)


duration = datetime.now() - start
print("Training completed in time: ", duration)
```

[212]

```python
# Evaluating the model on the training and testing set
score_cp = model_cp.evaluate(xcp_train, ycp_train, verbose=0)
print("Training Accuracy: ", score_cp[1])

score_cp = model_cp.evaluate(xcp_test, ycp_test, verbose=0)
print("Testing Accuracy: ", score_cp[1])
```

```
Training Accuracy:  0.6981541514396667
Testing Accuracy:  0.5844155550003052
```

Results

1. The CNN Classification model attained a training accuracy of 61.45% and validation accuracy of 54.11%

2. The CNN classification model with padding obtained a training accuracy of 69.81% and testing accuracy of 58.44%

3. The MLP Classifier achieved an accuracy of 64.24%

Conclusion

To conclude, out of the three classification models generated, the CNN Classifier attains the best results when the data is padded.

The implementation made sure that overfitting is avoided.

The accuracy can further be attained by increasing the size of the dataset and by tweaking the hyper-paramters.

References

- RezaChu - Speech emotion recognition with convolutional neural network, https://towardsdatascience.com/speech-emotion-recognition-with-convolution-neural-network-1e6bb7130ce3,2019.

- Shen L Pan Y, Shen P. Speech emotion recognition using support vector machine. int j smart home, https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.390.8138rep=rep1type=pdf, 2012.

References

- Neethu Sundar prasad. Speech emotion detection using machine learning techniques, https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1647context=etdprojects,2018.

- Dezhong PengZhang Yi Yuni Zeng, Hua Mao.Spectrogram based multi-task audio classification, https://link.springer.com/article/10.1007/s11042-017-5539-3, 2017

THANK YOU