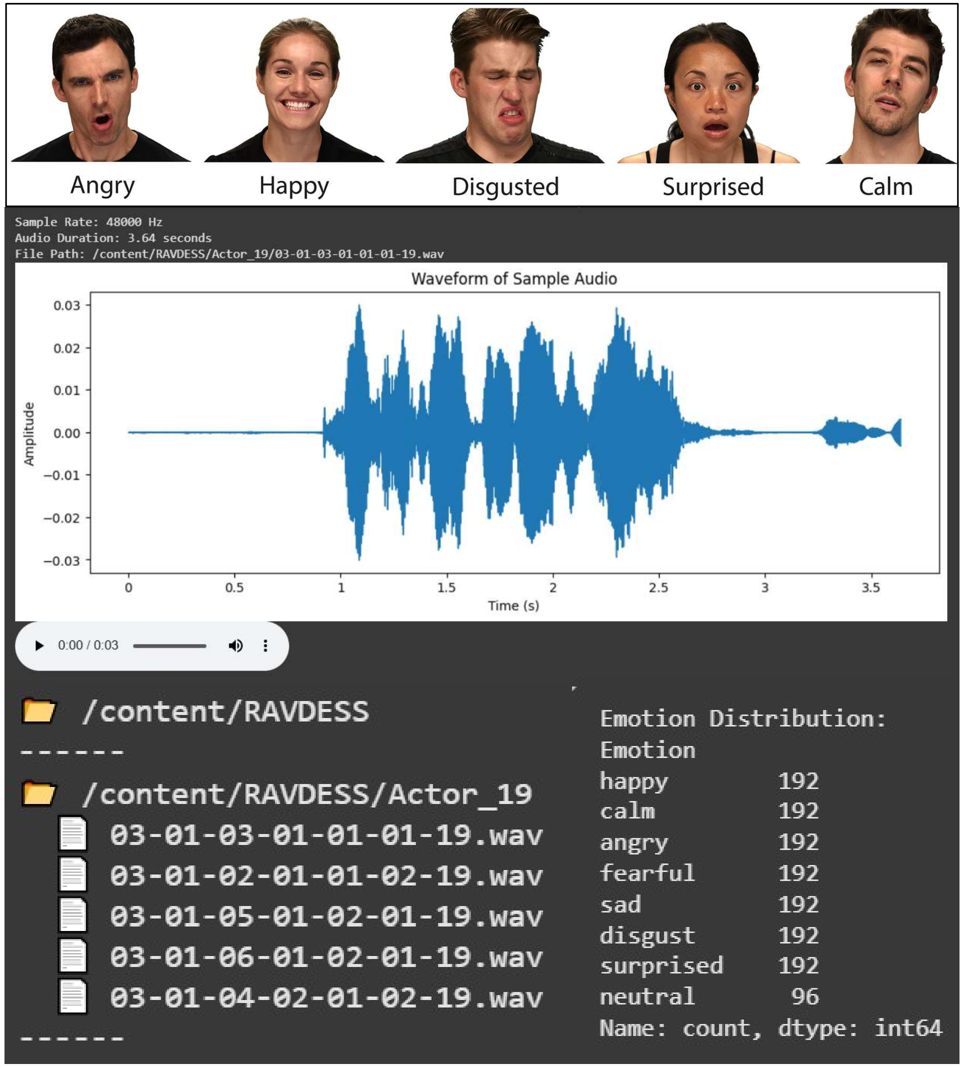
Speech Emotion Recognition Model

NURSHARINAH 6422706H

1. Dataset

- Dataset Used: "The Ryerson Audio-Visual Database of Emotional Speech and Song (**RAVDESS**)" by Livingstone & Russo is licensed under CC BY-NA-SC 4.0.
  - 1440 audio-only files (16-bit, 48 kHz .wav).
  - 8 emotions: calm, happy, sad, angry, fearful, surprise, disgust, neutral.
  - 24 professional actors (12 male, 12 female).

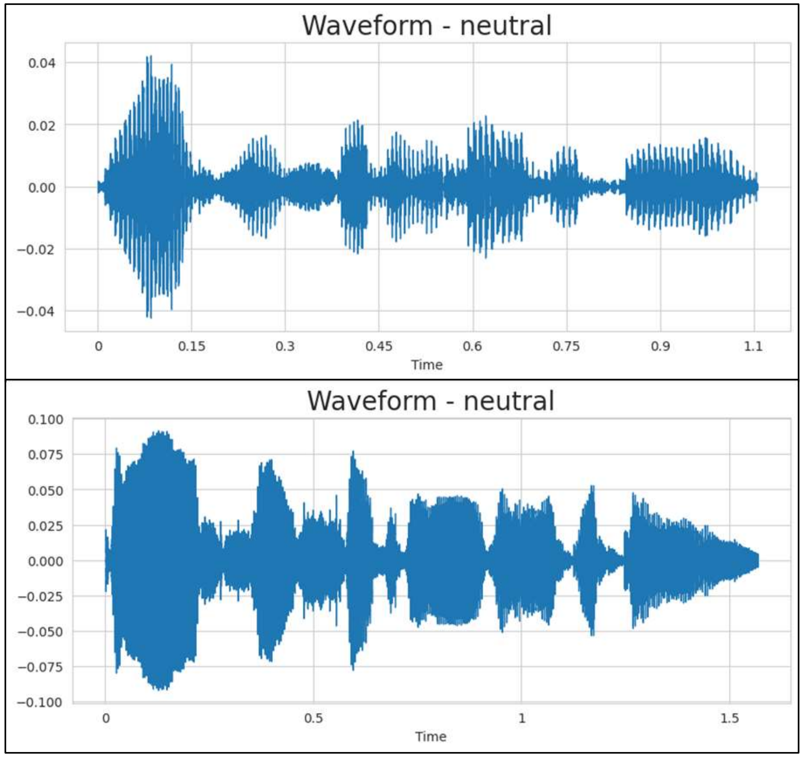


2. Data Preprocessing

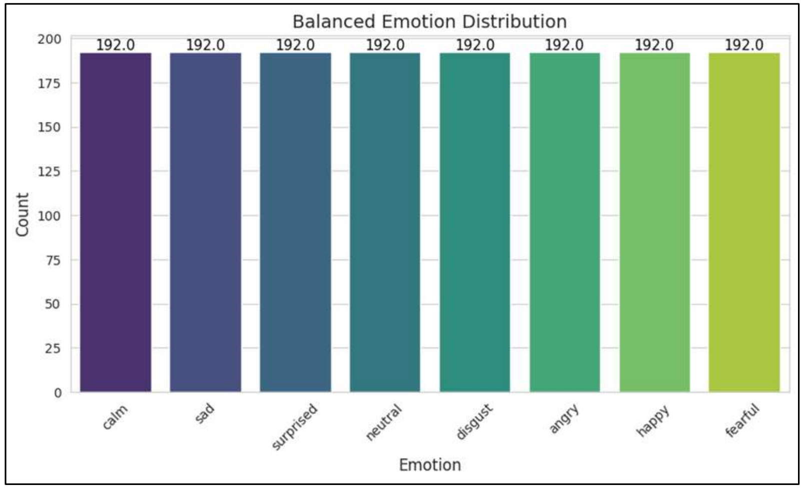
- Preprocessing Steps:
  1. **Trim Silence**: Remove leading and trailing silence from all audio files.
  2. **Handle Imbalance**:
     - Random Oversampling of "neutral" samples to balance with other emotions.
     - Stratified Under-Sampling to balance the total number of samples across neutral, positive, and negative emotion categories.
  3. **Emotion Mapping**: Map 8 emotions to 3 category labels (neutral, positive, negative).
  4. Feature Extraction – MFCCs & Spectrogram:
     - **Standardized Sample Rate**: Set all audio files to 16 kHz.
     - **Padding/Truncating**: Adjust all audio files to a consistent length based on dataset statistics.

3. Exploratory Data Analysis

1. Trim Silence



2. Handle Imbalance – Random Oversampling



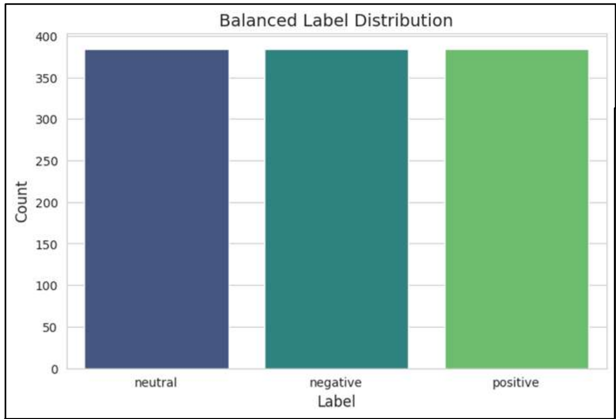
3. Exploratory Data Analysis

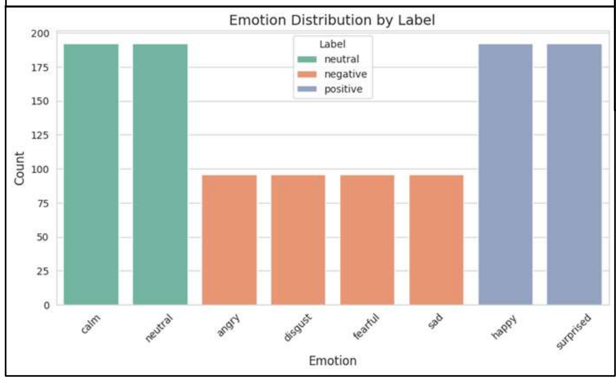
3. Emotion mapping to Labels

```
                  Filename  \
0  03-01-02-02-01-01-08.wav
1  03-01-04-02-02-02-24.wav
2  03-01-02-02-01-02-22.wav
3  03-01-08-01-02-01-13.wav
4  03-01-01-01-02-02-11.wav

                                         Filepath   Emotion  Gender  \
0  /content/RAVDESS_CLEANED/03-01-02-02-01-01-08.wav      calm  female
1  /content/RAVDESS_CLEANED/03-01-04-02-02-02-24.wav       sad  female
2  /content/RAVDESS_CLEANED/03-01-02-02-01-02-22.wav      calm  female
3  /content/RAVDESS_CLEANED/03-01-08-01-02-01-13.wav surprised    male
4  /content/RAVDESS_CLEANED/03-01-01-01-02-02-11.wav   neutral    male

      Label
0   neutral
1  negative
2   neutral
3  positive
4   neutral

Total Samples: 1536
```

```
Label Distribution:
Label
negative    768
neutral     384
positive    384
Name: count, dtype: int64
```

4. Handle Imbalance – Stratified Undersampling

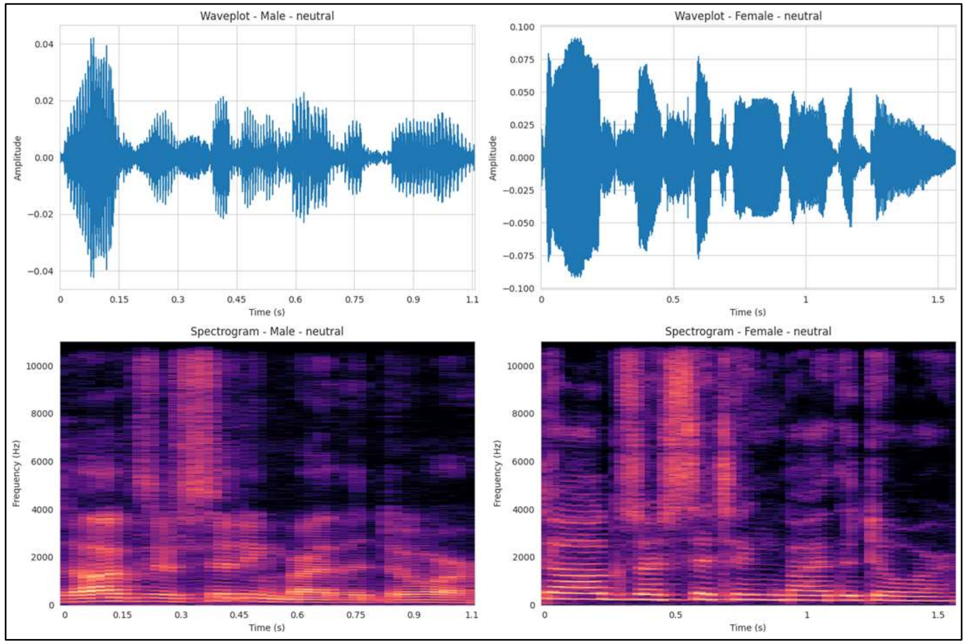




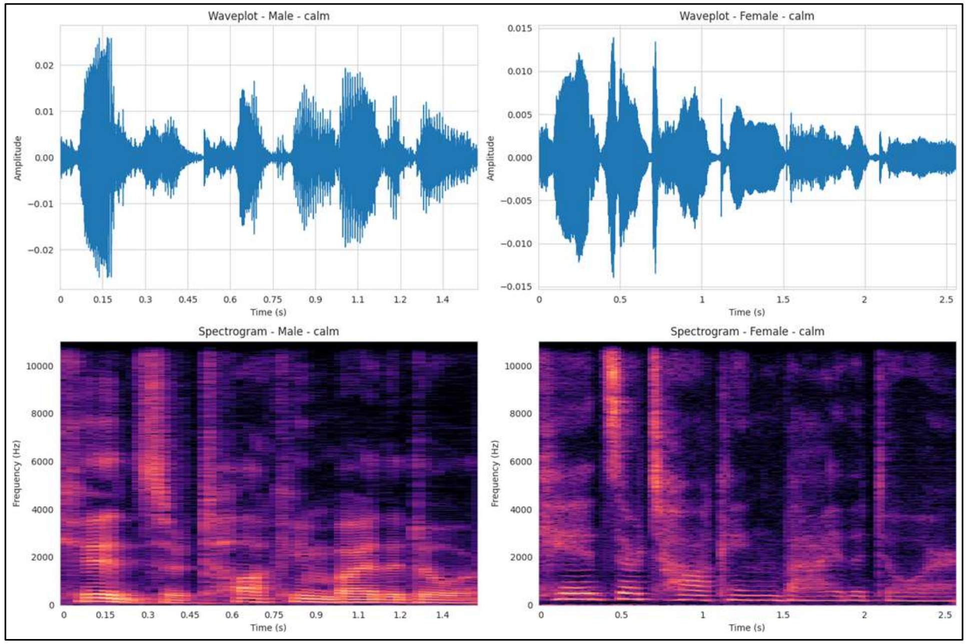

```
Emotion Count per Label:
Label     Emotion
negative  angry         96
          disgust       96
          fearful       96
          sad           96
neutral   calm         192
          neutral      192
positive  happy        192
          surprised    192
dtype: int64
```

3. Exploratory Data Analysis – Wave plot & Spectrogram

- Neutral – Male/Female

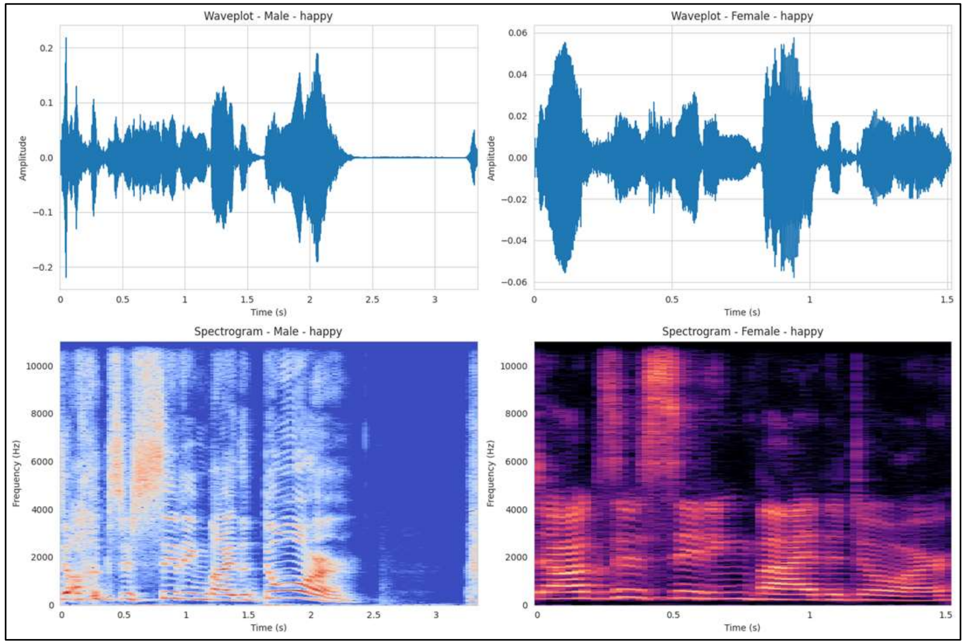


- Calm – Male/Female

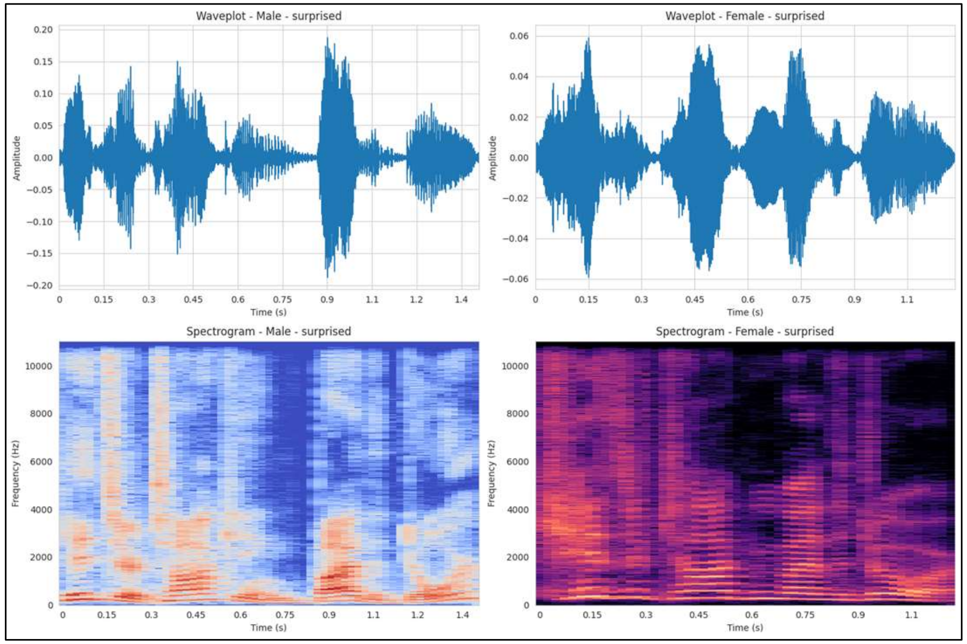


3. Exploratory Data Analysis – Wave plot & Spectrogram

- Happy – Male/Female

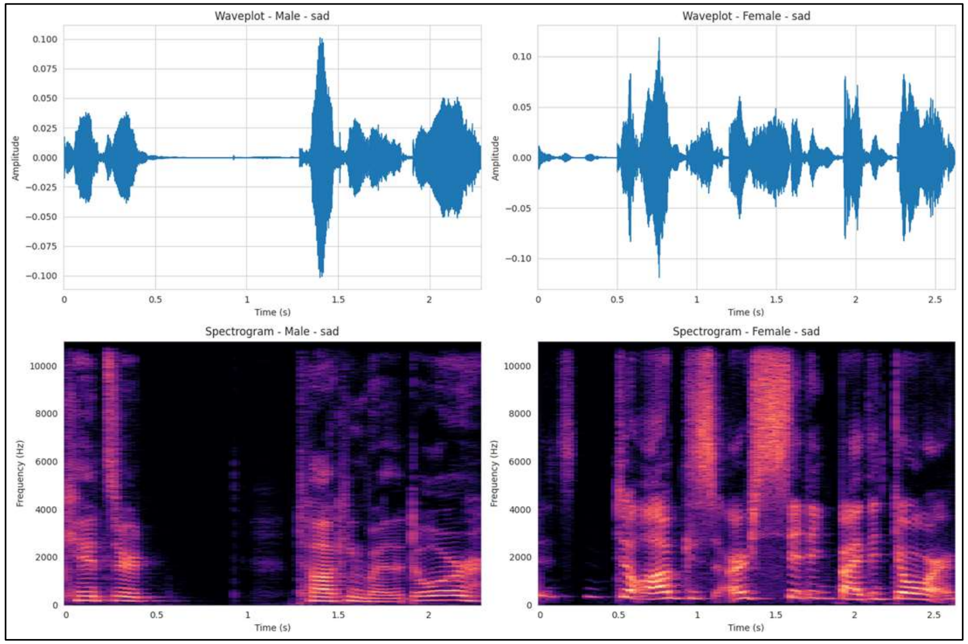


- Surprised – Male/Female

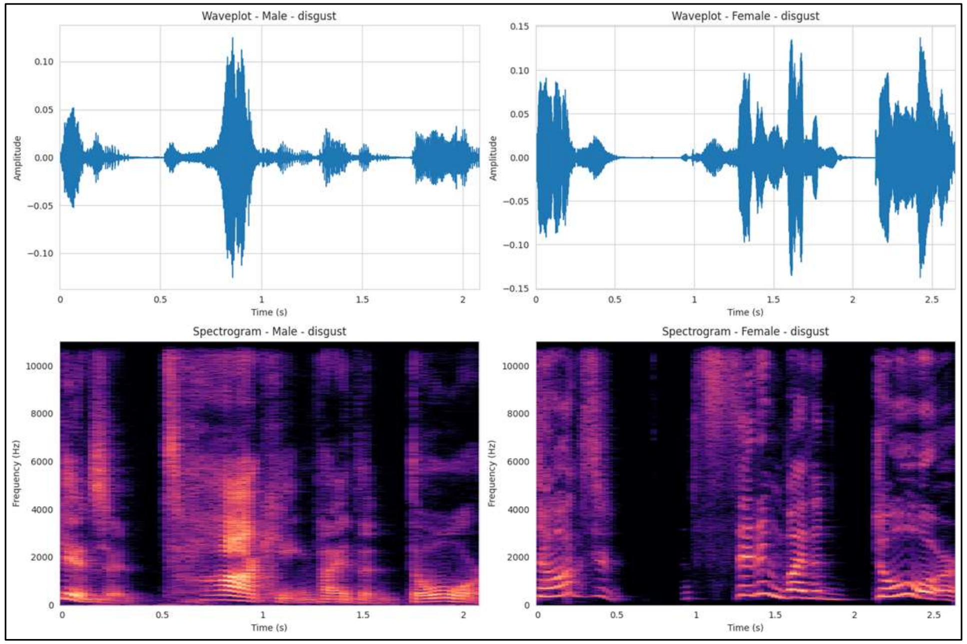


3. Exploratory Data Analysis – Wave plot & Spectrogram

- Sad – Male/Female

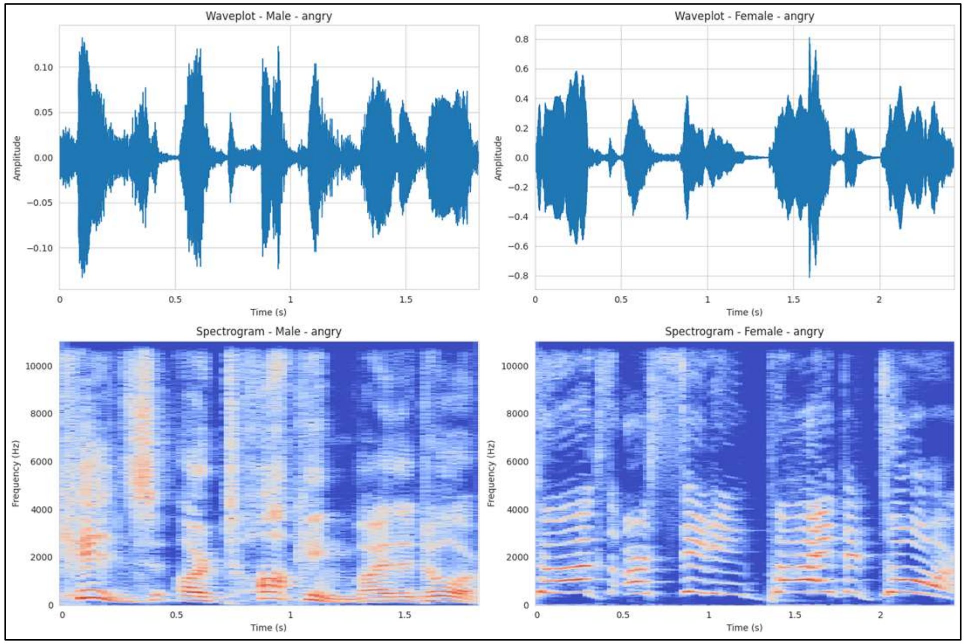


- Disgust – Male/Female

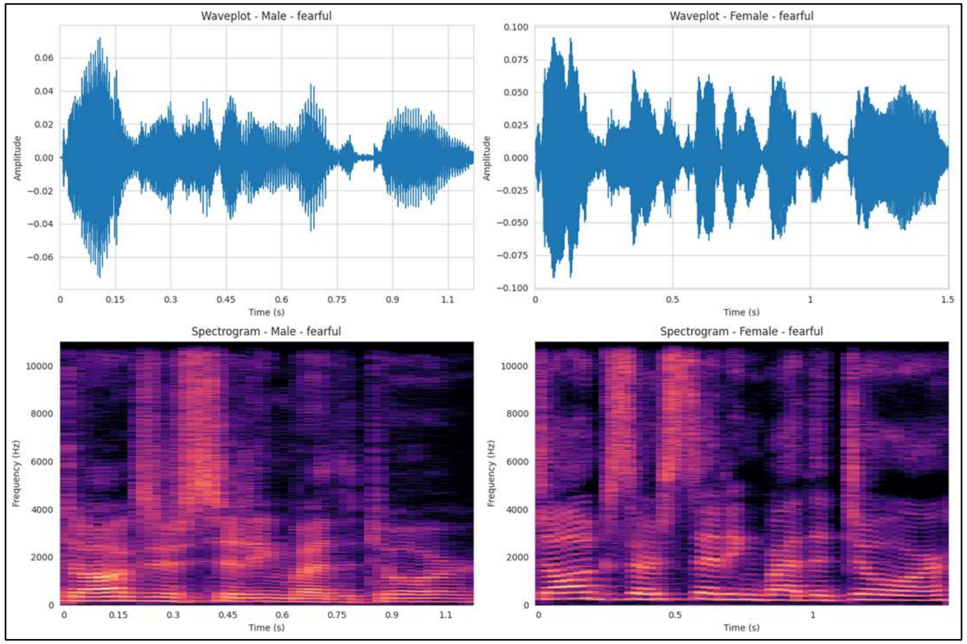


3. Exploratory Data Analysis – Wave plot & Spectrogram

- Angry – Male/Female



- Fearful – Male/Female



4. Feature Extraction

5. Standardized Sample Rate (all audio files) – 16Hz
6. Padding/Truncating – 2sec (which means 32,000 samples)

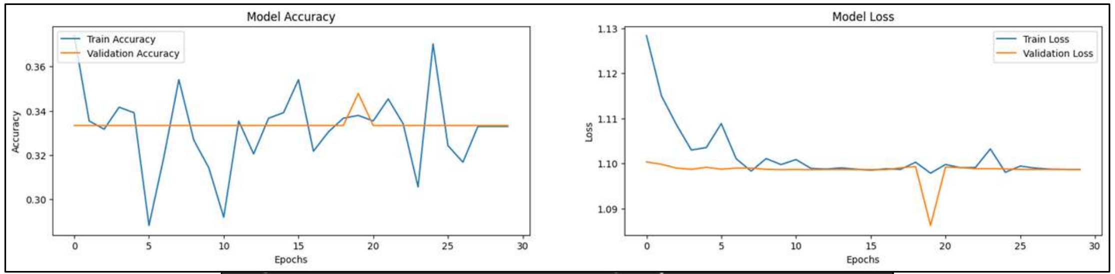
Reason for extra preprocessing before feature extraction:

- Preserve Speech Features. 16kHz provides enough information to capture the important speech characteristics while keeping the data manageable.
- Standardized audio length. Dataset average=1.64sec, min=0.80sec, max=3.34sec.
  - If audio <2sec: Pad with zeros.
  - If audio >2sec: truncate to 2sec.

```python
# Define parameters
TARGET_SR = 16000  # Fixed Sampling Rate (16kHz)
TARGET_DURATION = 2.0  # Fixed duration in seconds
TARGET_SAMPLES = int(TARGET_SR * TARGET_DURATION)  # 32000 samples

# Extract Features
mfcc_features = []
spec_features = []
labels = []

# Process each audio file
for index, row in tqdm(df_BalMap.iterrows(), total=len(df_BalMap)):
    file_path = row['Filepath']
    label = row['Label']

    # Load audio
    data, sr = librosa.load(file_path, sr=TARGET_SR)  # Resample to 16kHz

    # Pad/Truncate audio to 2 seconds (32000 samples)
    if len(data) < TARGET_SAMPLES:
        data = np.pad(data, (0, TARGET_SAMPLES - len(data)), mode='constant')
    else:
        data = data[:TARGET_SAMPLES]

    # ---- MFCC Extraction ----
    mfcc = librosa.feature.mfcc(y=data, sr=TARGET_SR, n_mfcc=40)  # (40, time_frames)
    mfcc_features.append(mfcc)

    # ---- Spectrogram Extraction ----
    spectrogram = librosa.feature.melspectrogram(y=data, sr=TARGET_SR, n_mels=128)  # (128, time_frames)
    spec_features.append(spectrogram)

    # Store labels
    labels.append(label)
```

5. Model Deployment – Model 1 (CNN + LSTM)

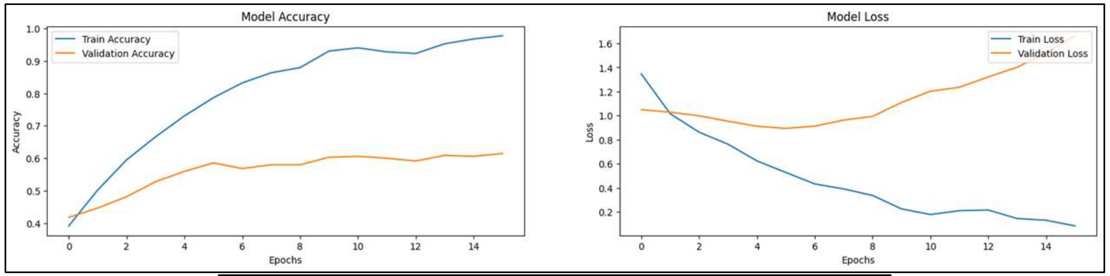
- **Outcome:**
  - Accuracy: Training = 33.74%, Validation = 33.33%.
  - Issue: The model's accuracy is stuck, and the loss remains constant (~1.0986), which suggests the model is making random guesses (log(3) ≈ 1.0986 for a 3-class classification).
- **Possible Causes:**
  - SoftMax outputs stuck: The model is not learning properly.
  - Architecture/Vanishing Gradients: The model might have inadequate architecture or suffer from vanishing gradients.




```
Train Data Classification Report:
------------------------------------------------------------
              precision    recall  f1-score   support

     Neutral       0.00      0.00      0.00       268
    Positive       0.33      1.00      0.50       268
    Negative       0.00      0.00      0.00       269

    accuracy                           0.33       805
   macro avg       0.11      0.33      0.17       805
weighted avg       0.11      0.33      0.17       805


Validation Data Classification Report:
------------------------------------------------------------
              precision    recall  f1-score   support

     Neutral       0.00      0.00      0.00       115
    Positive       0.33      1.00      0.50       115
    Negative       0.00      0.00      0.00       115

    accuracy                           0.33       345
   macro avg       0.11      0.33      0.17       345
weighted avg       0.11      0.33      0.17       345
```

5. Model Deployment – Model 2 (Enhanced CNN + LSTM)

- **Modifications:**
  - Increased number of LSTM layers and units to improve time-series feature extraction.
  - Added Batch Normalization to stabilize training and improve gradient flow.
  - Added Dropout (30%) to prevent overfitting and ReLU activation before SoftMax to improve learning in deeper networks.
- **Outcome:**
  - Training Accuracy = 97.18%, Validation Accuracy = 58.84%.
  - Issue: The model shows signs of overfitting, with a large gap between training and validation accuracy.



Train Data Classification Report:

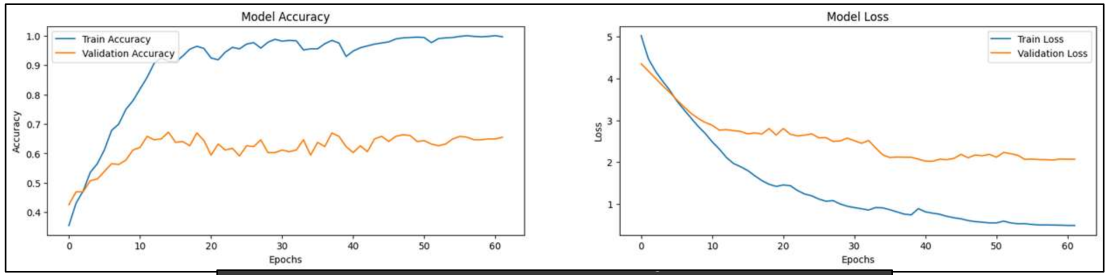
	precision	recall	f1-score	support
Neutral	1.00	1.00	1.00	268
Positive	1.00	1.00	1.00	268
Negative	1.00	1.00	1.00	269
accuracy			1.00	805
macro avg	1.00	1.00	1.00	805
weighted avg	1.00	1.00	1.00	805

Validation Data Classification Report:

	precision	recall	f1-score	support
Neutral	0.50	0.53	0.51	115
Positive	0.70	0.73	0.71	115
Negative	0.66	0.58	0.62	115
accuracy			0.61	345
macro avg	0.62	0.61	0.62	345
weighted avg	0.62	0.61	0.62	345

5. Model Deployment – Model 3 (Optimized CNN + LSTM) = Baseline

- Modifications:
  - Added L2 regularization to LSTM layers to penalize large weights and help reduce overfitting.
  - Introduced a learning rate scheduler that reduces the learning rate by a factor of 0.5 if validation loss doesn't improve for 5 epochs.
  - Increased Dropout rate slightly (from 0.3 to 0.4 or 0.5) to help with generalization.
- Outcome:
  - Training Accuracy = 99.67%, Validation Accuracy = 65.51%.
  - Improvement: The model shows better generalization with improved validation accuracy



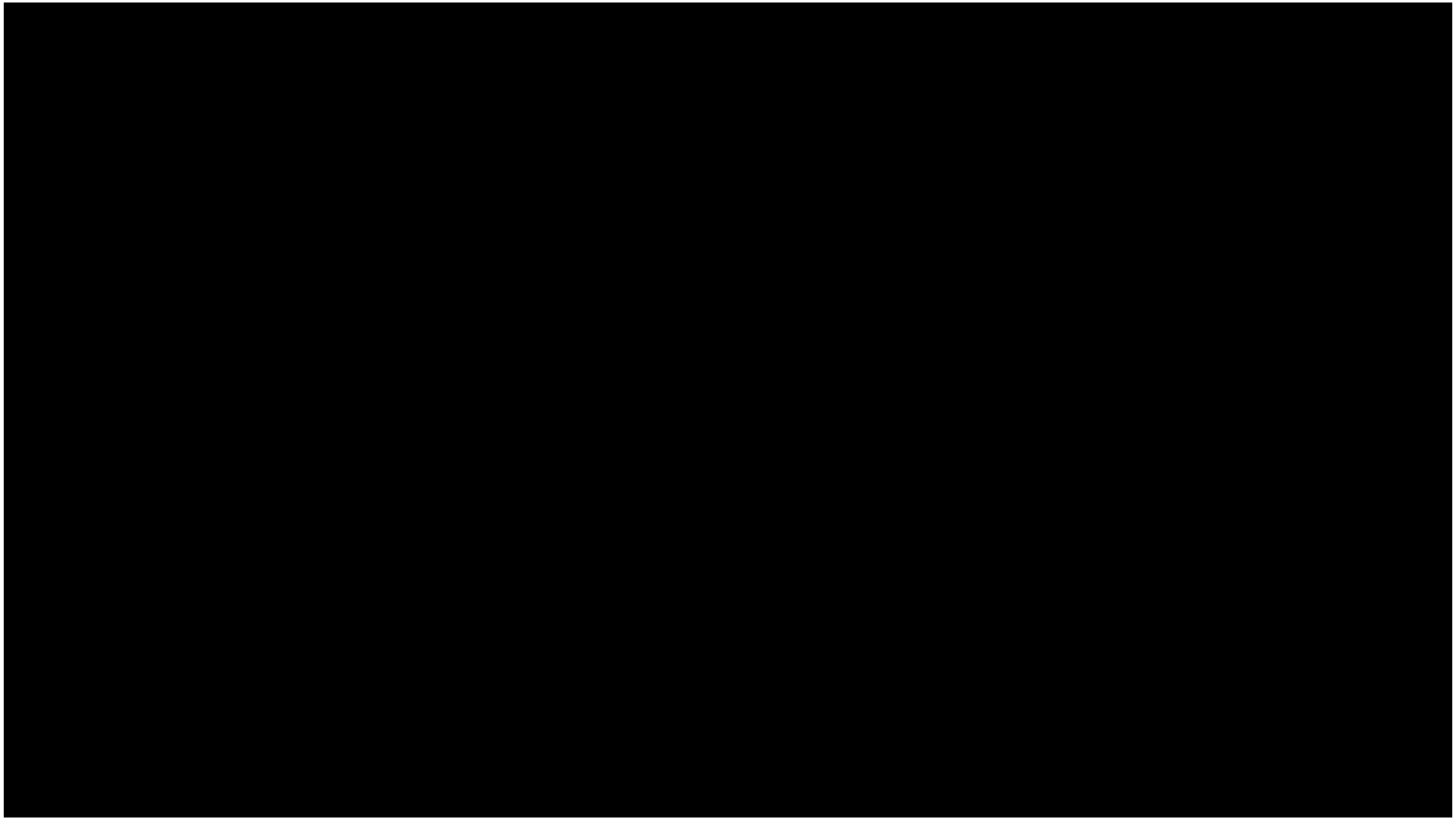

```
Train Data Classification Report:
-------------------------------------------------------
              precision    recall  f1-score   support

     Neutral       1.00      1.00      1.00       268
    Positive       1.00      1.00      1.00       268
    Negative       1.00      1.00      1.00       269

    accuracy                           1.00       805
   macro avg       1.00      1.00      1.00       805
weighted avg       1.00      1.00      1.00       805


Validation Data Classification Report:
-------------------------------------------------------
              precision    recall  f1-score   support

     Neutral       0.50      0.53      0.51       115
    Positive       0.70      0.73      0.71       115
    Negative       0.66      0.58      0.62       115

    accuracy                           0.61       345
   macro avg       0.62      0.61      0.62       345
weighted avg       0.62      0.61      0.62       345
```

5. Model Deployment – Model 4 (Transformer Wav2Vec)

- Successfully executed the same preprocessing steps as the custom architecture models (using CNN & LSTMs).
- Loaded "facebook/wav2vec2-base" transformer model from Hugging Face.
- Unfortunately, encountered troubles during training due to input size. Unable to complete debugging of issue during the course of the project timeline.

```
Epoch 1/100 starting...
Epoch 1/100:   0%|          | 0/26 [00:00<?, ?it/s]<ipython-input-16-c31dac6cb5c4>:17: UserWarning: To copy construct from a tensor, it is recommended to use sourceTenso
  waveform = torch.tensor(self.dataframe.iloc[idx]['Processed Audio'], dtype=torch.float32)
It is strongly recommended to pass the ``sampling_rate`` argument to this function. Failing to do so can result in silent errors that might be hard to debug.
Batch 1: Input values shape: torch.Size([32, 32000])
Epoch 1/100:   4%|▍         | 1/26 [00:01<00:35,  1.42s/it]It is strongly recommended to pass the ``sampling_rate`` argument to this function. Failing to do so can resu
Batch 2: Input values shape: torch.Size([32, 32000])
Epoch 1/100:   8%|▊         | 2/26 [00:02<00:30,  1.27s/it]
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-24-2d61a30cac37> in <cell line: 0>()
      1 # Training the model and getting losses and accuracies
----> 2 training_losses, validation_losses, train_accuracies, val_accuracies = train_model(model, train_dataloader, val_dataloader, optimizer, loss_fn, epochs=100)
      3 
      4 # Display Training Loss and Accuracy
      5 display_training_loss_accuracy(training_losses, validation_losses, train_accuracies, val_accuracies, epochs=10)

9 frames
/usr/local/lib/python3.11/dist-packages/torch/utils/data/_utils/collate.py in collate_tensor_fn(batch, collate_fn_map)
    270         storage = elem._typed_storage()._new_shared(numel, device=elem.device)
    271         out = elem.new(storage).resize_(len(batch), *list(elem.size()))
--> 272     return torch.stack(batch, 0, out=out)
    273 
    274 

RuntimeError: stack expects each tensor to be equal size, but got [32000] at entry 0 and [2, 32000] at entry 3
```

6. Application Deployment (Gradio & Hugging Face)



End.

Thank you.