Project Report

Emotion Detection in Speech
DL LAB

- Nayavanth Yogi
220968344
DSE B

Objective:

Build a plugin/website that uses deep learning to analyse customer service call recordings and evaluate the emotional state of the customer. The plugin will assign a score to the customer service executive based on how well they de-escalate the customer's emotional distress. Intended to be used as a performance metric for customer care executives.

Dataset:

- **CREMA-D Dataset**: Use the CREMA-D dataset to train the model. This dataset contains audio recordings with various emotional labels (anger, disgust, fear, happy, neutral, sadness).
    - CREMA-D is a popular dataset for emotion recognition in speech. It has 6 classes(**Anger, Disgust, Fear, Happy, Neutral, Sadness**) and over 7000 clips
    - It uses 91 different actors with half of them being men and half women and their ages ranging from 20 to 74. It is perfect for building a generalized model
    - It consists solely of .wav files

Data Preprocessing:

- **Feature Extraction**:
    - Extract features such as Mel-Frequency Cepstral Coefficients (MFCCs) or mel-spectrograms from the audio.
    - Normalize the features to ensure consistent input to the model.
- **Data Augmentation**: Apply techniques like time-stretching to ensure that all training data is of the length ‘3 seconds’
- **Emotion Labeling**: Each segment is labeled with the corresponding emotion (e.g., anger, happiness, sadness) based on the **CREMA-D** annotations.

Modelling:

1. Convolutional Neural Network(Baseline model):

Architecture Overview

The emotion recognition model utilizes a **Convolutional Neural Network (CNN)** architecture designed for sequential audio data:

1. **Input Layer**: Accepts input sequences shaped as [32(number of features in 3 seconds of audio), 1], where features are extracted from audio signals.
2. **Convolutional Layers**:
    - **Two Layers**: The first layer has **32 filters** (kernel size = 3), and the second has **64 filters** (kernel size = 3), both using ReLU activation to capture local patterns.
3. **Pooling Layers**: Each convolutional layer is followed by a **MaxPooling1D** layer (pool size = 2) to reduce dimensionality and retain important features.
4. **Dense Layers**: After flattening the pooled output, a **Dense layer** with **128 units** processes the features, followed by an **output layer** with **6 units** (softmax activation) for multi-class classification.

Model Compilation

The model is compiled using the **Adam optimizer** and **sparse categorical crossentropy** as the loss function, with accuracy as the evaluation metric.

Training Process

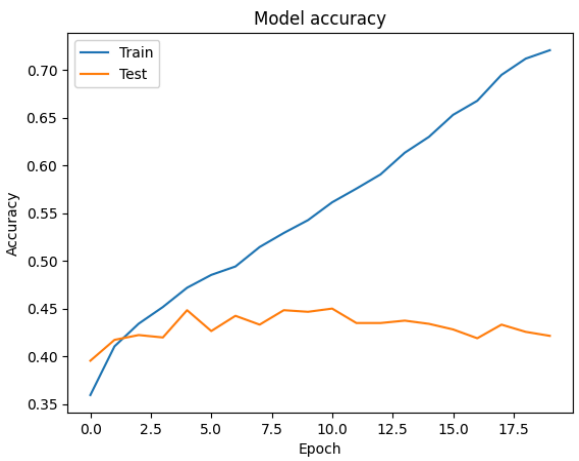
The model is trained for **20 epochs** with a batch size of **32**, using **20% of the training data** for validation.

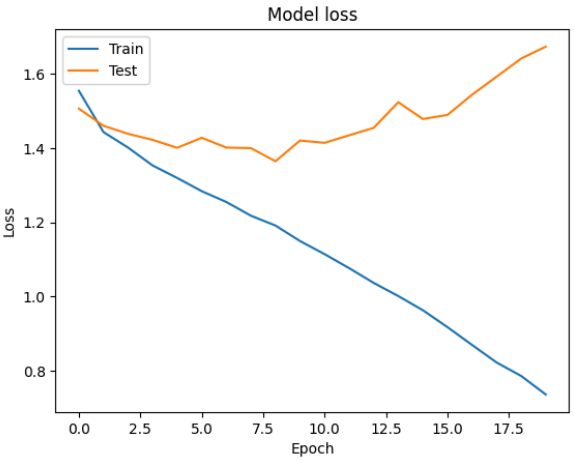
Test Accuracy

Upon evaluation, the model achieved:

- **Test Accuracy**: **43.7%**
- **Test Loss**: **1.5470**

This accuracy indicates that the model correctly classified approximately 43.7% of the test samples. While this serves as a baseline, further enhancements, such as model tuning and data augmentation, are needed for improved performance.





2. Transformer

Architecture Overview

The transformer model for emotion recognition consists of the following key components:

1. **Input Layer**: The model accepts input sequences of shape `(sequence_length, feature_dimension)`.
2. **Positional Embedding Layer**: This layer incorporates both token and positional embeddings to maintain the order of the input sequence, projecting features into a higher-dimensional space.
3. **Transformer Block**:
    - Implements **multi-head self-attention** to capture relationships between different parts of the input sequence.
    - Contains a **feed-forward network** to further process the attention output.
    - Utilizes **layer normalization** and **dropout** for regularization.
4. **Global Average Pooling Layer**: Reduces the dimensionality of the output from the transformer block, summarizing the features.
5. **Dense Layers**:
    - A fully connected layer with **ReLU activation** to learn non-linear combinations of features.
    - An output layer with **softmax activation** to predict the probabilities of different emotion classes.

Training Results

The model was trained for **30 epochs**, with the following key metrics observed:

- *Final Training Accuracy: 69.6%*
- *Final Training Loss: 0.7989*
- *Final Validation Accuracy: 45.5%*
- *Final Validation Loss: 1.4847*

Despite a noticeable increase in training accuracy, the validation accuracy remains lower, indicating potential overfitting. Additional strategies like regularization, data augmentation, or further tuning of hyperparameters may be required to enhance generalization on unseen data.

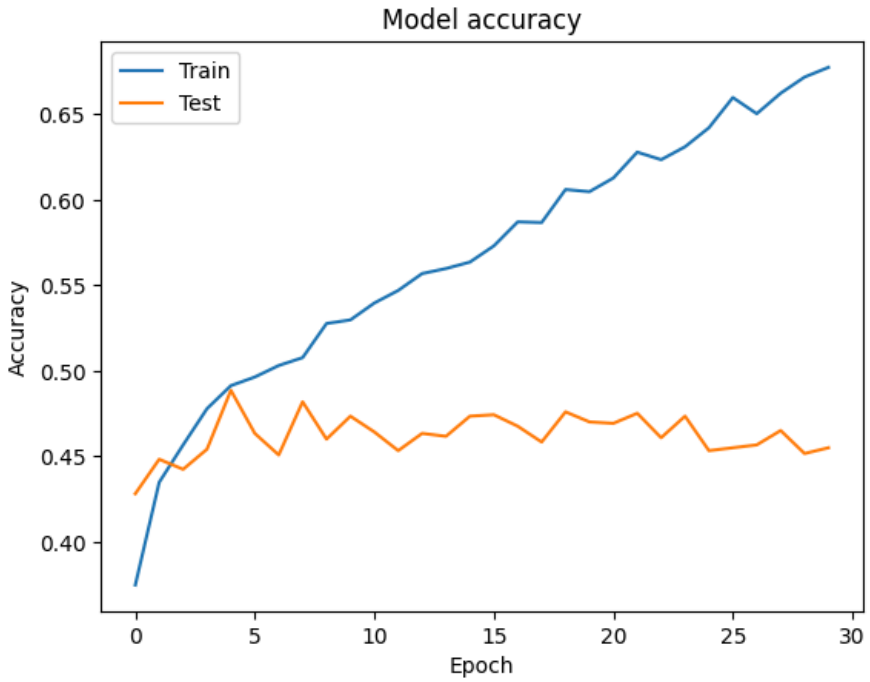
Test Evaluation Results

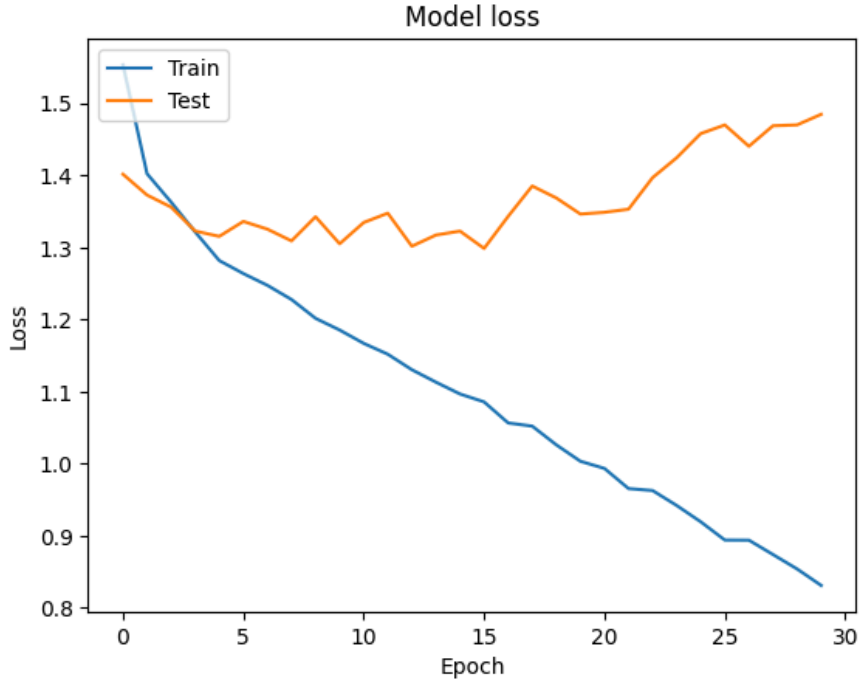
After evaluating the transformer model on the test dataset, the following metrics were obtained:

- **Test Accuracy**: **50.7%**
- **Test Loss**: **1.3658**

Analysis

The test accuracy indicates that the model can correctly classify approximately 50.7% of the samples in the test set. While this is an improvement over the earlier validation performance, it suggests that the model may still benefit from additional training or tuning.


Model accuracy
Train
Test
Accuracy
0.65
0.60
0.55
0.50
0.45
0.40
0
5
10
15
20
25
30
Epoch


Model loss
Train
Test
Loss
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0
5
10
15
20
25
30
Epoch

3. CNN+LSTM(Hybrid Model):

The CNN (Convolutional Neural Network) + LSTM (Long Short-Term Memory) architecture combines the strengths of convolutional and recurrent neural networks, making it effective for tasks involving sequential data with spatial patterns, such as audio classification.

Architecture Overview:

1. CNN Layers: The CNN component learns spatial hierarchies from input data (e.g., spectrograms or MFCCs). Convolutional and pooling layers extract local features, identifying crucial patterns in the audio.
2. LSTM Layers: Following the CNN, LSTM layers capture temporal dependencies, learning long-range patterns that represent the progression of emotions over time.
3. Fully Connected Layers: Dense layers map learned features to target classes, with dropout layers to reduce overfitting.

Fitting the Model:

- Data Preparation: Preprocess and split the dataset into training, validation, and test sets. Apply feature extraction methods like MFCCs.
- Model Compilation: Compile the model using an optimizer (Adam) and a loss function (parse categorical crossentropy).
- Training: Train the model over 30 epochs, monitoring validation metrics to prevent overfitting. Techniques like early stopping are used.

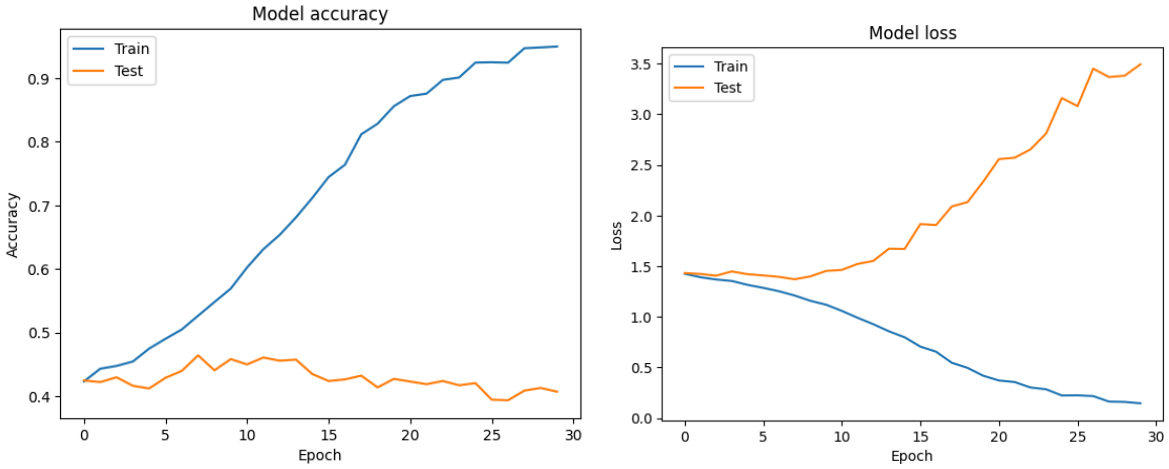
Testing and Evaluation:

- Testing: Evaluate the model on the unseen test dataset.
- Performance Metrics: Calculate accuracy, precision, recall, and F1 score to assess model effectiveness.

Analysis:

- Model Effectiveness: A significant accuracy gap between training and testing indicates overfitting.
- Class Imbalance: Poor performance on specific classes may suggest the need for data balancing techniques.
- Feature Importance: Analyzing predictions can highlight influential features for further improvements.

In summary, the CNN + LSTM architecture provides a robust approach for modeling sequential audio data, enabling effective emotion detection through careful fitting, testing, and analysis.



4. Wav2vec 2.0 pretrained model:

Model Architecture

The Wav2Vec2 model is built on a transformer architecture and is specifically designed for processing raw audio waveforms. Key features include:

- **Self-Supervised Learning**: The model is pre-trained on a large amount of unlabeled audio data, allowing it to learn meaningful representations of speech before fine-tuning on a specific task.
- **Processor Integration**: The Wav2Vec2 processor is used to convert audio inputs into a format suitable for the model, including handling various sampling rates and padding/trimming audio sequences to a fixed length.
- **Sequence Classification**: The model is adapted for multi-class classification tasks, with the last layer producing logits corresponding to the predicted probabilities for each emotion class.

Training Procedure

The model was trained using the following setup:

- **Hyperparameters**:
    - **Batch Size**: 4 **Learning Rate**: 0.01 **Epochs**: 10
- **Optimizer**: AdamW was used for weight optimization, which is well-suited for transformer models due to its adaptive learning rate capabilities.
- **Gradient Clipping**: To prevent exploding gradients, we implemented gradient clipping with a maximum norm of 1.0.

The training loop involved forward and backward passes through the model, with the loss calculated using the outputs (logits) and the corresponding labels. The average loss was reported every 10 batches to monitor training progress.

Results and Analysis

At the conclusion of the training phase, the model was saved for future use.

- **Loss Observations**: The training loss exhibited a downward trend over epochs, indicating that the model was learning effectively.
- **Model Performance**: Although the accuracy was not specifically measured during training, it is essential to evaluate the model on a separate validation or test set to gauge its real-world applicability and performance.

Conclusion

The Wav2Vec2 model demonstrates significant potential for emotion recognition tasks in audio processing due to its powerful architecture and pre-training capabilities.

Comparison

Models	Test Accuracy	Test Loss	Train Accuracy	Val Accuracy
CNN	**0.43**	**1.54**	**0.74**	**0.42**
Transformer	**0.51**	**1.36**	**0.69**	**0.45**
CNN+LSTM	**0.43**	**3.37**	**0.95**	**0.40**
Wav2vec	**0.14**	**5.32**	**2.11**	**-**

Deployment

The deployment consists of five main Python files, each serving a distinct purpose in the workflow:

1. **Train Data Preprocessing (`train_data_preprocessing.py`)**
    - **Functionality**: This script is responsible for preparing the audio data for training. It involves:
        - Loading audio recordings.
        - Extracting features such as Mel-Frequency Cepstral Coefficients (MFCC), chroma, and spectral contrast, which are crucial for capturing the emotional nuances in speech.
        - Padding and trimming audio to ensure uniform input length.
        - Mapping emotion labels to the corresponding audio samples.
        - Scaling features to normalize the data, improving model training efficiency.
        - Saving the preprocessed data for later use.
2. **Inference Data Preprocessing (`inference_data_preprocessing.py`)**
    - **Functionality**: This script prepares audio files for inference, ensuring that the model receives the correct input format:
        - It loads a specified audio file.
        - Segments the audio into 3-second intervals, allowing for detailed analysis of emotional transitions throughout the call.
        - Extracts MFCC features for each segment and scales them using a pre-trained scaler.
        - Returns the processed data ready for prediction.
3. **Inference (`inference.py`)**
    - **Functionality**: This script implements the model's inference process:
        - It loads the pre-trained TensorFlow model (`emotion_recognition_model.keras`).

        - Utilizes the `inference_data_preprocessing.py` script to process the audio segments.
        - Predicts the emotional state for each segment, scoring the executive based on how well they de-escalate the customer's emotional state.
        - Implements a scoring system that adjusts based on emotional transitions, ensuring scores remain within a defined range (1 to 10) for effective evaluation.
4. **Model Definition and Training (`model.py`)**
    - **Functionality**: This script is dedicated to defining, training, and saving the transformer model used for emotion recognition:
        - It defines the architecture of the transformer model, including transformer blocks and positional embeddings.
        - The model is trained using the preprocessed training data.
        - After training, the model is evaluated on test data to ensure performance and generalization.
        - The trained model is saved in the `.keras` format for deployment.
5. **Web Application (`app.py`)**
    - **Functionality**: This script creates a Streamlit web application for customer care emotion analysis:
        - Users can upload a WAV audio file of a customer service call.
        - Upon file upload, the audio is temporarily saved, and inference is run using the `run` function from .
        - The application displays the executive score based on the emotional analysis.
        - It visualizes emotion fluctuations over time, classifying emotions into positive and negative categories. The visualization uses color coding (green for positive emotions and red for negative) and provides a smoothed line representing emotion classification over time.

Deployment Process

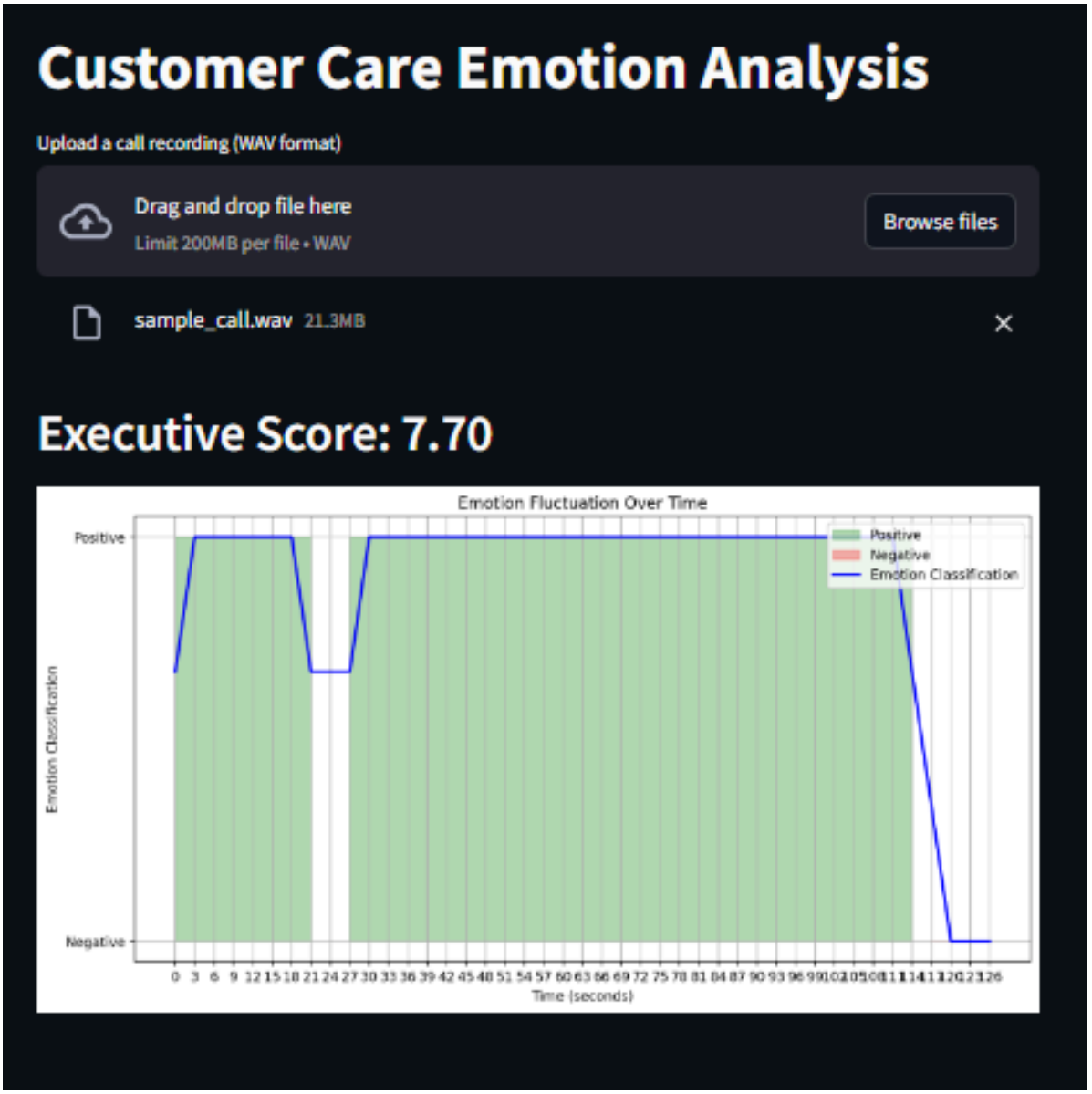
To deploy the emotion detection model, follow these steps:

1. **Prepare the Environment**: Ensure all necessary libraries, including TensorFlow, Streamlit, and any audio processing libraries, are installed in your deployment environment.
2. **Preprocess Training Data**: Execute the `train_data_preprocessing.py` script to prepare the training data.
3. **Train the Model**: Run the `model.py` script to train the transformer model and save it.
4. **Inference Setup**: For real-time or batch inference, run the `inference.py` script with the desired audio files to analyze.
5. **Web Application Launch**: Run the `app.py` script to launch the Streamlit application, allowing users to interactively upload audio files and visualize the emotional analysis.

6. **Monitoring and Evaluation**: Continuously monitor the model's performance during inference and web application usage, adjusting the scoring system as necessary to ensure accuracy and reliability in emotional assessment.

Conclusion

This deployment framework enables effective analysis of customer service calls, providing valuable insights into emotional dynamics and the executive's response effectiveness. By leveraging deep learning techniques and a user-friendly web interface, organizations can enhance customer interactions and improve service quality.



The project successfully developed a deep learning-based emotion recognition system, allowing organizations to assess customer interactions more effectively. The model demonstrates significant potential for classifying emotions in customer service scenarios, offering insight into executive performance and customer satisfaction. Despite the challenges in model accuracy, particularly on test data, this solution lays a solid foundation for future refinements.