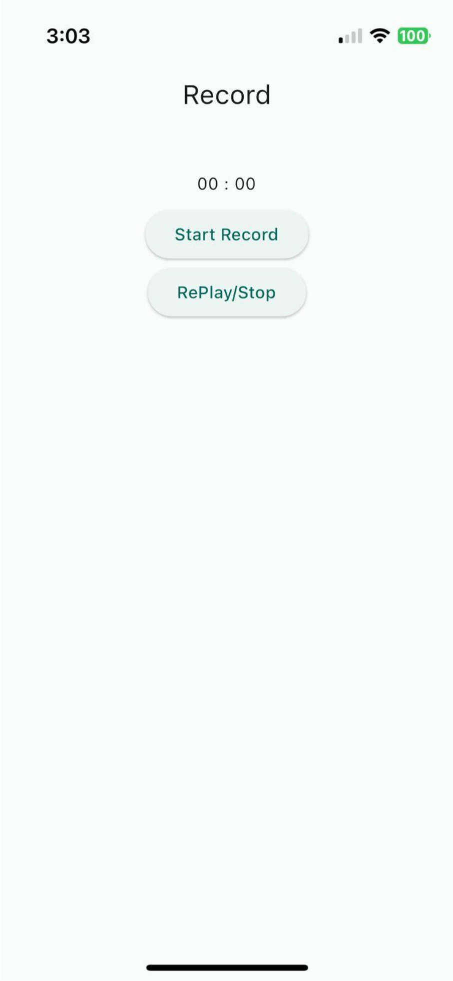
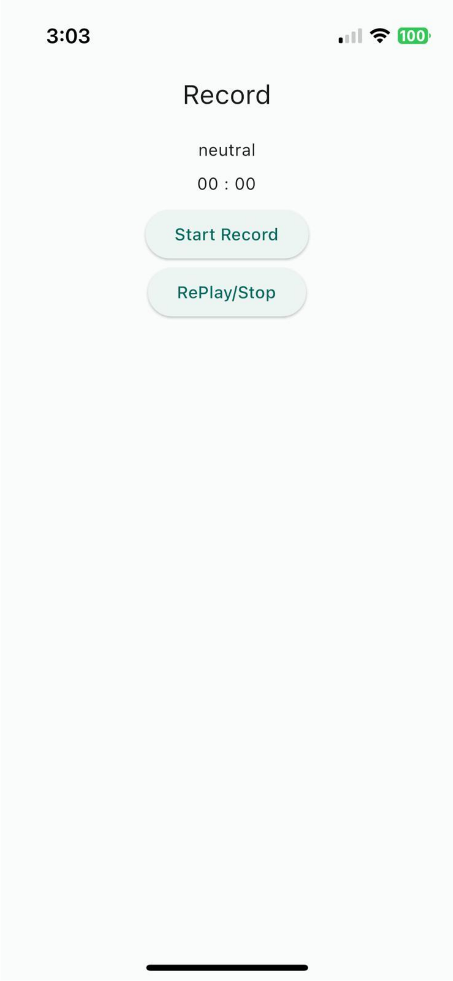
CS 582 - Machine Learning
Audio Sentiment Analysis

1. Project Idea

Audio sentiment analysis involves analyzing the emotional tone or sentiment conveyed in an audio recording, like speech or music. It uses computational methods and algorithms to automatically detect and classify the sentiment expressed in the audio. We aim to examine the emotional tone of the audio we capture for our project. Our goal is to determine whether the tone is positive, negative, or neutral.





2. Dataset for Audio Sentiment Analysis

There are several datasets for audio analysis, but we used the one from ***Kaggle*** because there are total of 390 audio files which are fully labelled and each audio file is neatly cleaned and pre-processed.

3. BiDirectional Long Short-term Memory for Audio Sentiment Analysis

For this project idea, there are several deep-learning models which are researched by most beginner and pre-intermediate machine-learning engineers.

Even TensorFlow gives the guideline for audio data analysis. However, the way TensorFlow want us to do is, firstly to extract the features from the audio data and then convert it to spectrogram and perform classification based on spectrogram images.

Nonetheless, in our opinion, we want to do classification or analysis based directly on audio features. That's why we have come up with this brand new idea and we have found out that the result is pretty good and even better than other approaches.

4. Platform and library

We chose the TensorFlow library with Keras backend because it provides many deep-learning algorithms natively and we can also take advantage of GPUs without configuring some complicated steps.

5. Number of Classes

Our project only solves three-class classification problems, namely, positive, negative and neutral. Since we have researched fine-tuned model for this three-class classification problems, we are positive that our approach can be applied to multiclass classification problems with a little bit of hyperparameter tuning.

6. Real-world problems (Phone Call Support for Customers)

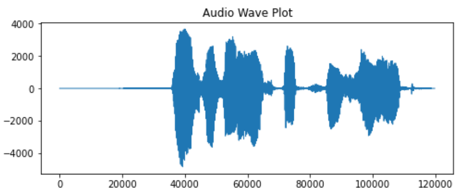
Audio Sentiment Analysis can be used for customer support to analyze the behaviour and emotions of both support personnel and customers during interactions. This analysis helps identify areas for improvement in customer service.

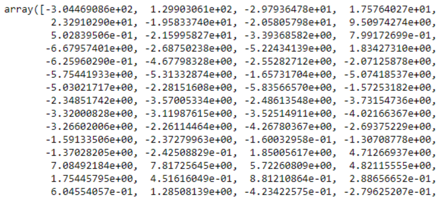
Example

During customer support phone calls, the tone of the conversation can vary from negative to positive, neutral to positive, positive to neutral, and so on.

7. Feature Extraction

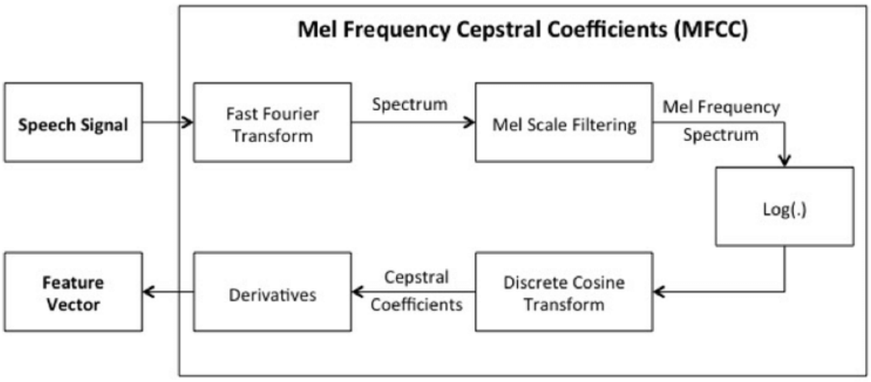
MFCC (Mel Frequency Cepstral Coefficients) is commonly used for audio feature extraction due to its effectiveness in representing the spectral characteristics of human speech and other audio signals.





```
array([-3.04469086e+02,  1.29903061e+02, -2.97936478e+01,  1.75764027e+01,
        2.32910290e+01, -1.95833740e+01, -2.05805798e+01,  9.50974274e+00,
        5.02839506e-01, -2.15995827e+01, -3.39368582e+00,  7.99172699e-01,
       -6.67957401e+00, -2.68750238e+00, -5.22434139e+00,  1.83427310e+00,
       -6.25960290e-01, -4.67798328e+00, -2.55282712e+00, -2.07125878e+00,
       -5.75441933e+00, -5.31332874e+00, -1.65731704e+00, -5.07418537e+00,
       -5.03021717e+00, -2.28151608e+00, -5.83566570e+00, -1.57253182e+00,
       -2.34851742e+00, -3.57005334e+00, -2.48613548e+00, -3.73154736e+00,
       -3.32000828e+00, -3.11987615e+00, -3.52514911e+00, -4.02166367e+00,
       -3.26602006e+00, -2.26114464e+00, -4.26780367e+00, -2.69375229e+00,
       -1.59133506e+00, -2.37279963e+00, -1.60032958e-01, -1.30708778e+00,
       -1.37028205e+00, -2.42508829e-01,  1.85005617e+00,  4.71266937e+00,
        7.08492184e+00,  7.81725645e+00,  5.72260809e+00,  4.82115555e+00,
        1.75445795e+00,  4.51616049e-01,  8.81210864e-01,  2.88656652e-01,
        6.04554057e-01,  1.28508139e+00, -4.23422575e-01, -2.79625207e-01,
```

Block diagram of MFCC



Librosa Library to implement MFCC

One popular Python library for calculating MFCC is Libross. Librosa is a widely used audio processing library that provides various functionalities for audio analysis, including feature extraction like MFCCs.

8. Algorithms we have researched and comparison

As we mentioned above, we have used BiLSTM over any other deep-learning algorithms. Actually, we first come up with LSTM but there are some issues and we finally decided to use BiLSTM. We would like to discuss some main advantages of BiLSTM.

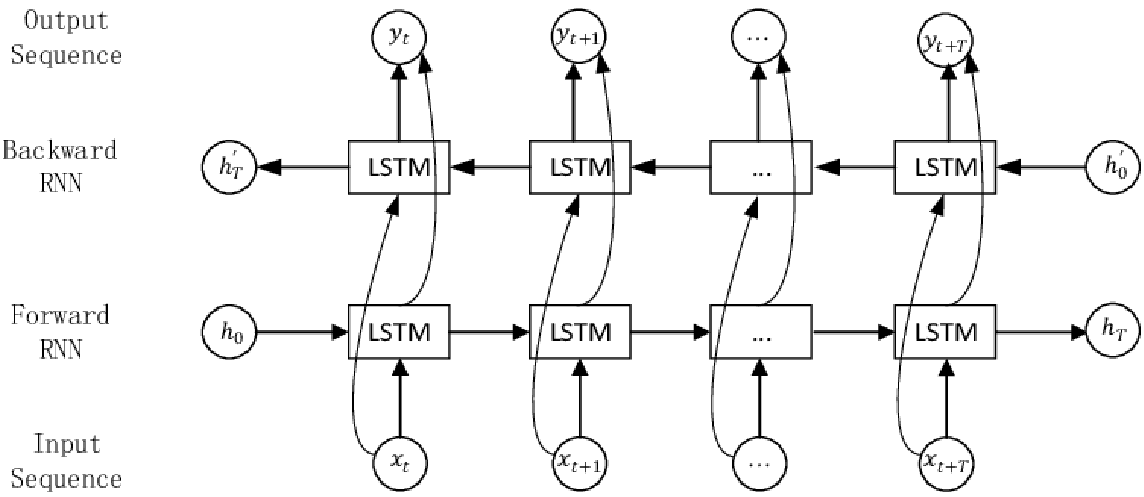
Sequential data analysis: BiLSTM is a neural network architecture designed for analyzing sequential data, such as text or time series.

Bidirectional processing: It consists of two LSTM layers - one processing the sequence in the forward direction and the other in the backward direction.

Capturing context: By processing the sequence in both directions, BiLSTM captures contextual information from both preceding and succeeding elements.

Enhanced understanding: This bidirectional approach enables the model to better understand dependencies and patterns in the sequence.

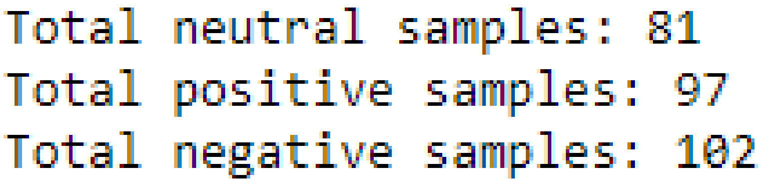
Applications: BiLSTM has been successfully used in various tasks, including sentiment analysis, named entity recognition, and speech recognition, where understanding the entire sequence is crucial.



General Architecture of Bi-LSTM

Why BiLSTM and not LSTM

We have a limited amount of data so we need to use a more complicated network.



```
Total neutral samples: 81
Total positive samples: 97
Total negative samples: 102
```

BiDirectional: Ability to capture both past and future context.

Contextual understanding: BiLSTM captures both past and future context, allowing it to better understand the dependencies and patterns in audio data.

Enhanced feature representation: BiLSTM can effectively represent complex patterns and long-term dependencies in the audio, improving the model's ability to capture nuanced sentiment expressions.

Holistic sentiment inference: Audio sentiment analysis requires considering the entire sequence to understand the emotional context. BiLSTM's bidirectional nature helps capture intonations, variations, and emotional cues, leading to more accurate sentiment inference.

Performance advantage: Empirical studies show that BiLSTM often outperforms unidirectional LSTM models in various natural language processing tasks, indicating its potential for improved performance in audio sentiment analysis.

How do we get more data?

- Data augmentation (changing pitch, speed, …)

- Collect more data

But even if we can manage to do that we still have to analyse the result and of course, the augmented data won't help the model learn that much.

Why not CNN for our project

Common approach: CNN (Convolutional Neural Network) has been widely used for audio analysis tasks, including sentiment analysis, by many developers.

Exploring new approaches: Instead of following the common path, we want to explore alternative methods to gain fresh insights and potentially improve results.

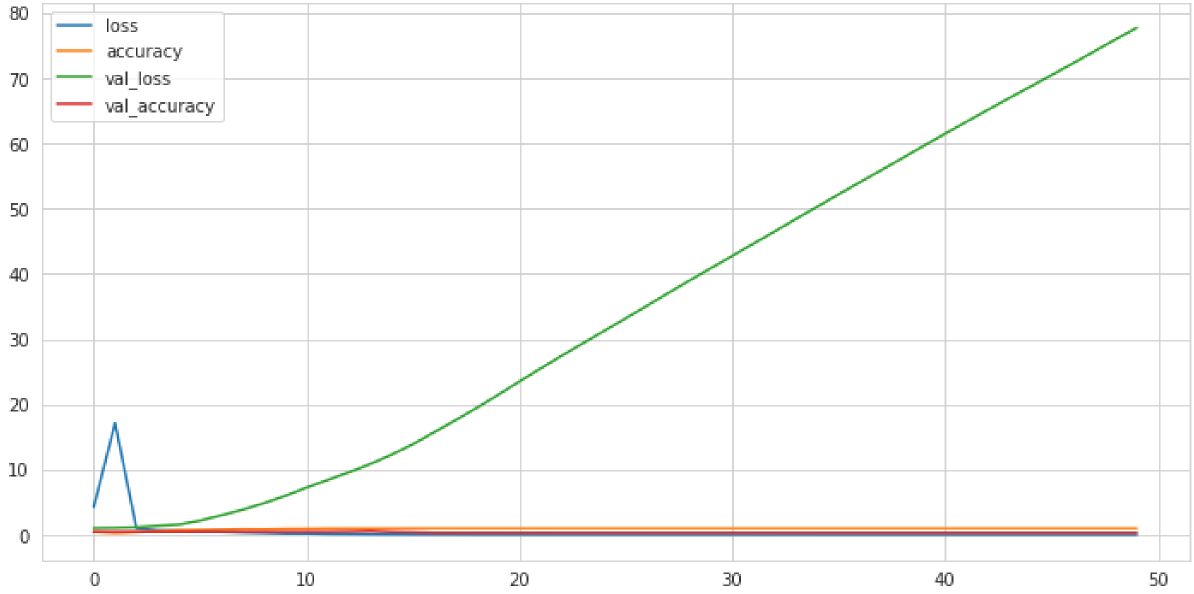
Direct audio processing: Instead of converting audio to a spectrogram (a visual representation of audio), we aim to process audio directly. This approach may preserve fine-grained audio details that could be lost during spectrogram conversion.

Accuracy range: By using the proposed method, we have achieved accuracy levels ranging from 60% to 85%. While this range may vary depending on the dataset and other factors, it demonstrates the potential of our approach.

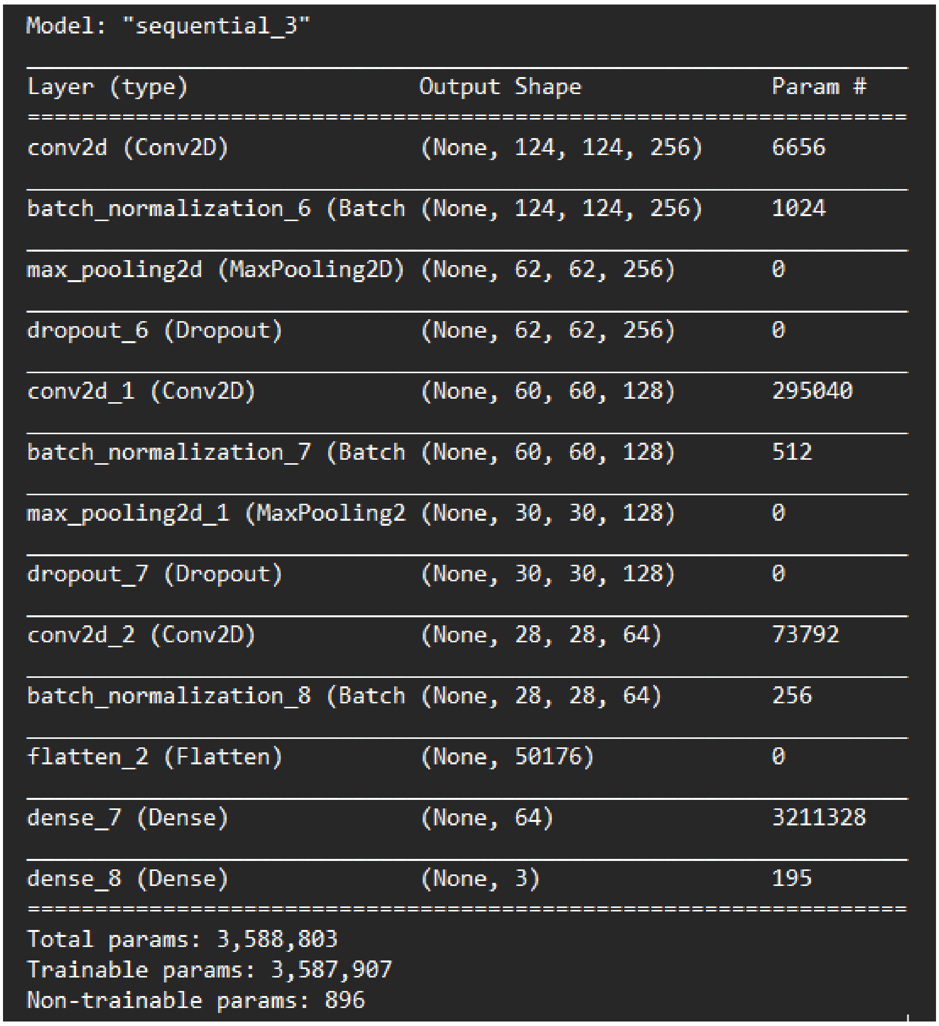
9. Analysis & Results

CNN2D

This is the original work of other people using Convolutional Neural Network with 2D. This test runs with 50 epochs and an early stopping technique was also applied. But unfortunately, as we can see the accuracy looks good but there is a possibility of overfitting.



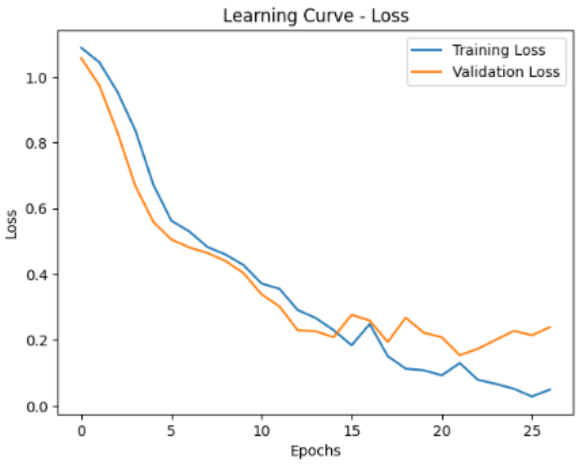
This is the model summary of the above model.

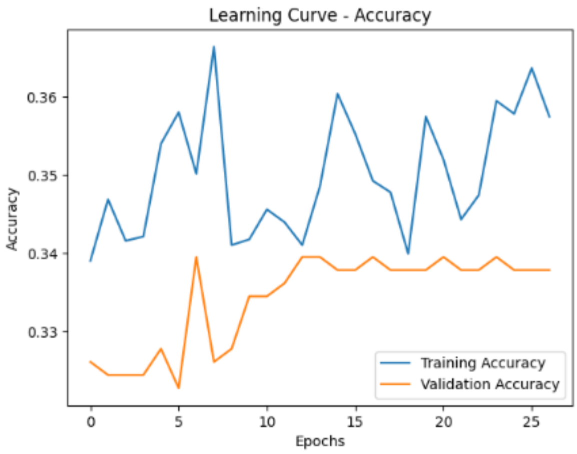


```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 124, 124, 256)     6656
_________________________________________________________________
batch_normalization_6 (Batch (None, 124, 124, 256)     1024
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 62, 62, 256)       0
_________________________________________________________________
dropout_6 (Dropout)          (None, 62, 62, 256)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 60, 60, 128)       295040
_________________________________________________________________
batch_normalization_7 (Batch (None, 60, 60, 128)       512
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 30, 128)       0
_________________________________________________________________
dropout_7 (Dropout)          (None, 30, 30, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 64)        73792
_________________________________________________________________
batch_normalization_8 (Batch (None, 28, 28, 64)        256
_________________________________________________________________
flatten_2 (Flatten)          (None, 50176)             0
_________________________________________________________________
dense_7 (Dense)              (None, 64)                3211328
_________________________________________________________________
dense_8 (Dense)              (None, 3)                 195
=================================================================
Total params: 3,588,803
Trainable params: 3,587,907
Non-trainable params: 896
```

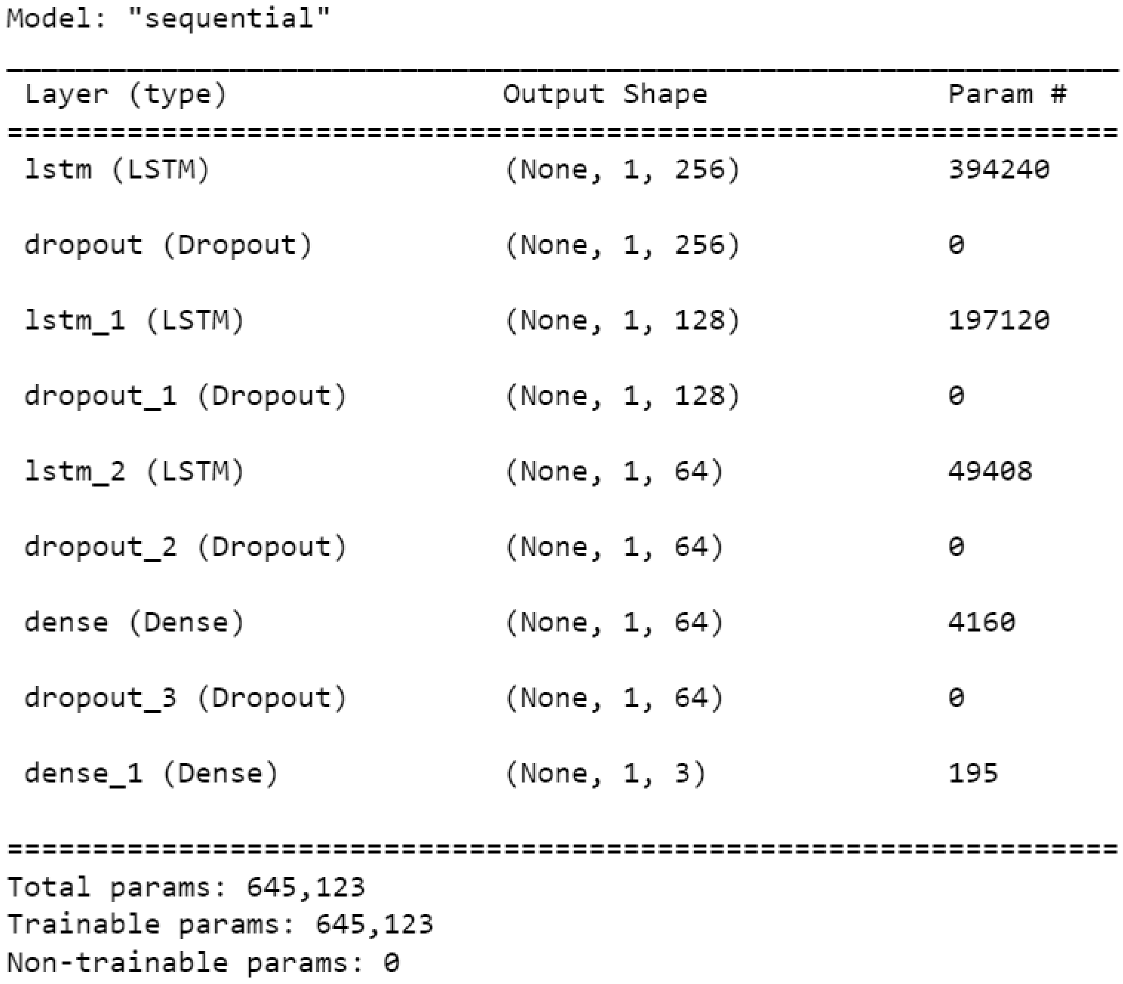
LSTM

When we tested with LSTM, the learning is quite good but the accuracy is worse than CNN2D.





Here is model summary for LSTM.

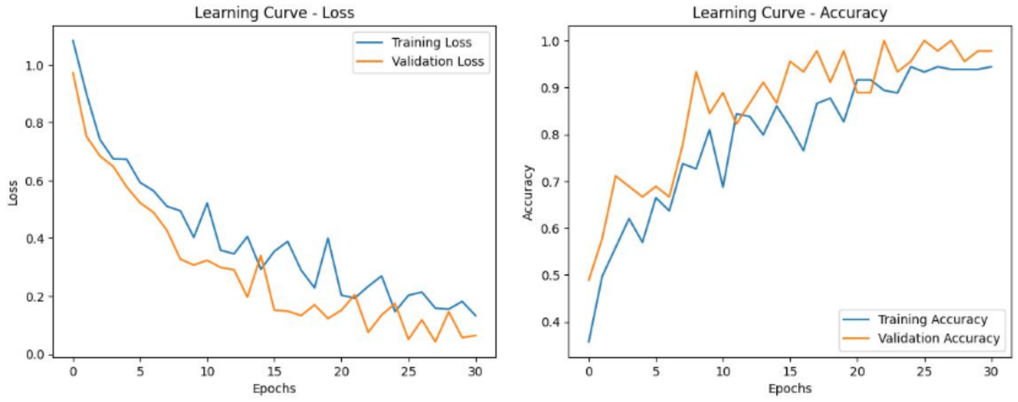


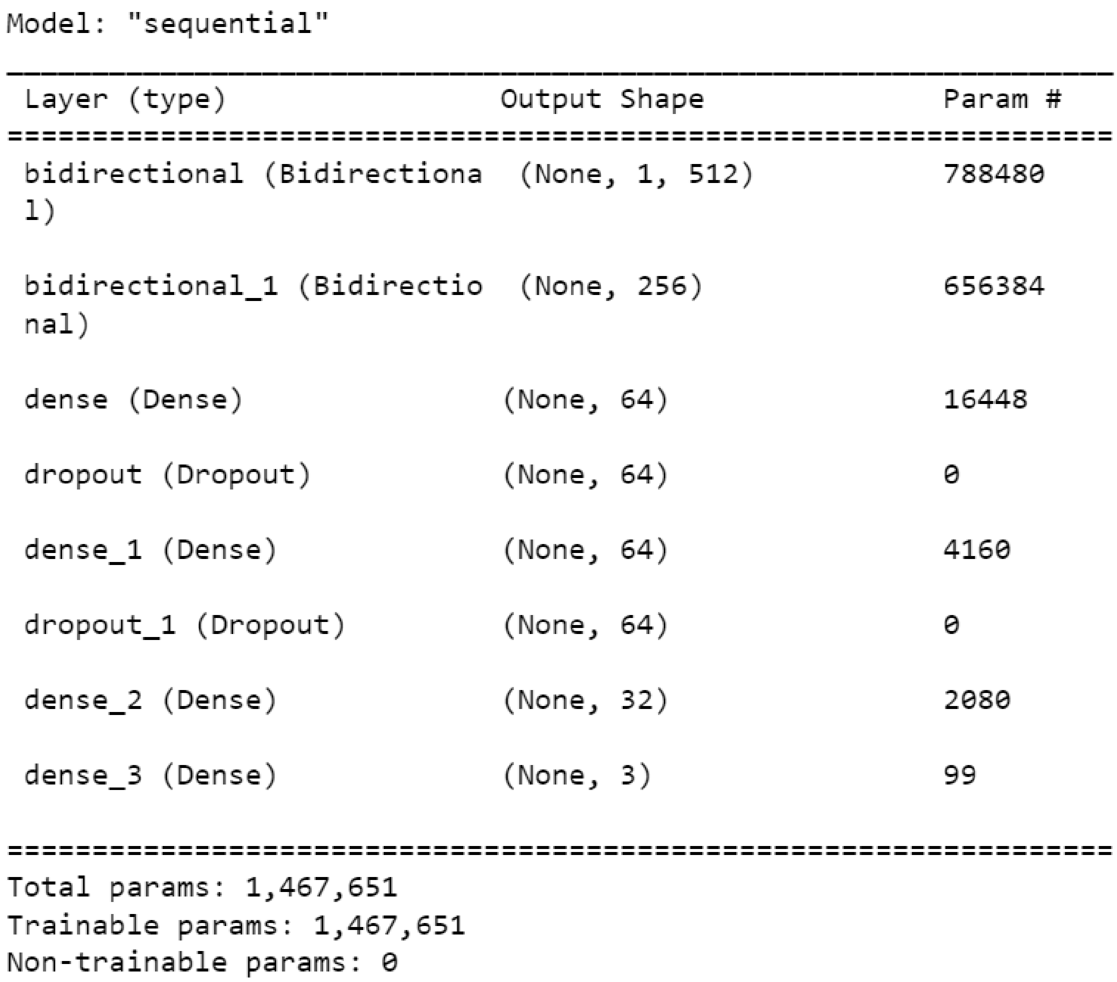
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 1, 256)            394240

 dropout (Dropout)           (None, 1, 256)            0

 lstm_1 (LSTM)               (None, 1, 128)            197120

 dropout_1 (Dropout)         (None, 1, 128)            0

 lstm_2 (LSTM)               (None, 1, 64)             49408

 dropout_2 (Dropout)         (None, 1, 64)             0

 dense (Dense)               (None, 1, 64)             4160

 dropout_3 (Dropout)         (None, 1, 64)             0

 dense_1 (Dense)             (None, 1, 3)              195

=================================================================
Total params: 645,123
Trainable params: 645,123
Non-trainable params: 0
```

BiLSTM

When we used BiLSTM to classify our problem, things look pretty good and the result is amazing. We have used ‘rmsprop: Root Mean Square Propagation’ optimizer and ‘Sparse Categorical Crossentropy’ loss function. We initially tends to have 200 epochs but with an early stopping method, the model learns quickly with only 23 epochs and the result is really good.

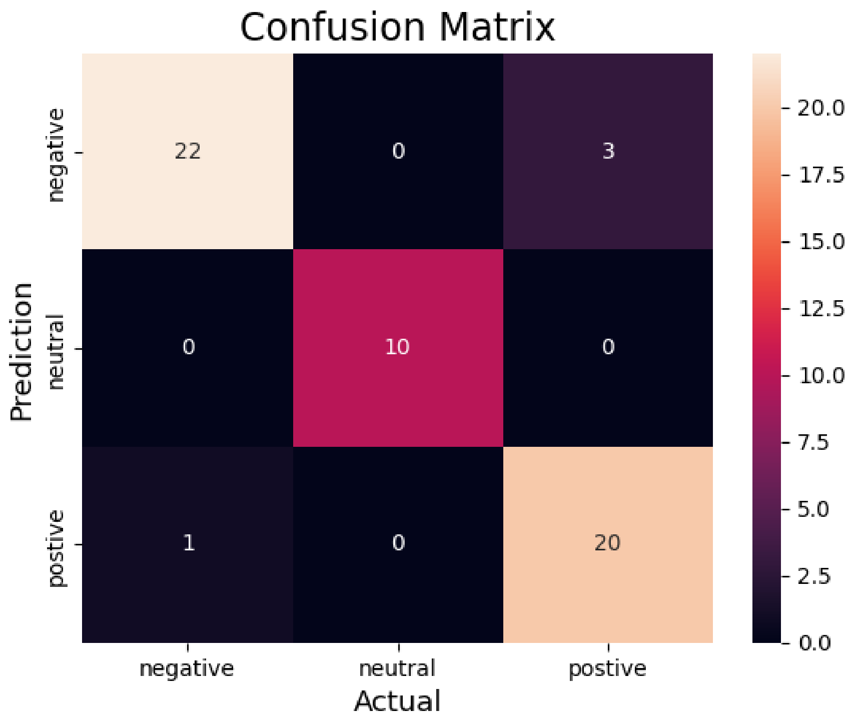
We have got an accuracy of 92.86% for testing with only 0.24 loss. Here is our model summary and learning curve.





```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 bidirectional (Bidirectiona  (None, 1, 512)           788480    
 l)                                                              
                                                                 
 bidirectional_1 (Bidirectio  (None, 256)              656384    
 nal)                                                            
                                                                 
 dense (Dense)               (None, 64)                16448     
                                                                 
 dropout (Dropout)           (None, 64)                0         
                                                                 
 dense_1 (Dense)             (None, 64)                4160      
                                                                 
 dropout_1 (Dropout)         (None, 64)                0         
                                                                 
 dense_2 (Dense)             (None, 32)                2080      
                                                                 
 dense_3 (Dense)             (None, 3)                 99        
                                                                 
=================================================================
Total params: 1,467,651
Trainable params: 1,467,651
Non-trainable params: 0
```

Here is the confusion matrix for our fine-tuned model:



10. Current Issues & Future Works

Currently, we have the following issues:

a. **Accent Issue**
   - Need more data to address accent-related challenges.
   - Improving communication and understanding in multicultural environments.

b. **Noise Reduction**
   - Developing advanced algorithms and techniques to reduce noise in various domains.
   - Enhancing audio technologies to remove unwanted noise.
   - Improving signal-to-noise ratio for better quality and clarity.

If possible, we will try to improve our works by implementing following:

a. **Speech Analysis and Recognition**
   - Determining gender (male/female) from speech signals.
   - Extracting more nuanced emotion information from speech.
   - Advancing speech recognition systems for accurate identification of individuals.

b. **Adding More Data**
   - Collecting and incorporating a larger and more diverse dataset.
   - Expanding the range of accents, languages, and speech patterns.
   - Collaborating with communities and organizations for representative data

11. Related Works

Gender Classification

Neural Network Structure: A Fully Connected Network comprising multiple dense layers

Class Categories: Binary Classification with 2 classes

Source: For more information, please refer to the following resource: "***Gender Recognition by Voice using TensorFlow in Python***"

12. References

- *https://ceur-ws.org/Vol-2328/3_2_paper_17.pdf*
- *https://ieeexplore.ieee.org/abstract/document/6376622*
- *https://arxiv.org/ftp/arxiv/papers/1802/1802.06209.pdf*
- *https://www.hindawi.com/journals/wcmc/2022/4444388/*