



**Specialist Diploma in Applied  
Artificial Intelligence**

Post Diploma Certificate in  
Applied Artificial Intelligence

ITI110

Deep Learning Project

**Group Final Project Report**  
**Team: NAK**

1944757V | Babu Logiah Arun Kumar  
1995149W | Karupaiyan Kalaivani  
6422706H | Nursharinah Binte Sohaimi

# Table of Content

<b>1. Motivation:</b>	<b>2</b>
<b>2. Model Development:</b>	<b>4</b>
2.1 Arun	4
2.1.1 Dataset	4
2.1.1.1 Dataset Collection	4
2.1.1.2 Dataset Description	4
2.1.2 Model Architecture	4
2.1.3 Methods	5
2.1.3.1 LSTM model training with the Airline customer tweet dataset	5
2.1.3.2 Fine tuning the BERT-base model with the Airline customer tweet dataset	5
2.1.4 Performance Tuning & Optimization	5
2.1.4.1 LSTM Model	5
2.1.4.2 BERT Model	6
2.1.5 Application Development	7
2.2 Kalai	8
2.2.1 Dataset Summary	8
2.2.1.1 Dataset	8
2.2.1.2 Dataset Applications	8
2.2.1.3 Airline-wise and Sentiment Count	8
2.2.1.4 Feature Engineering	9
2.2.2 Performance Tuning & Optimization	9
2.2.2.1 LSTM	9
2.2.2.2 Transformers	10
2.2.2.3 BERT	11
2.2.3 Application Development	12
2.3 Sha	13
2.3.1 Dataset Summary	13
2.3.2 Performance Tuning & Optimization	14
2.3.2.1 Metrics	14
2.3.3 Performance Tuning & Optimization	14
2.3.3.1 Model 1: CNN + LSTM	14
2.3.3.2 Model 2: Enhanced CNN + LSTM	15
2.3.3.3 Model 3: Optimized CNN + LSTM	15
2.3.3.4 Model 4: Transformer (Word2Vec)	16
2.3.4 Basic deployment of Speech Emotion Recognition Model	16
2.3.4.1 Tools and Frameworks Used	16
2.3.4.2 Summary of Application Development	16
2.3.4.3 Evaluation of Solution	17

<b>3. Application Development:</b>	<b>17</b>
3.1 Integration of Services/Models	17
<b>4. Teamwork:</b>	<b>18</b>
<b>Appendices</b>	<b>19</b>
Appendix A - Sha	19
1. Model Performance graphs & Classification reports	19
Model 1: CNN + LSTM	19
Model 2: Enhanced CNN + LSTM	19
Model 3: Optimized CNN + LSTM	20
2. Demo for SER Application	21

# 1. Motivation:

The airline industry, like many other customer-centric industries, relies heavily on customer feedback and interactions to maintain high levels of service quality. However, despite the growing adoption of customer service technologies, it remains challenging for frontline customer service operators to fully understand the emotional state of customers in real-time. Often, operators rely on their intuition or limited verbal cues to gauge emotions. However, this is prone to bias and may lead to missed opportunities for providing optimal service. Inaccurate recognition of a customer's emotional state can lead to poor customer experiences, potentially increasing frustration or dissatisfaction.

A significant portion of customer service interactions is based on phone calls or voice-based feedback. In such cases, the emotional tone of the customer's voice provides critical contextual information that can help a service representative adjust their response or level of empathy accordingly. Additionally, textual feedback such as online reviews, emails, and chat messages provides another layer of emotion that can be analyzed to improve service quality.

Currently, there are two major challenges within this context:

1. **Speech Emotion Recognition (SER):** Detecting emotions from tone, pitch, and pace of speech to infer emotion.
2. **Text Sentiment Analysis:** Extracting emotions from the words used by the customer and analyzing the sentiment expressed in the text. While this is a valuable approach, it may miss subtleties such as sarcasm or the intensity of emotions conveyed through speech.

This project aims to tackle the problem of improving emotional understanding by combining both speech and text data through deep learning techniques. By doing so, we can provide a more robust solution to detect emotions accurately and in real-time, enabling customer service operators in the airline industry to tailor their responses more effectively, reduce the risk of misunderstandings, and ultimately enhance the customer experience.

## **The Setting**

The project focuses on the airline industry, where customer service representatives handle a wide variety of emotionally charged interactions. These could involve distressed passengers due to delayed or canceled flights, passengers seeking assistance with special requests, or customers who are simply unhappy with their travel experience. Understanding the emotional state of the customer in such interactions is essential for the airline to improve its service, resolve issues quickly, and maintain customer satisfaction. By implementing a multimodal approach using both speech and text data, we aim to develop an advanced emotion detection system capable of offering more accurate predictions of the customer's emotional state, and thus better inform the decision-making process in customer interactions. This is especially important in the airline industry, where customer service is a key differentiator in brand loyalty.

## 2. Model Development:

### 2.1 Arun

#### 2.1.1 Dataset

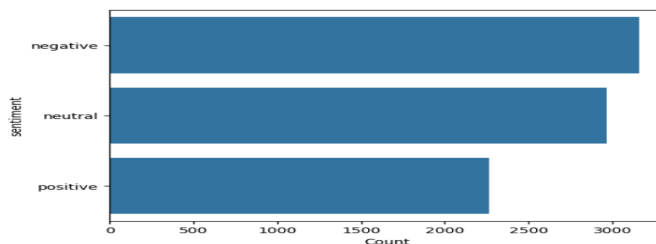
##### 2.1.1.1 Dataset Collection

Link: <https://www.kaggle.com/datasets/crowdfunder/twitter-airline-sentiment/data>

**Task:** Sentiment classification and Emotion Detection and **Sentiment Labels:** {0: "neutral", 1: "positive", 2: "negative"} The dataset is initially downloaded from Kaggle in **SQLite DB format** and subsequently loaded into a DataFrame from the 'tweets' table.

##### 2.1.1.2 Dataset Description

The sentiment distribution in the dataset is as follows: **Negative:** 9019, **Neutral:** 2968, and **Positive:** 2263, which is **imbalanced**. To balance the dataset, some of the Negative sentiments were removed from the dataframe. The final sentiment distribution is **Negative:** 3157, **Neutral:** 2968, and **Positive:** 2263



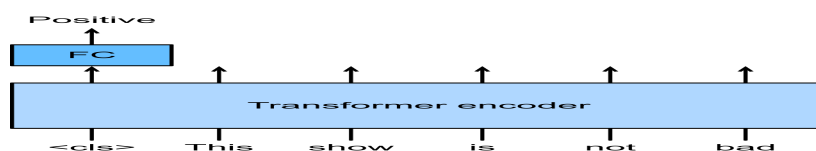
##### 2.1.1.3 Data Pre-Processing

- 1.Target Label Mapping: {"negative": 0, "positive": 1, "neutral": 2} using Label Encoder
- 2.Data cleaning: Removing special characters, digits, unnecessary symbols, and stop words
- 3.Tokenization: Converting the text to numerical vectors using relevant tokenizer. Ex: BERTTokenizer for BERT model

##### 2.1.2 Model Architecture



**Fine Tuned Model:**



## 2.1.3 Models Trained

### 2.1.3.1 LSTM model training with the Airline customer tweet dataset

1. Data PreProcessing: removing special characters, digits, unnecessary symbols, and stop words.
2. Encoding the target variable using Label Encoder
3. Tokenizing and converting the text into numerical vectors
4. The model is constructed with an embedding layer to transform words into dense vectors, followed by a dropout layer for regularization. It includes an LSTM layer with dropout for sequence learning and concludes with a dense layer using softmax activation for multi-class classification. The model is optimized with the Adam optimizer and uses categorical cross-entropy loss.
5. Performance evaluation of model using classification report

### 2.1.3.2. Fine tuning the BERT-base model with the Airline customer tweet dataset

1. Data PreProcessing: Tokenization is performed with the BERT Tokenizer, adding special tokens to separate sentences for classification. Sequences are padded to a constant length, and an attention mask is created with 0s for pad tokens and 1s for real tokens.
2. The pre-trained BERT model is loaded, with a dropout layer added at 30% probability to prevent overfitting. The forward method passes the input through BERT, dropout, and a linear layer to generate final class scores.
3. The model is fine-tuned by training on airline customer tweet text, setting the optimizer, batch size, learning rate, and number of epochs parameters. These parameters are configured for optimal performance during training.
4. Performance evaluation of model
5. Save the high accuracy fine tuned model for Sentiment classification

## 2.1.4 Performance Tuning & Optimization

### 2.1.4.1 LSTM Model Hyperparameter Tuning

Runs	Description	Classification Report				
Initial Training	embedding layer (500 input dimensions, 120 output dimensions), spatial dropout (0.4), an LSTM layer (176 units, 0.2 dropout and recurrent dropout), and a softmax output layer with 3 classes, optimized using Adam and categorical cross-entropy loss for multi-class classification.		precision	recall	f1-score	support
		0	0.75	0.70	0.72	986
		1	0.64	0.71	0.67	852
		2	0.75	0.70	0.73	679
		accuracy			0.71	2517
		macro avg	0.71	0.71	0.71	2517
		weighted avg	0.71	0.71	0.71	2517

Learning rate applied	The learning rate is decayed by 0.9 after 5 epochs, and the model is trained for 10 epochs with a batch size of 32 using this scheduler	<table><tr><td></td><td>precision</td><td>recall</td><td>f1-score</td><td>support</td></tr><tr><td>0</td><td>0.71</td><td>0.71</td><td>0.71</td><td>986</td></tr><tr><td>1</td><td>0.63</td><td>0.67</td><td>0.65</td><td>852</td></tr><tr><td>2</td><td>0.75</td><td>0.70</td><td>0.72</td><td>679</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.69</td><td>2517</td></tr><tr><td>macro avg</td><td>0.70</td><td>0.69</td><td>0.69</td><td>2517</td></tr><tr><td>weighted avg</td><td>0.69</td><td>0.69</td><td>0.69</td><td>2517</td></tr></table>		precision	recall	f1-score	support	0	0.71	0.71	0.71	986	1	0.63	0.67	0.65	852	2	0.75	0.70	0.72	679	accuracy			0.69	2517	macro avg	0.70	0.69	0.69	2517	weighted avg	0.69	0.69	0.69	2517
	precision	recall	f1-score	support																																	
0	0.71	0.71	0.71	986																																	
1	0.63	0.67	0.65	852																																	
2	0.75	0.70	0.72	679																																	
accuracy			0.69	2517																																	
macro avg	0.70	0.69	0.69	2517																																	
weighted avg	0.69	0.69	0.69	2517																																	
Bidirectional layer added	Bidirectional layer for the better context understanding	<table><tr><td></td><td>precision</td><td>recall</td><td>f1-score</td><td>support</td></tr><tr><td>0</td><td>0.74</td><td>0.72</td><td>0.73</td><td>986</td></tr><tr><td>1</td><td>0.65</td><td>0.67</td><td>0.66</td><td>852</td></tr><tr><td>2</td><td>0.72</td><td>0.73</td><td>0.73</td><td>679</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.71</td><td>2517</td></tr><tr><td>macro avg</td><td>0.71</td><td>0.71</td><td>0.71</td><td>2517</td></tr><tr><td>weighted avg</td><td>0.71</td><td>0.71</td><td>0.71</td><td>2517</td></tr></table>		precision	recall	f1-score	support	0	0.74	0.72	0.73	986	1	0.65	0.67	0.66	852	2	0.72	0.73	0.73	679	accuracy			0.71	2517	macro avg	0.71	0.71	0.71	2517	weighted avg	0.71	0.71	0.71	2517
	precision	recall	f1-score	support																																	
0	0.74	0.72	0.73	986																																	
1	0.65	0.67	0.66	852																																	
2	0.72	0.73	0.73	679																																	
accuracy			0.71	2517																																	
macro avg	0.71	0.71	0.71	2517																																	
weighted avg	0.71	0.71	0.71	2517																																	

**The LSTM model achieved a maximum accuracy of 71% even after performance tuning.**

#### 2.1.4.2 BERT Model Hyperparameter Tuning

The BERT model reached a peak accuracy of 81% after hyperparameter tuning, which contributed to better model generalization. Throughout the fine-tuning process, from the initial stages to the final one, the validation loss improved, and overfitting was minimized. The final high-accuracy model was exported and deployed on Hugging Face for application development

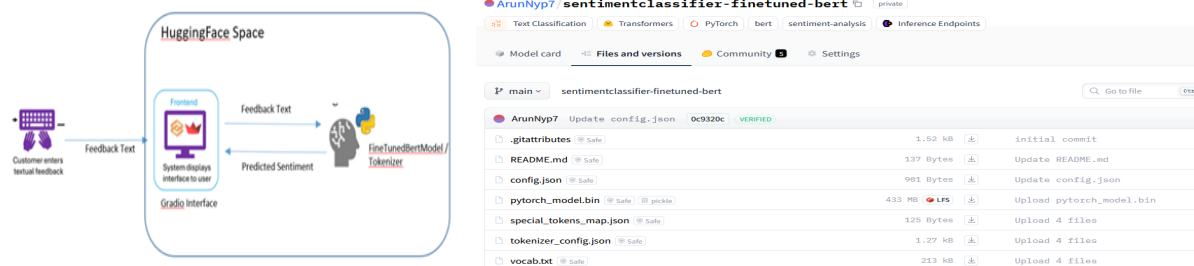
Runs	Description	Accuracy/Loss	Val Accuracy
Initial Fine Tuning	parameters: MAX_LEN=100, BATCH_SIZE=16, EPOCHS=10, and LEARNING_RATE=2e-5, AdamW optimizer, CrossEntropyLoss, and a learning rate scheduler		80%
Weight decay applied	Weight decay of 0.01 added in the Adam optimizer		81.3%
Increased Batch Size, Learning Rate	parameters: MAX_LEN=100, BATCH_SIZE=32, EPOCHS=10, and LEARNING_RATE=3e-5, AdamW optimizer, CrossEntropyLoss, and a learning rate scheduler		81.38%

## 2.1.5 Application Development

### 2.1.5.1 FineTuned Bert Sentiment Classification Model

<https://huggingface.co/spaces/ArunNyp7/ArunNyp7-sentimentclassifier-finetuned-bert>

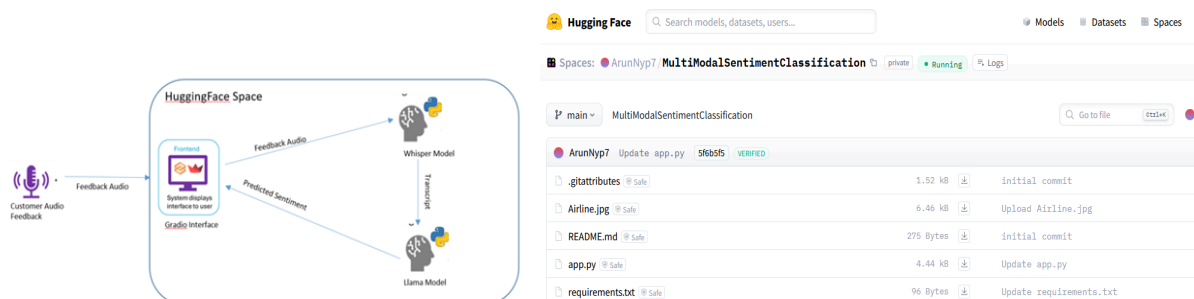
- Deployed FineTuned Bert Model exported as pytorch.bin along with the Tokenizer in the HuggingFace repo
- Developed Gradio interface to get the text input and display predicted sentiment



### 2.1.5.2 Multimodal Sentiment Classification using Whisper and Llama

<https://huggingface.co/spaces/ArunNyp7/MultiModalSentimentClassification>

- Created a API token in the GROQ to access the Whisper and Llama model
- Developed Gradio interface to get the Audio Input and convert it to Text transcript using Whisper model
- Developed the Gradio interface to get the audio input and convert to text using pre-trained Whisper model and display the predicted sentiment and emotions from the pre-trained Llama LLM model



Please refer to the appendix for the Gradio interface APP.



## 2.2 Kalai

### 2.2.1 Dataset Summary

#### 2.2.1.1 Dataset:

Link: <https://www.kaggle.com/datasets/crowdfunder/twitter-airline-sentiment/data>

**Task:** Sentiment classification

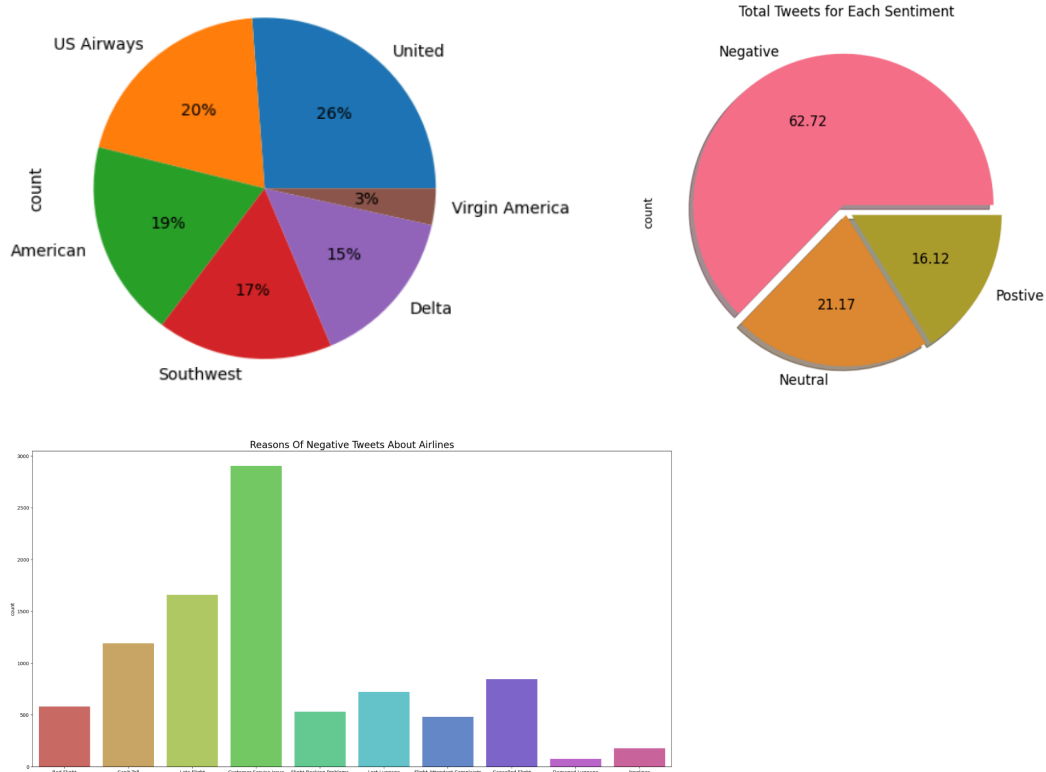
**Labels:**

- **Negative (0):** Complaints, bad experiences, delays, poor service
- **Neutral (1):** General statements, facts, or mixed opinions
- **Positive (2):** Praise, good experiences, appreciation

#### 2.2.1.2 Dataset Applications:

- **Sentiment Analysis:** Classifying tweets into negative, neutral, and positive categories.
- **Customer Feedback Analysis:** Understanding common complaints about airlines.
- **Fine-tuning NLP Models:** Great dataset for training transformer models like LSTM/BERT.

#### 2.2.1.3 Airline-wise and Sentiment Count:



#### 2.2.1.4 Feature Engineering:

- Combine negative reason with tweet
- Remove @username / url / emojis / stop words
- Remove Punctuation & Special Characters
- Separate alphanumeric

#### 2.2.2 Performance Tuning & Optimization

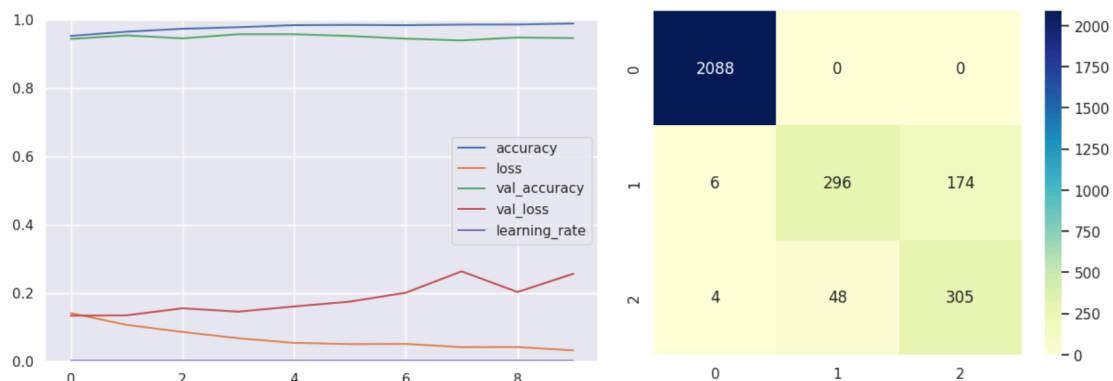
##### 2.2.2.1 LSTM: (Before & Learning\_rate=0.2)

Accuracy: 0.6436					Accuracy: 0.6436				
	precision	recall	f1-score	support		precision	recall	f1-score	support
negative	0.64	1.00	0.78	1880	negative	0.64	1.00	0.78	1880
neutral	0.00	0.00	0.00	582	neutral	0.00	0.00	0.00	582
positive	0.00	0.00	0.00	459	positive	0.00	0.00	0.00	459
accuracy			0.64	2921	accuracy			0.64	2921
macro avg	0.21	0.33	0.26	2921	macro avg	0.21	0.33	0.26	2921
weighted avg	0.41	0.64	0.50	2921	weighted avg	0.41	0.64	0.50	2921

##### 2.2.2.2 Transformers:

Accuracy: 0.9459					Accuracy: 0.9206				
	precision	recall	f1-score	support		precision	recall	f1-score	support
negative	1.00	1.00	1.00	2088	negative	1.00	1.00	1.00	2088
neutral	0.88	0.77	0.82	476	neutral	0.86	0.62	0.72	476
positive	0.75	0.86	0.80	357	positive	0.64	0.85	0.73	357
accuracy			0.95	2921	accuracy			0.92	2921
macro avg	0.88	0.88	0.87	2921	macro avg	0.83	0.83	0.82	2921
weighted avg	0.95	0.95	0.95	2921	weighted avg	0.93	0.92	0.92	2921

#### Training and Validation Accuracy/Loss and Confusion Matrix:



##### 2.2.2.3 BERT:

Fine-tuning **BERT** requires choosing the best **learning rate**, **batch size**, and **epochs**.

- Trained with `epochs=3` & `learning_rate=2e-5`

Epoch	Training Loss	Validation Loss
1	0.135500	0.141151
2	0.145000	0.185379
3	0.072000	0.209042

```
TrainOutput(global_step=2193, training_loss=0.10974100785227166, metrics={'train_runtime': 907.5178, 'train_samples_per_second': 38.621, 'train_steps_per_second': 2.416, 'total_flos': 2305465544487168.0, 'train_loss': 0.10974100785227166, 'epoch': 3.0})
```

```
[183/183 00:22]
{'eval_loss': 0.20904161036014557,
 'eval_runtime': 22.5454,
 'eval_samples_per_second': 129.561,
 'eval_steps_per_second': 8.117,
 'epoch': 3.0}
```

- Use Grid Search for Hyperparameters

Epoch	Training Loss	Validation Loss	Accuracy
1	0.010000	0.438335	0.941116
2	0.013400	0.452077	0.946936
3	0.011100	0.453779	0.945909

```
lr: 2e-05, batch: 16, epochs: 3, eval_accuracy: 0.9459089352961315
```

[2193/2193 06:20, Epoch 3/3]			
Epoch	Training Loss	Validation Loss	Accuracy
1	0.015300	0.433673	0.944197
2	0.024800	0.373541	0.951044
3	0.016600	0.347087	0.948305

```
lr: 3e-05, batch: 16, epochs: 3, eval_accuracy: 0.9483053748716193
```

Epoch	Training Loss	Validation Loss	Accuracy
1	0.036000	0.383691	0.935981
2	0.027300	0.372247	0.943512
3	0.018500	0.388933	0.944882

```
1r: 5e-05, batch: 16, epochs: 3, eval_accuracy: 0.9448818897637795
Best Hyperparameters: {'lr': 3e-05, 'batch': 16, 'epochs': 3} with Accuracy: 0.9483053748716193
```

- Using FP16 (Mixed Precision) reduces memory usage & speeds up training. It will reduce **GPU** memory usage by 50%, allowing larger batch sizes.

Epoch	Training Loss	Validation Loss
1	0.018200	0.375958
2	0.026800	0.388826
3	0.011000	0.371484

```
TrainOutput(global_step=2193, training_loss=0.01766131036204396, metrics=
{'train_runtime': 443.097, 'train_samples_per_second': 79.1, 'train_steps_per_second':
4.949, 'total_flos': 2305465544487168.0, 'train_loss': 0.01766131036204396, 'epoch':
3.0})
```

```
[183/183 00:06]
{'eval_loss': 0.3714837431907654,
 'eval_runtime': 6.7431,
 'eval_samples_per_second': 433.186,
 'eval_steps_per_second': 27.139,
 'epoch': 3.0}
```

### 2.2.3 Application Development

- Deployed in hugging face with `epochs=3` & `learning rate=2e-5`

model.safetensors: 100%  438M/438M [00:15<00:00, 37.2MB/s]

README.md: 100%  5.17k/5.17k [00:00<00:00, 139kB/s]

```
CommitInfo(commit_url='https://huggingface.co/kalaivaniramachandran39/bert-airline-sentiment/commit/2cf667ea608c45b5815f22c010b4e2344f80d', commit_message='Upload tokenizer', commit_description='', oid='2cf667ea608c45b5815f22c010b4e2344f80d', pr_url=None, repo_url=RepoUrl('https://huggingface.co/kalaivaniramachandran39/bert-airline-sentiment'), endpoints=('https://huggingface.co',), repo_type='model', repo_id='kalaivaniramachandran39/bert-airline-sentiment'), pr_revision=None, pr_num=None)
```

```
config.json: 100% ██████████ 881/881 [00:00<00:00, 29.0kB/s]
```

```
model.safetensors: 100%  438M/438M [00:10<00:00, 42.4MB/s]
```

tokenizer\_config.json: 100%  1.27k/1.27k [00:00<00:00, 50.0kB/s]

vocab.txt: 100%  232k/232k [00:00<00:00, 3.67MB/s]

```
special_tokens_map.json: 100% ██████████ 125/125 [00:00<00:00, 3.02kB/s]
```

- Tested from CoLab using the deployed Huggingface model

```

1 from transformers import pipeline
2
3 model_name = "kalaivaniramachandran39/bert-airline-sentiment"
4
5 # Use the uploaded model from Hugging Face
6 sentiment_pipeline = pipeline("text-classification", model=model_name)
7
8 # Predict sentiment of a tweet
9 text = "The flight was amazing! Great service. 🌟"
10 print(sentiment_pipeline(text))
11 # {'negative': 0, 'neutral':1,'positive':2}

```

Device set to use cuda:0  
[{'label': 'LABEL\_2', 'score': 0.9965595602989197}]

```

1 # Predict sentiment of a tweet
2 text = "bad servie 🌟"
3 print(sentiment_pipeline(text))
4 # {'negative': 0, 'neutral':1,'positive':2}

```






[{'label': 'LABEL\_1', 'score': 0.9775744080543518}]

## - Deployed BERT model with Grid Search for Hyperparameters & Test

```

1 from transformers import BertForSequenceClassification, BertTokenizer
2 import torch
3
4 # Load model & tokenizer
5 model_name = "kalaivaniramachandran39/bert-airline-sentiment_gs" # Update this with your model's repo
6 tokenizer = BertTokenizer.from_pretrained(model_name)
7 model = BertForSequenceClassification.from_pretrained(model_name)

```

tokenizer\_config.json: 100%  1.27k/1.27k [00:00<00:00, 30.8kB/s]  
vocab.txt: 100%  232k/232k [00:00<00:00, 1.89MB/s]  
special\_tokens\_map.json: 100%  695/695 [00:00<00:00, 15.3kB/s]  
config.json: 100%  913/913 [00:00<00:00, 22.0kB/s]  
model.safetensors: 100%  438M/438M [00:10<00:00, 42.0MB/s]

```

1 print(predict_sentiment("The flight was delayed for hours. Terrible experience!")) # Expected: negative
2 print(predict_sentiment("The service was okay, nothing special.")) # Expected: neutral
3 print(predict_sentiment("I had a fantastic time flying with this airline!")) # Expected: positive

```






positive  
neutral  
positive

## - Deployed BERT model with fp16 (Mixed precision) & Test

```

1 from transformers import BertForSequenceClassification, BertTokenizer
2 import torch
3
4 # Load model & tokenizer
5 model_name = "kalaivaniramachandran39/bert-airline-sentiment_fp" # Update this with your model's repo
6 tokenizer = BertTokenizer.from_pretrained(model_name)
7 model = BertForSequenceClassification.from_pretrained(model_name)
8

```

tokenizer\_config.json: 100%  1.27k/1.27k [00:00<00:00, 61.7kB/s]  
vocab.txt: 100%  232k/232k [00:00<00:00, 4.49MB/s]  
special\_tokens\_map.json: 100%  695/695 [00:00<00:00, 18.3kB/s]  
config.json: 100%  910/910 [00:00<00:00, 22.7kB/s]  
model.safetensors: 100%  438M/438M [00:10<00:00, 42.6MB/s]

Model with Sample Texts

```

1 print(predict_sentiment("The flight was delayed for hours. Terrible experience!")) # Expected: negative
2 print(predict_sentiment("The service was okay, nothing special.")) # Expected: neutral
3 print(predict_sentiment("I had a fantastic time flying with this airline!")) # Expected: positive

```

positive  
neutral  
positive

## 2.3 Sha

### 2.3.1 Dataset Summary

The Ryerson Audio-Visual Database of Emotional Speech and Song ([RAVDESS](#))" by Livingstone & Russo is licensed under CC BY-NA-SC 4.0 dataset, consists of 1440 high-quality audio files recorded by 24 professional actors (12 male, 12 female), captures 8 distinct emotions: calm, happy, sad, angry, fearful, surprise, disgust, and neutral.

With its diverse range of emotional speech and clear, high-resolution recordings, RAVDESS provides an ideal foundation for building a deep learning model focused on Speech Emotion Recognition (SER). By analyzing speech features such as tone, pitch, and pace, this dataset enables accurate emotion detection, helping to address challenges in recognizing and understanding emotions from speech, ultimately enhancing the emotional awareness of systems in real-time applications like customer service.

### 2.3.2 Performance Tuning & Optimization

#### 2.3.2.1 Metrics

For all models, the following will be provided:

- Accuracy and Loss Graphs: These will show the evolution of both training and validation accuracy and loss over the epochs.
- Classification Report: This report will detail accuracy, precision, recall, and F1 score for both the training and validation datasets to better assess the models' performance in different metrics.

### 2.3.3 Performance Tuning & Optimization

Please see appendix A for each model's Accuracy and Loss performance graphs, and Classification Reports for train and validation datasets.

#### 2.3.3.1 Model 1: CNN + LSTM

##### Architecture & Features:

- The initial model combines Convolutional Neural Networks (CNN) for feature extraction from the raw MFCCs (Mel-frequency cepstral coefficients) and Long Short-Term Memory (LSTM) layers for processing sequential speech data.
- The CNN layers help extract important temporal features, while the LSTM layers aim to capture long-term dependencies in speech patterns.
- Preprocessing includes trimming silence, balancing the dataset, and fixing the sample rate to ensure consistent input for the model.

#### Performance:

- Training Accuracy: 33.42%
- Validation Accuracy: 33.33%
- Loss: ~1.0986 (constant, suggesting random guesses)

#### Possible causes include:

- Softmax Outputs Stuck: The model may not be learning effectively.
- Architecture/Vanishing Gradients: The LSTM model may suffer from vanishing gradients, making it difficult for the model to learn from sequential data.

Summary: Model 1 provides a baseline but struggles with learning and generalization due to potential issues in architecture. The constant loss and accuracy suggest the need for further optimization.

### 2.3.3.2 Model 2: Enhanced CNN + LSTM

#### Modifications & Fine-tune Hyperparameters (builds on Model 1):

- Increased the number of LSTM layers and units to improve the extraction of temporal features from speech.
- Introduced Batch Normalization to stabilize training and improve gradient flow.
- Added Dropout (30%) to reduce overfitting, and used ReLU activation before the softmax layer to enhance learning in deeper networks.

#### Performance:

- Training Accuracy: 97.18%
- Validation Accuracy: 58.84%
- Loss: Improved compared to Model 1, but still indicates a gap between training and validation performance.

#### Issues Identified:

- There is a clear overfitting issue, as evidenced by the significant gap between training and validation accuracy.
- While training accuracy is high, the model fails to generalize well to the validation data.

Summary: Model 2 shows improved training performance but struggles with generalization. It demonstrates progress with the introduction of Batch Normalization and increased LSTM capacity, but further adjustments to regularization are required to reduce overfitting.

### 2.3.3.3 Model 3: Optimized CNN + LSTM

#### Modifications & Fine-tune Hyperparameters (builds upon Model 2):

- L2 Regularization is added to the LSTM layers to penalize large weights and reduce overfitting.
- A Learning Rate Scheduler is introduced, which reduces the learning rate by a factor of 0.5 if the validation loss doesn't improve for 5 epochs, allowing the model to adapt its learning pace.

- The Dropout rate is increased slightly (from 0.3 to 0.4 or 0.5) to further help with generalization.

#### Performance:

- Training Accuracy: 99.67%
- Validation Accuracy: 65.51%
- Loss: Further reduced compared to Models 1 and 2. Has better generalization with an improved training and validation accuracy.

Summary: Model 3 is the most optimized version, demonstrating significantly improved generalization and training stability. It combines regularization, normalization, and learning rate scheduling, leading to better performance than the previous models, making it the chosen baseline for the project.

#### 2.3.3.4 Model 4: Transformer (Word2Vec)

Summary: Model 4 follows the same preprocessing steps as the previous models, utilizing CNN and LSTM layers for feature extraction and temporal modeling. In addition, it integrates a transformer model, specifically the “facebook/wav2vec2-base” from Hugging Face, for speech-to-text processing. This transformer model is designed for automatic speech recognition (ASR) and was expected to provide rich features for emotion detection in speech.

#### Challenges Encountered:

- **Input Size Issue:** The model encountered difficulties related to the input size during training, which prevented the model from being trained successfully.

Given the time constraints, debugging and resolving these issues were not completed, but this model offers promising potential for future improvements once the input size problem is addressed.

### 2.3.4 Basic deployment of Speech Emotion Recognition Model

Please see appendix A for demonstration example screenshots.

#### 2.3.4.1 Tools and Frameworks Used

The tools and frameworks used to develop the application include:

- **Google Colab:** Used for training, debugging, and deploying models. Google Colab provides a cloud-based environment with GPU support, which was essential for model training.
- **Hugging Face API:** Used for hosting and deploying Model 3 in a scalable manner. It allows easy integration into web applications and provides access to the model via an API endpoint.
- **Gradio:** Used to create a user-friendly interface for the model, allowing users to upload audio files and receive emotion predictions.

### 2.3.4.2 Summary of Application Development

1. **Model Deployment:** The baseline model (Model 3) was saved and deployed to the Hugging Face Model Hub using the Hugging Face API from Google Colab. This allowed the model to be accessed and used for inference via a cloud-based API.
2. **User Interface with Gradio:** A user-friendly interface was created using Gradio, allowing users to upload audio files for emotion prediction. Model 3 processes the audio and returns the predicted emotion label (neutral, positive, or negative) in real-time.
3. **Model Inference & Evaluation:** The deployed model was able to correctly predict the emotion for approximately 2 out of 3 test samples, aligning with Model 3's accuracy of 65%. Although the current accuracy is moderate, it demonstrates a functional solution that can be improved with future adjustments, such as debugging issues with Model 4.

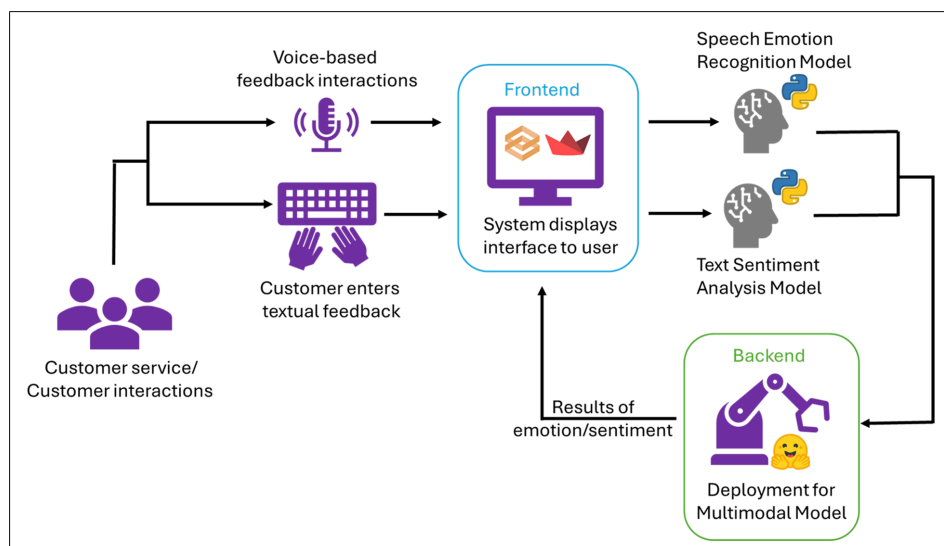
### 2.3.4.3 Evaluation of Solution

The deployed application provides a functional solution to the Speech Emotion Recognition task, with Model 3 performing at an accuracy (validation) of 65.51%. While this accuracy is not yet perfect, it is sufficient to demonstrate the feasibility of the approach and provide a usable tool for real-world applications, such as improving customer service interactions.

Conclusion: The application provides a functional, deployable solution for Speech Emotion Recognition using deep learning. The integration of Model 3 into the Gradio interface via Hugging Face was successful, offering users the ability to upload audio and receive emotion predictions. While the current accuracy of 67% indicates room for improvement, the project is a strong foundation that can be further enhanced as debugging issues with Model 4 are resolved.

## 3. Application Development:

### 3.1 Integration of Services/Models





The application integrates multiple services and models to classify customer sentiment from both text and speech feedback. Here's how:

**1. Text Feedback:**

- Customers submit written feedback, which is preprocessed and passed through a transformer-based model (e.g., BERT) for sentiment classification.
- The sentiment label (e.g., neutral, positive, negative) is displayed to the user via the Gradio/Streamlit frontend.

**2. Speech Feedback:**

- Features like pitch and tone are extracted from the audio, and a deep learning model (e.g., LSTM or Transformer) classifies the emotional tone.
- The result is displayed to the user on the frontend.
- Audio feedback is transcribed to text using a speech-to-text system.

**3. Frontend with Gradio/Streamlit:**

- Gradio/Streamlit provides an intuitive interface for customers to input text or provide audio feedback. It sends the data to the backend for processing and displays the sentiment results.

**4. Backend with Python and Hugging Face:**

- The models are developed using Python, with Hugging Face deployed for API management. The backend processes the feedback, communicates with the deployed models, and returns sentiment classifications to the frontend.

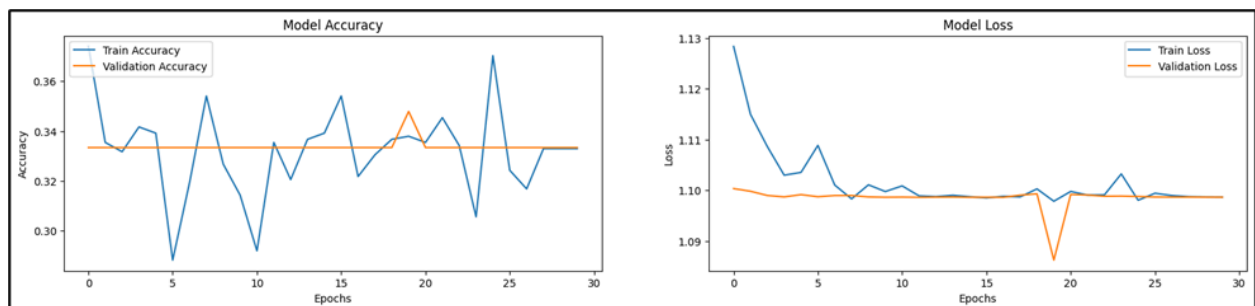
In essence, the text and speech feedback platforms work together through a combination of Python-based model development, Hugging Face deployment, and Gradio/Streamlit's user-friendly interface, allowing the seamless detection and display of customer sentiment.

## Appendices

### Appendix A - Sha

#### 1. Model Performance graphs & Classification reports

##### Model 1: CNN + LSTM



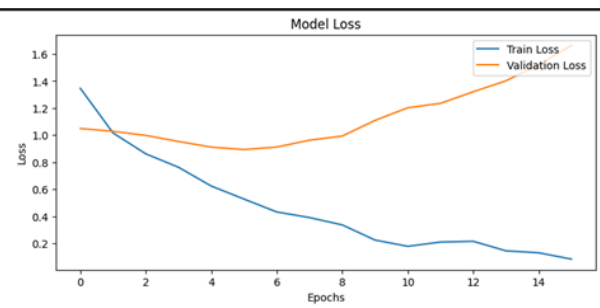
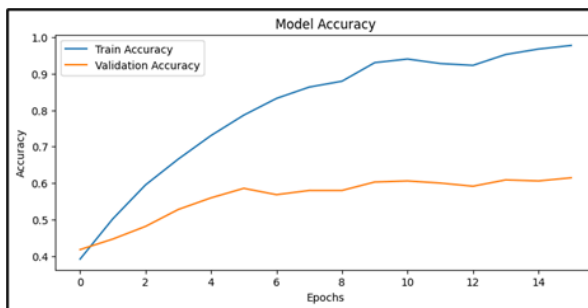
#### Train Data Classification Report:

	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

## Model 2: Enhanced CNN + LSTM



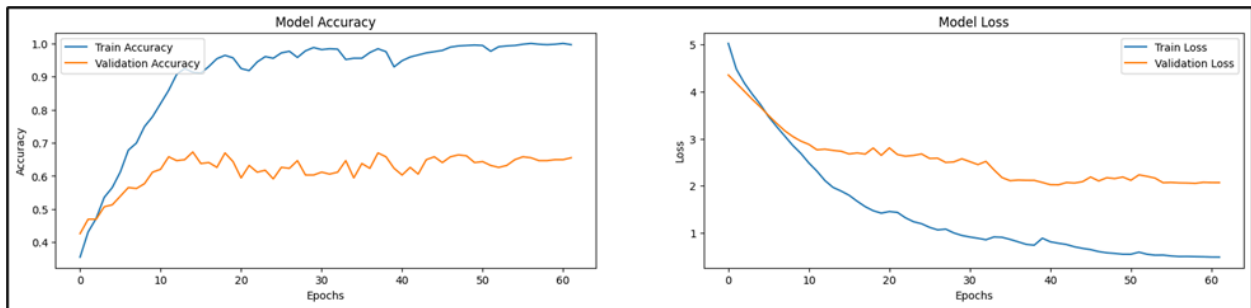
#### Train Data Classification Report:

	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

## Model 3: Optimized CNN + LSTM



### Train Data Classification Report:

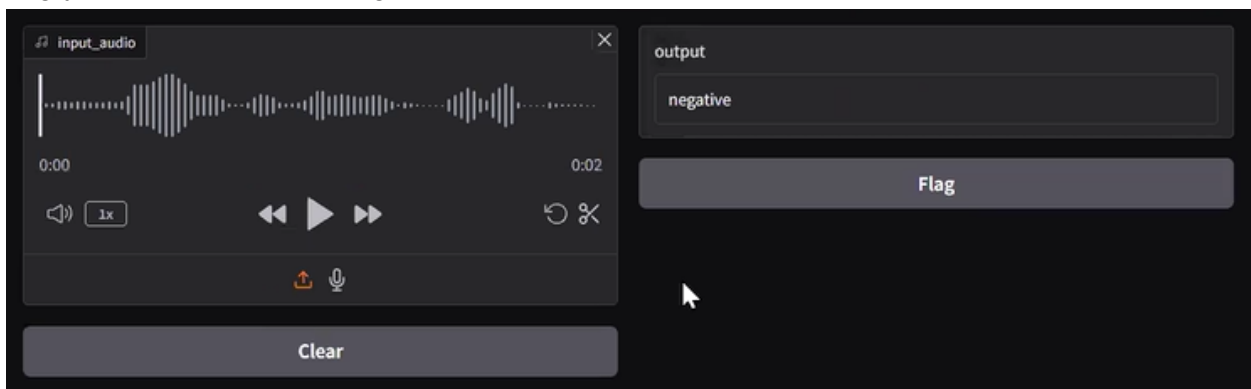
	precision	recall	f1-score	support
Neutral	1.00	1.00	1.00	268
Positive	1.00	1.00	1.00	268
Negative	1.00	1.00	1.00	269
accuracy			1.00	805
macro avg	1.00	1.00	1.00	805
weighted avg	1.00	1.00	1.00	805

### Validation Data Classification Report:

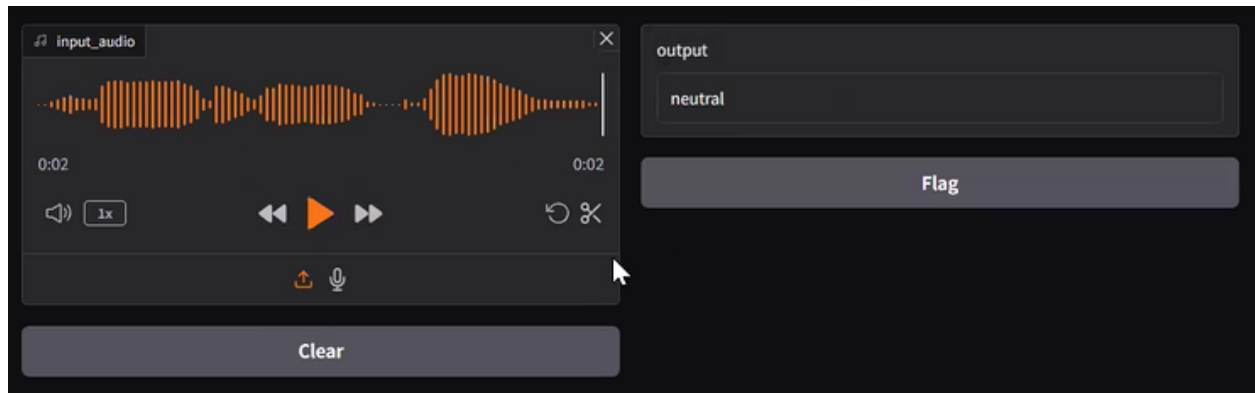
	precision	recall	f1-score	support
Neutral	0.50	0.53	0.51	115
Positive	0.70	0.73	0.71	115
Negative	0.66	0.58	0.62	115
accuracy			0.61	345
macro avg	0.62	0.61	0.62	345
weighted avg	0.62	0.61	0.62	345

## 2. Demo for SER Application

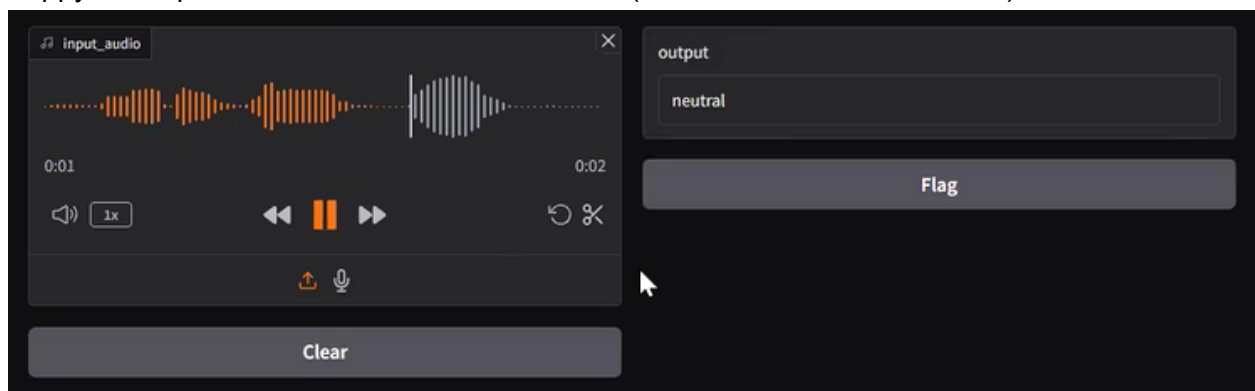
Angry audio predicted as Negative emotion label (correct).



Neutral audio predicted as Neutral emotion label (correct).



Happy audio predicted as Neutral emotion label (incorrect, should be Positive).




## Appendix B -Arun Kumar


1. Gradio interface for the Multimodel Sentiment classifier using pre-trained models(Whisper and Llama)


## Airline Customer Service Portal

Upload Audio

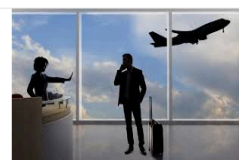


0:030:03





Image



Response

```
1  {
2    "ActualDialog": "But in this one section we welcomed auditors.",
3    "Sentiment": "Positive",
4    "Emotion": "Joy",
5    "ContributedText": "welcomed auditors"
6  }
```

Sentiment

Positive

Emotion

Joy

Emotion with Emoji

😊

## 2. Gradio interface for the Fine-tuned Bert model deployed in the Huggingface

Spaces

ArunNyp7

AzunNyp7-sentimentclassifier-finetuned-bert

👍 like 0

🟢 Running

📄 Logs

App

Files

Community

Settings

🔍

🌐

Sentiment Classification using Fine-Tuned BERT Model

Enter Text

Good airline

Label

Positive

Classify Sentiment

## 3. Sentiment prediction evaluation of the deployed fine tuned bert model using Huggingface LLM inference client

ble of contents

Jpload Fine Tuned BERT Model to  
uggingface

HuggingFace Client Login

Push model to github repo

Test uploaded model using the  
Huggingface Inference Client

Section

+ Code + Text

## Test uploaded model using the Huggingface Inference Client

[ ] !pip install requests

```
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests) (2.3.0)
Requirement already satisfied: certifi<=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests) (2025.1.31)
```

```
[ ] from huggingface_hub import InferenceClient
import json
```

```
[ ] repo_id="Arunkp7/sentimentclassifier-finetuned-bert"
llm_client=InferenceClient(model=repo_id,timeout=120,token="hf_CyIvLFqavJXkiwDmmELLNOHvMiwRKOclf")
```

```
[ ] def call_llm(inference_client: InferenceClient, prompt: str):
    response= inference_client.post(
        json={
            "inputs":prompt,
        },
        task="text-classification"
    )
    return response
```

```
[ ] response=call_llm(llm_client,"This airline timing matches my travel period")
print(response)
```

```
b'[[{"label":"POSITIVE","score":0.8981769680976868},{"label":"NEGATIVE","score":0.10182306170463562}]']
```

```
response=call_llm(llm_client,"This airline timing does not matches my travel period")
print(response)
```

```
b'[[{"label":"NEGATIVE","score":0.9997693896293664},{"label":"POSITIVE","score":0.00023056913050822914}]']
```

```
[ ] response=call_llm(llm_client,"This airline timing does not matter me the most")
print(response)
```

```
b'[[{"label":"NEGATIVE","score":0.9994499087333679},{"label":"POSITIVE","score":0.0005500498227775097}]']
```

```
[ ] response=call_llm(llm_client,"This airline timing can improve their timings")
print(response)
```

```
b'[[{"label":"NEGATIVE","score":0.97899329662323},{"label":"POSITIVE","score":0.021006640046834946}]']
```

```
[ ] Start coding or generate with AI.
```