XIAMEN UNIVERSITY MALAYSIA

Course Code	:	AIT301
Course Name	:	Advanced Machine Learning
Lecturer	:	Zhang Yingqian
Academic Session	:	202409
Assessment Title	:	Project (Final Assessment)
Submission Due Date	:	13/12/2024

Prepared by :

Student ID	Student Name
AIT2209940	Kwah Wen Yao
AIT2209942	Leong Yao Cheng
AIT2209954	Tay Re Zen
AIT2209608	Liew Cheah Ming

Date Received :

Feedback from Lecturer:

Mark:

Own Work Declaration

I/We hereby understand my/our work would be checked for plagiarism or other misconduct, and the softcopy would be saved for future comparison(s).

I/We hereby confirm that all the references or sources of citations have been correctly listed or presented and I/we clearly understand the serious consequence caused by any intentional or unintentional misconduct.

This work is not made on any work of other students (past or present), and it has not been submitted to any other courses or institutions before.

Signature: Cheng Kwah Cheah Tay

Date: 13/12/2024

Table of Contents

Title

Speech Recognition System

Abstract

Speech recognition is an essential part of current artificial intelligence, enabling applications like virtual assistants, transcription services, and accessibility aids. This research expands on traditional voice recognition by introducing emotion recognition, a novel feature that improves system engagement and empathy. By leveraging multiple datasets (CREMA-D, TESS, SAVEE, RAVDESS), audio features were extracted and analyzed using advanced deep learning architectures. A hybrid Conv1D and GRU model was developed to identify seven emotions from speech data while maintaining high speech recognition accuracy. Thus, this study highlights the potential of integrating emotion recognition into traditional speech systems, enabling more human-like interactions.

Introduction

Speech recognition technology has altered how humans interact with machines, becoming an essential component of applications such as voice assistants, transcription services, and smart devices. However, standard speech recognition algorithms sometimes lack the ability to detect the speaker's emotional context, limiting their usefulness in domains such as mental health, education, and customer service.

This study tries to close the gap by incorporating emotion recognition into speech recognition systems. Emotion identification improves traditional systems by detecting the speaker's emotional state, allowing for more meaningful and adaptive human-computer interactions. This breakthrough uses cutting-edge datasets and deep learning algorithms to accurately classify speech content and emotional expressions.

Our method combines feature extraction with tools such as Librosa with a novel deep learning model that blends Convolutional Neural Networks (Conv1D) and Gated Recurrent Units (GRU). This architecture is intended to successfully capture both temporal and contextual aspects of speech. By adding emotion recognition, this study hopes to push the bounds of existing voice recognition algorithms, making them more intuitive, empathic, and human-like.

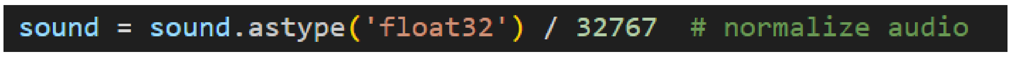
Methodology

Speech recognition is now a cornerstone of modern technology, enabling advances in natural language processing, human-computer interface, and assistive technologies. In this context, DeepSpeech, a recurrent neural network-based model for speech recognition from start to finish, has emerged as a transcription accuracy and efficiency standard. However, developments in machine learning provide opportunities to investigate fresh approaches that can improve existing models in terms of accuracy, computational efficiency, and robustness.

Dataset Preparation

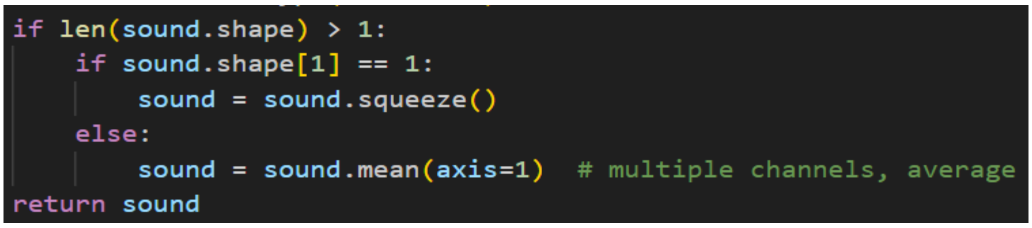
Audio Loading

Any fair comparison is built on the use of identical datasets for training and evaluation. For this study, we chose a dataset of different voice samples with varying accents, noise levels, and durations. Firstly, the raw audio signals are loaded from .wav files using soundfile library in our dataset.py file, as we are going to perform audio loading to prepare it for feature extraction. For the first step, we will do normalization by converting the audio data which is represented as 16-bit integers to float32 values in the range of [-1, 1] by dividing the maximum value of 32767 (for 16-bit).



```
sound = sound.astype('float32') / 32767  # normalize audio
```

The purpose of this is to ensure the amplitude values are standardized, avoiding issues with large input values during training. Moving on to channel handling, we will observe if the audio has multiple channels, it will be averaged to create a single channel or one channel is selected from the audio.

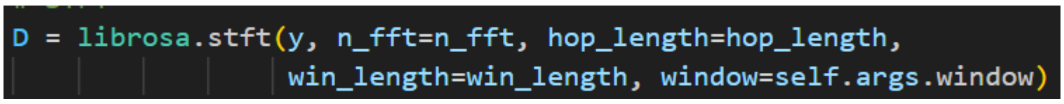


```
if len(sound.shape) > 1:
    if sound.shape[1] == 1:
        sound = sound.squeeze()
    else:
        sound = sound.mean(axis=1)  # multiple channels, average
return sound
```

This is because the model expects a single-channel audio to simplify processing. Lastly, the audio is resampled to a fixed sample rate which is 16000 Hz in our case, and this is to ensure uniform audio length and feature representation in the input data to reduce variability.

Feature Extraction

After audio loading, we will now move on to feature extraction which is the most important part in dataset preparation. In our project, we will not be directly using raw audio waveforms for training because of their high dimensionality. Instead, we will be extracting features like spectrograms to represent the audio in the time-frequency domain. First, we will implement Short-Time Fourier Transform (STFT), but what is STFT?



```
D = librosa.stft(y, n_fft=n_fft, hop_length=hop_length,
                 win_length=win_length, window=self.args.window)
```

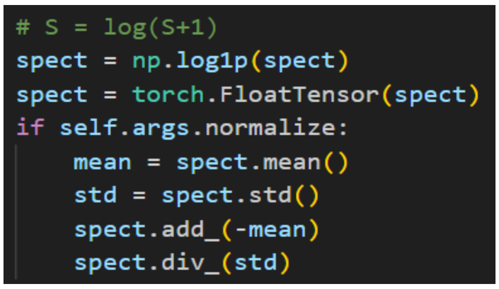
What is Short-Time Fourier Transform (STFT) ?

Basically, Short-Time Fourier Transform (STFT) is a mathematical technique used to analyze the frequency content of a signal over a period of time. This technique is very for signals which have frequency content that changes over time, for example, SPEECH!

How does it work?

The STFT is applied to break the audio signal into overlapping time frames and then compute the frequency of each frame. Firstly, it will apply a windowing function to each frame before computing the Fourier Transform to minimize edge artifacts in the frames. The formula for the function is $w[n] = 0.54 - 0.46cos((2\pi n) / (N-1))$. The parameters will be window_size which represents the size of the window (seconds) and window_stride which is the step size between consecutive windows. Next, the STFT will convert the signal into a complex-valued spectrogram, where the magnitude represents the signal's intensity at a given time.

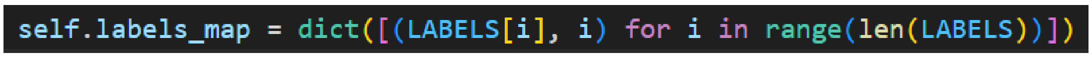
After that, the magnitude of the spectrogram is computed and transformed into a logarithmic scale with Log-Spectrogram = log(1 + |STFT Magnitude|) to compress the dynamic range of the signal, making smaller values more significant and suitable for training. Lastly, the spectrogram values are normalized to zero mean and unit variance.



```
# S = log(S+1)
spect = np.log1p(spect)
spect = torch.FloatTensor(spect)
if self.args.normalize:
    mean = spect.mean()
    std = spect.std()
    spect.add_(-mean)
    spect.div_(std)
```

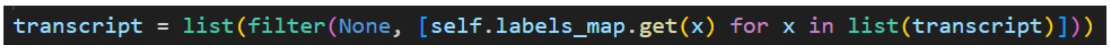
Label Encoding

In our datasets, transcriptions are provided as text files with the corresponding audio files and these texts are converted into numerical labels for model training. First step for label encoding, which is vocabulary creation, we have created a fixed set of characters, where each character is mapped to a unique integer.



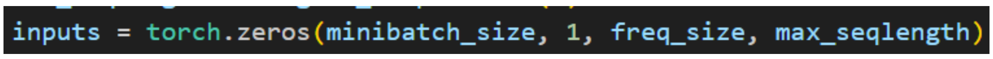
```
self.labels_map = dict([(LABELS[i], i) for i in range(len(LABELS))])
```

Second step, which is text processing, we will strip the text of whitespace and convert uppercase to match the vocabulary format. Each character is then replaced with its corresponding integer. The blank token(_) will then be used during training to represent alignment gaps between the input audio and the transcript.



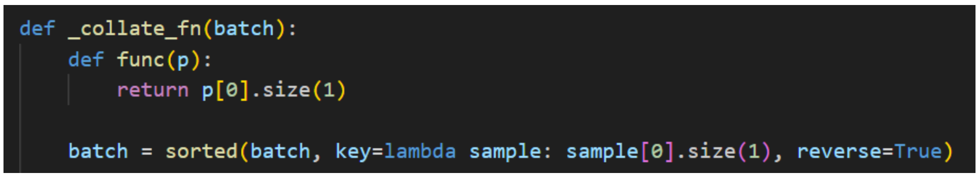
```
transcript = list(filter(None, [self.labels_map.get(x) for x in list(transcript)]))
```

After label encoding is done, we will be moving on to the next step, which is batching and padding the dataset. Our audio files and transcript vary in length but neural networks require inputs of uniform size, so we will pad audio spectrograms with zeros to the length of the longest sample in the batch and concatenate transcription labels and their sizes will be stored for alignment during loss computation.



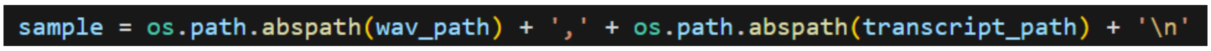
```
inputs = torch.zeros(minibatch_size, 1, freq_size, max_seqlength)
```

We will also be using the collation function to ensure uniformity as mentioned in batch processing by sorting samples by audio length in descending order, padding audio spectrograms to match the longest one in the batch and storing the original sequence lengths to use during training.



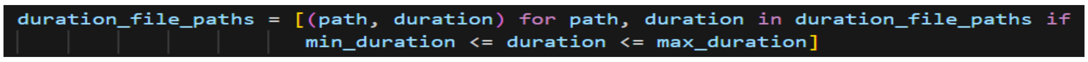
```
def _collate_fn(batch):
    def func(p):
        return p[0].size(1)

    batch = sorted(batch, key=lambda sample: sample[0].size(1), reverse=True)
```

Last part of the dataset preparation, which is manifest file creation. We will generate manifest files to store the paths of audio and corresponding transcription files. This will help during the dataset loading process. Each .wav file is paired with its corresponding .txt file, and the manifest contains the absolute paths of these pairs.



```
sample = os.path.abspath(wav_path) + ',' + os.path.abspath(transcript_path) + '\n'
```

We will also be excluding files outside a specific duration range (either too short or too long) to prevent issues during training.



```
duration_file_paths = [(path, duration) for path, duration in duration_file_paths if
                       min_duration <= duration <= max_duration]
```

Model Architecture

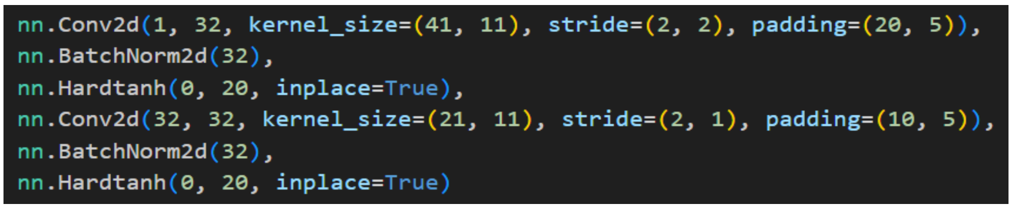
For our model, it is inspired by the DeepSpeech architecture. It combines convolutional layers for feature extraction, recurrent layers for capturing temporal dependencies, fully connected lakers for the final predictions for characters, and a softmax layer to convert logits into probabilities during inference.

Convolutional Layers

Let’s take a deeper look at our model, firstly at the convolutional layers. They are used to extract high-level, time-frequency features from the input spectrogram, and these layers reduce the input’s dimensionality while preserving essential features.

For the first convolutional layer: The kernel size is (41, 11) where time x frequency dimensions. Stride (2,2) for down-sampling, and Padding (20,5) to preserve spatial dimensions Second convolutional layer has a kernel size of (21, 11) and stride of (2,1), to preserve frequency resolution and padding of (10, 5). As for the trainable parameters, for layer 1 the parameters are (32 x 1 x 41 x 11) + 32 which is 14464 and for layer 2 there are (32 x 32 x 21 x 11) + 32 parameters which is 236576 parameters for a total of 251040 total parameters for convolutional layers.

Batch normalization is also applied on the output of each convolutional layer to normalize them and accelerate training as well as improve stability, and the activation function used is HardTanh which is a non-linear activation function.

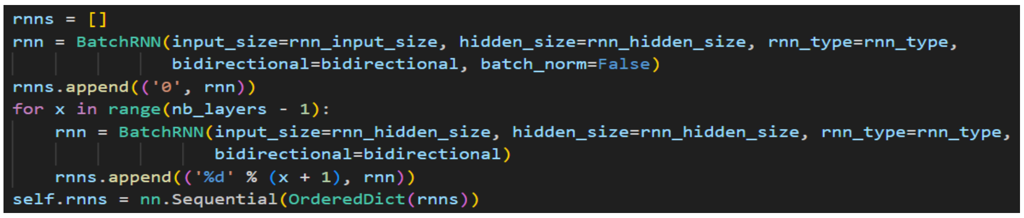


```
nn.Conv2d(1, 32, kernel_size=(41, 11), stride=(2, 2), padding=(20, 5)),
nn.BatchNorm2d(32),
nn.Hardtanh(0, 20, inplace=True),
nn.Conv2d(32, 32, kernel_size=(21, 11), stride=(2, 1), padding=(10, 5)),
nn.BatchNorm2d(32),
nn.Hardtanh(0, 20, inplace=True)
```

Recurrent Layers

As stated the Recurrent Neural Networks (RNN) are used to capture temporal dependencies in the input data, making them ideal for processing sequential data like speech.

As for the architecture, the RNN type we are using is Long Short-Term Memory (LSTM) layers, which is used to handle long-term dependencies in the sequence. We used 3 recurrent layers, each with 1024 hidden units with bidirectional configuration to process input sequence in both forward and backward directions, improving context comprehension. This is important, as bidirectional context allows the model to learn context from both directions by summing the outputs of forward and backward passes. Lastly, batch normalization is also applied between recurrent layers to stabilize training.

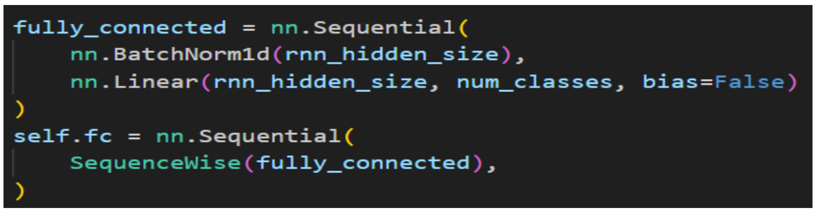


```
rnns = []
rnn = BatchRNN(input_size=rnn_input_size, hidden_size=rnn_hidden_size, rnn_type=rnn_type,
               bidirectional=bidirectional, batch_norm=False)
rnns.append(('0', rnn))
for x in range(nb_layers - 1):
    rnn = BatchRNN(input_size=rnn_hidden_size, hidden_size=rnn_hidden_size, rnn_type=rnn_type,
                   bidirectional=bidirectional)
    rnns.append(('%d' % (x + 1), rnn))
self.rnns = nn.Sequential(OrderedDict(rnns))
```

Fully Connected Layer

The fully connected layer is used to map the RNN outputs to a vector of probabilities over the character vocabulary.

For the architecture of the fully connected layer, its input size is 1024 with output size of 29 which is the Number of Classes (A to Z, space, and underscore). Batch normalization is also applied to the input of the fully connected layer for regularization.

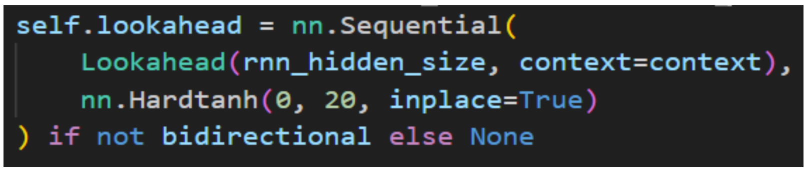


```python
fully_connected = nn.Sequential(
    nn.BatchNorm1d(rnn_hidden_size),
    nn.Linear(rnn_hidden_size, num_classes, bias=False)
)
self.fc = nn.Sequential(
    SequenceWise(fully_connected),
)
```

As for the number of trainable parameters, (Input Size x Output Size) + Output Size = (1024 x 29) + 29 = 29725 parameters.

Lookahead Layer

The purpose of this lookahead layer is applied only in unidirectional RNNs to enable the network to anticipate future inputs. This is not used in bidirectional models.

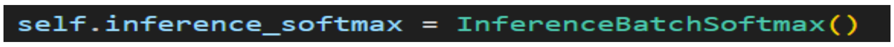


```python
self.lookahead = nn.Sequential(
    Lookahead(rnn_hidden_size, context=context),
    nn.Hardtanh(0, 20, inplace=True)
) if not bidirectional else None
```

It has input and output channels of 1024 with a kernel size of 20. The number of parameters will be 20 X 1024 = 20480 trainable parameters.

Softmax Layer

The softmax layer converts the network's raw logits into probabilities over the vocabulary. These probabilities are then used for decoding.



```python
self.inference_softmax = InferenceBatchSoftmax()
```

This layer is not trainable as it does not have any trainable parameters.

Input and Output Dimensions

As for the input, the spectrogram will have a matrix of size T x F, where T is the number of time frames and F is the number of frequency bins. Batch Dimension will be: B x 1 x F x T, where B is the batch size.

As for the output, for the character probabilities is tensors of size B x T x C, where C is the number of classes (which is 29, A to Z, space, and underscore)

Overall Model Flow

Input: An input of spectrogram of size B x 1 x F x T is passed to the convolutional layers

Convolution: The features that are extracted are flattened and reshaped into T x B x H, where H is the feature size.

Recurrent Layers: We will then process the sequence with LSTMs, outputting T x B x H

Fully Connected Layer: Maps the features to C classes. Outputting logits of shape T x B x C, where each time step predicts probabilities for each character in vocabulary.

Softmax Layer: Convert logit to probabilities over the vocabulary. In training, this layer is bypassed as raw logits are used directly by the CTC loss. In inference, output probabilities of shape T x B x C.

Training

For our speech-to-text model, training involves optimizing the model’s parameters to minimize the error between its predicted transcription and the actual correct transcriptions. The process includes defining loss function, optimizer, and learning rate scheduler.

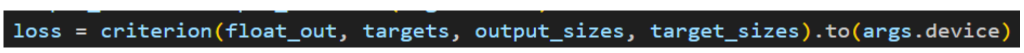
Loss function

We will be using the Connectionist Temporal Classification (CTC) Loss which is specifically designed for sequence-to-sequence problems where the alignment between input, in our case audio spectrogram, and output, which is transcriptions, is unknown.

CTC Loss introduces “blank” tokens to represent no prediction at specific time steps and finds the best alignment between predicted and real sequences by summing over all possible alignments. The formula is:

$$\mathcal{L}_{CTC} = -\log\left(\sum_{\pi \in \mathcal{A}(y)} P(\pi|x)\right)$$

Where P(π|x) is the probability of alignment π given input x, and A(y) means all valid alignments of target sequence y.

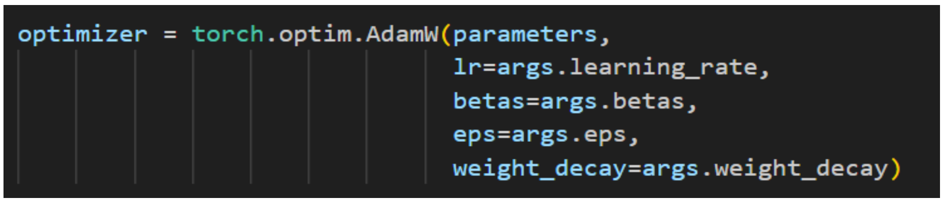


```
loss = criterion(float_out, targets, output_sizes, target_sizes).to(args.device)
```

Optimizer

We are using the AdamW optimizer which is a modified version of Adam that utilizes weight decay in the gradient update rule. This helps prevent overfitting and ensures better generalization for model performance.

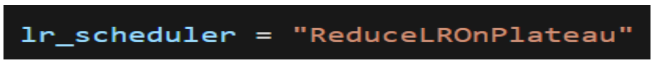
Let's take a look at our parameters, firstly the learning rate is set to 1 x 10^-5 and Betas of (0.9, 0.999), to control the moving averages of gradient and squared gradient. Epsilon is 1 x 10^-8, this is a small value to avoid division by zero, and weight decay is 1 x 10^-4, penalizing large weights for better convergence.



```
optimizer = torch.optim.AdamW(parameters,
                              lr=args.learning_rate,
                              betas=args.betas,
                              eps=args.eps,
                              weight_decay=args.weight_decay)
```

Learning Rate Scheduler

For our learning rate scheduler, we are using the ReduceLROnPlateau which reduces the learning rate when the validation loss stagnates, ensuring the model does not get stuck in a local minima.



```
lr_scheduler = "ReduceLROnPlateau"
```

It can prevent overfitting by slowing down learning when the model is close to convergence, and it improves final accuracy by fine-tuning with smaller learning rates that keep getting smaller.

Evaluation

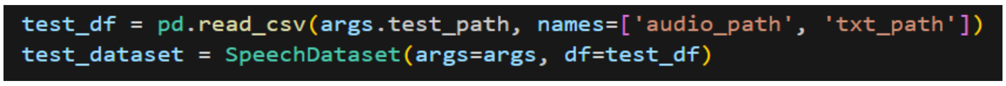
Evaluation can be considered the most important phase in any machine learning pipeline as it assesses the model's performance on unseen data, determining how well the model performs and generalizes to new inputs. For our speech-to-text system, the evaluation focuses on comparing the predicted transcription against ground-truth labels using metrics that capture both word level accuracy and character level accuracy.

Objectives of Evaluation

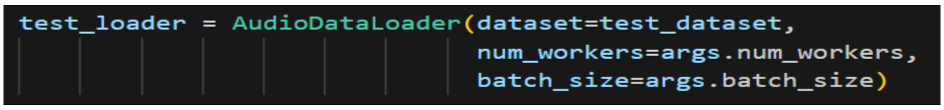
Evaluate how closely the model is able to predict transcription to match the actual transcription and ensure the model performs well on unseen data outside of the training set, as well as identify specific types of error (character or word) for future improvements.

Evaluation Pipeline

Evaluation starts with preparing the test dataset, which contains the important unseen audio files and their corresponding transcription.

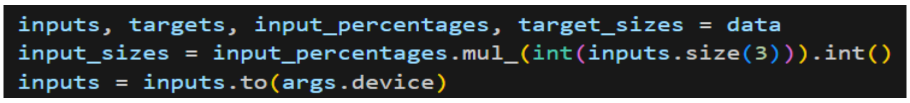


```
test_df = pd.read_csv(args.test_path, names=['audio_path', 'txt_path'])
test_dataset = SpeechDataset(args=args, df=test_df)
```

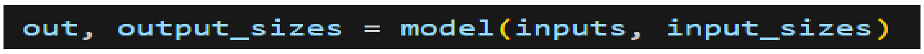


```
test_loader = AudioDataLoader(dataset=test_dataset,
                              num_workers=args.num_workers,
                              batch_size=args.batch_size)
```

The model then processes the test dataset in batches and for each batch, the inputs (spectrogram) are passed through the model, then for the output, the model outputs logits representing character probabilities for each time frame.

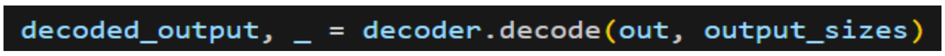


```
inputs, targets, input_percentages, target_sizes = data
input_sizes = input_percentages.mul_(int(inputs.size(3))).int()
inputs = inputs.to(args.device)
```



```
out, output_sizes = model(inputs, input_sizes)
```

Then, we will decode the model's output logits into human readable text.



```
decoded_output, _ = decoder.decode(out, output_sizes)
```

Metric Calculation

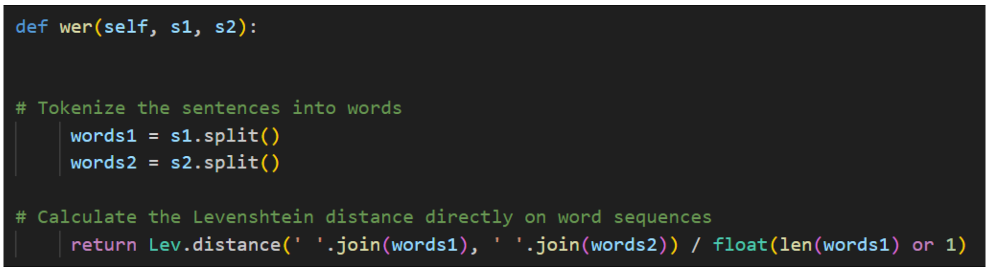
Two main key metrics will be used to evaluate transcription quality

<u>Word Error Rate (WER)</u>

WER measures the difference between the predicted transcript and the ground truth transcript at the word level with the formula,

WER = (S + D + I) / N

Where, S is the number of substitutions (words incorrectly replaced), D is the number of deletions (words omitted in the prediction), I is the number of insertions (extra words in the prediction) and N is the total number of words in the ground truth.



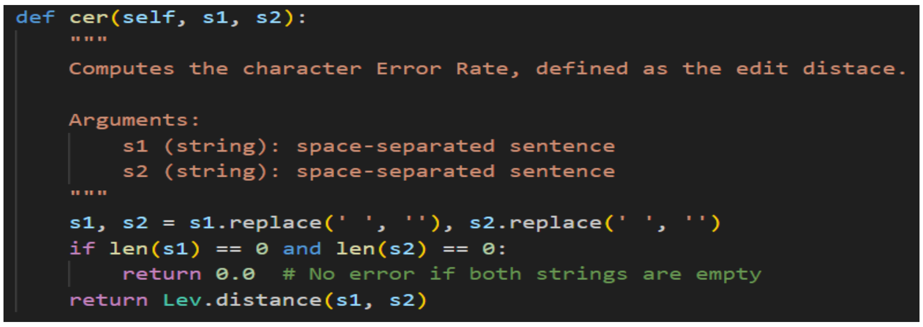
```
def wer(self, s1, s2):


# Tokenize the sentences into words
    words1 = s1.split()
    words2 = s2.split()

# Calculate the Levenshtein distance directly on word sequences
    return Lev.distance(' '.join(words1), ' '.join(words2)) / float(len(words1) or 1)
```

<u>Character Error Rate (CER)</u>

CER measures the difference at the character level, and this provides a finer-grained assessment of transcription accuracy. Where the formula is

CER = (S + D + I) / N

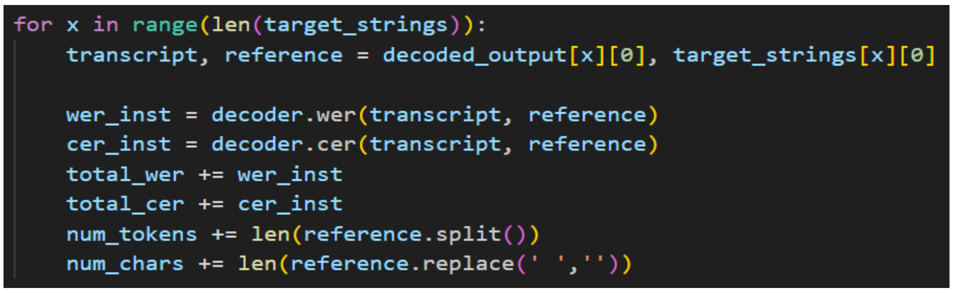
Where, S is the number of substitutions (characters incorrectly replaced), D is the number of deletions (characters omitted in the prediction), I is the number of insertions (extra characters in the prediction) and N is the total number of characters in the ground truth.



```
def cer(self, s1, s2):
    """
    Computes the character Error Rate, defined as the edit distace.

    Arguments:
        s1 (string): space-separated sentence
        s2 (string): space-separated sentence
    """
    s1, s2 = s1.replace(' ', ''), s2.replace(' ', '')
    if len(s1) == 0 and len(s2) == 0:
        return 0.0  # No error if both strings are empty
    return Lev.distance(s1, s2)
```

Logging and Aggregating Results

The WER and CER are computed for each transcription in the test set and aggregated to provide overall performance metrics.



```
for x in range(len(target_strings)):
    transcript, reference = decoded_output[x][0], target_strings[x][0]

    wer_inst = decoder.wer(transcript, reference)
    cer_inst = decoder.cer(transcript, reference)
    total_wer += wer_inst
    total_cer += cer_inst
    num_tokens += len(reference.split())
    num_chars += len(reference.replace(' ',''))
```

Designing the Novel Model

The novel model is designed to introduce an advanced emotion recognition system that uses auditory information collected from multiple datasets.Thus, this section will show the steps taken to preprocess data, extract features, and train a novel neural network architecture tailored for emotion classification.

Data Preprocessing and Augmentation

Firstly, the dataset we use are RAVDESS, CREMA-D, TESS, and SAVEE, which each of the datasets is different. Let us briefly introduce them. For CREMA-D, this dataset contains 7,442 audio recordings from 91 actors (48 male, 43 female) ages 20 to 74 which consist from various racial and ethnic origins and they said 12 sentences on six different emotions (anger, disgust, fear, happy, neutral, sad) at differing intensities (low, medium, high, unspecified). For TESS (Toronto Emotional Speech Set), two female actresses (aged 26 and 64) recorded a total of 2,800 audio files and each actress pronounced 200 target words contained in the carrier phrase "Say the word _" to represent seven emotions (anger, disgust, fear, happiness, pleasant surprise, sadness, and neutrality). In addition, the recordings are in WAV format and sorted by actress and emotion. For SAVEE (Surrey Audio-Visual Expressed Emotion), which contains recordings of four male speakers aged 27 to 31, speaking 15 lines per emotion. The dataset includes seven emotion categories (Anger, Disgust, Fear, Happiness, Sadness, Surprise, and Neutral), with a total of 120 utterances per speaker that balance phonetics and emotion-specific situations. Lastly, for RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song), the collection contains 1,440 audio files of 24 professional actors (12 males and 12 females) saying two statements, which the dataset also includes seven emotions (Calm, Happy, Sad, Angry, Fearful, Surprise, and Disgust) with two intensity levels (Normal, Strong) and a neutral expression.

Next, moving on to the feature augmentation technique. This is to improve generalization and enhance the diversity of training data with the following augmentation method. The first method is noise injection,which could add Gaussian noise to audio signals. Then, time stretching which could modify the payback speed of the audio. Next, is shifting which temporarily shifts audio samples. At last, pitch shifting which alter the pitch of audio data

Lastly, let's move on to Waveform and Spectrogram Analysis. Waveforms and spectrograms were utilised to analyse audio signals, validate augmented samples, and understand the auditory patterns associated with each emotion. Besides that, waveforms represented amplitude variations across time, emphasising intensity differences. For example, rage had sharp, high amplitudes, but melancholy was smoother and softer. Other than that, spectrograms gave time-frequency representations of energy distribution across frequencies,like happy and angry emotions had greater frequencies than sad and neutral. Augmented data such as noise addition, pitch shifting, was visually tested to verify realistic patterns, proving its usefulness for augmenting the dataset while preserving emotion-specific properties.

Feature Extraction

Feature extraction was performed using Librosa to capture essential audio characteristics. The Zero-Crossing Rate (ZCR) is a measurement of the rate at which a signal varies across zero amplitude, showing signal variability. Then, Mel-Frequency Cepstral Coefficients (MFCCs) collected timbral textures that are critical for identifying emotions. Next, Chroma Features represented pitch class profiles and highlighted tonal information. Besides that, Root Mean Square (RMS) Energy measures signal intensity and reflects emotional energy levels. Finally, Mel Spectrograms encoded frequency material in a perceptually relevant manner, facilitating the distinction of emotional emotions.

Model Architecture

A deep learning architecture was created to balance accuracy, efficiency, and generalisation in emotion recognition. The model's Convolutional Feature Extraction component used several Conv1D layers with progressively smaller kernel sizes (7, 5, 3) to capture complicated temporal auditory patterns, while MaxPooling layers reduced feature dimensionality and computational cost. Then, Batch Normalisation was used to stabilise and accelerate training. For Sequence Modelling, a Bidirectional GRU layer with 128 units is incorporated to capture contextual dependencies by processing information from both past and future frames, which was supplemented by another GRU layer with 64 units to refine sequential representations and improve feature encoding. Finally, L2 Regularisation and Classification used dropout layers to reduce overfitting and improve robustness, while Dense and Flatten layers used a softmax activation function to classify 8 distinct emotion categories, ensuring reliable prediction across diverse datasets.

Training Process

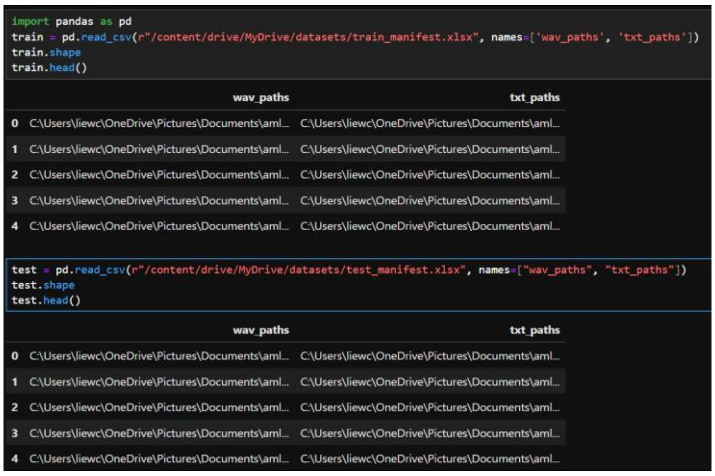
The model was trained with the Adam optimiser at a learning rate of 0.0005, which was chosen for its ability to manage non-stationary objectives. Then, Categorical cross-entropy was employed as the loss function, which is appropriate for multiclass emotion classification applications. To balance computational efficiency and effective model convergence, batch sizes of 16 and 50 epochs were used. However we have increased the batch size and epochs during tuning of the model and have reached the final batch size of 64 and epochs of 500. A second validation set was used to monitor performance and fine-tune hyperparameters, reducing the likelihood of overfitting. We will also be incorporating ReduceLROnPlateau to dynamically adjust the learning rate such that the model will be more stable during later training stages, and we have tuned the patience parameter manually and found it the best when patience is of 35.

Work Segregation

Liew Cheah Ming is in charge of developing, debugging, handling, and running the speech to text recognition code model, ensuring no issues with the code. Tay Re Zen work is on developing the novel model, which is the speech emotion recognition system, by working with Liew Cheah Ming to incorporate it into the speech to text recognition system. Kwah Wen Yao is both working on the code and report, as he is debugging and understanding the code that Liew Cheah Ming and Tay Re Zen is developing while translating the progress into the report. Lastly, Leong Yao Cheng is in charge of the whole report by making sure the content is correct and the report is formatted and tidy.

Validation/ Verification

Firstly, we will verify the file paths by checking their existence and ensuring they are correctly paired with transcript paths, ensuring no mismatch.



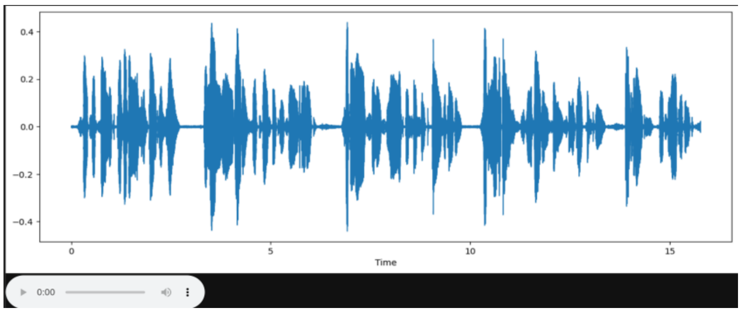
```python
import pandas as pd
train = pd.read_csv(r"/content/drive/MyDrive/datasets/train_manifest.xlsx", names=['wav_paths', 'txt_paths'])
train.shape
train.head()
```

	wav_paths	txt_paths
0	C:\Users\liewc\OneDrive\Pictures\Documents\aml...	C:\Users\liewc\OneDrive\Pictures\Documents\aml...
1	C:\Users\liewc\OneDrive\Pictures\Documents\aml...	C:\Users\liewc\OneDrive\Pictures\Documents\aml...
2	C:\Users\liewc\OneDrive\Pictures\Documents\aml...	C:\Users\liewc\OneDrive\Pictures\Documents\aml...
3	C:\Users\liewc\OneDrive\Pictures\Documents\aml...	C:\Users\liewc\OneDrive\Pictures\Documents\aml...
4	C:\Users\liewc\OneDrive\Pictures\Documents\aml...	C:\Users\liewc\OneDrive\Pictures\Documents\aml...

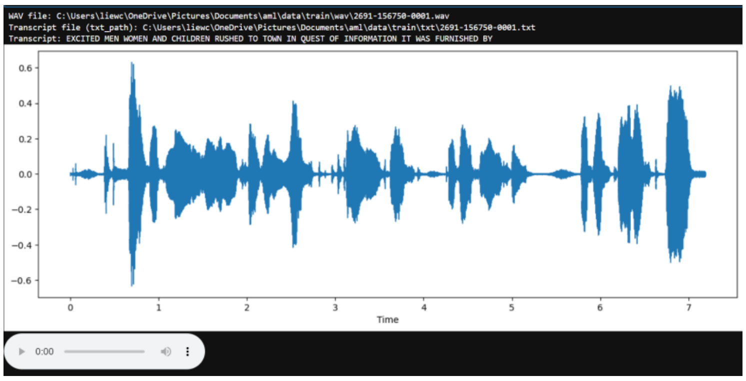
```python
test = pd.read_csv(r"/content/drive/MyDrive/datasets/test_manifest.xlsx", names=["wav_paths", "txt_paths"])
test.shape
test.head()
```

	wav_paths	txt_paths
0	C:\Users\liewc\OneDrive\Pictures\Documents\aml...	C:\Users\liewc\OneDrive\Pictures\Documents\aml...
1	C:\Users\liewc\OneDrive\Pictures\Documents\aml...	C:\Users\liewc\OneDrive\Pictures\Documents\aml...
2	C:\Users\liewc\OneDrive\Pictures\Documents\aml...	C:\Users\liewc\OneDrive\Pictures\Documents\aml...
3	C:\Users\liewc\OneDrive\Pictures\Documents\aml...	C:\Users\liewc\OneDrive\Pictures\Documents\aml...
4	C:\Users\liewc\OneDrive\Pictures\Documents\aml...	C:\Users\liewc\OneDrive\Pictures\Documents\aml...

We will then analyze a specific sample from the dataset using librosa function to read the audio file, the sample rate is returned and automatically confirmed during the loading process. An audio playback will also be generated using ipd.Audio, allowing for auditory inspection in the notebook.

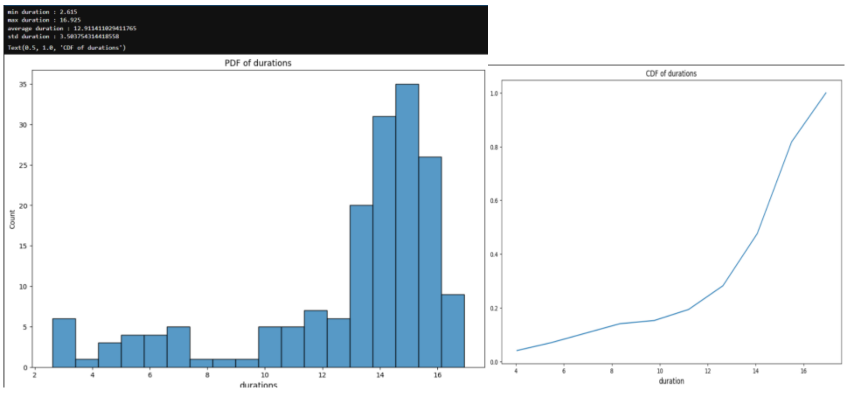


We will also be generating an audio signal by validating the path for audio file and transcript file and verifying and reading the corresponding transcript file. We will be using the same librosa function to load the audio by plotting its waveform.



```
WAV file: C:\Users\liewc\OneDrive\Pictures\Documents\aml\data\train\wav\2691-156750-0001.wav
Transcript file (txt_path): C:\Users\liewc\OneDrive\Pictures\Documents\aml\data\train\txt\2691-156750-0001.txt
Transcript: EXCITED MEN WOMEN AND CHILDREN RUSHED TO TOWN IN QUEST OF INFORMATION IT WAS FURNISHED BY
```

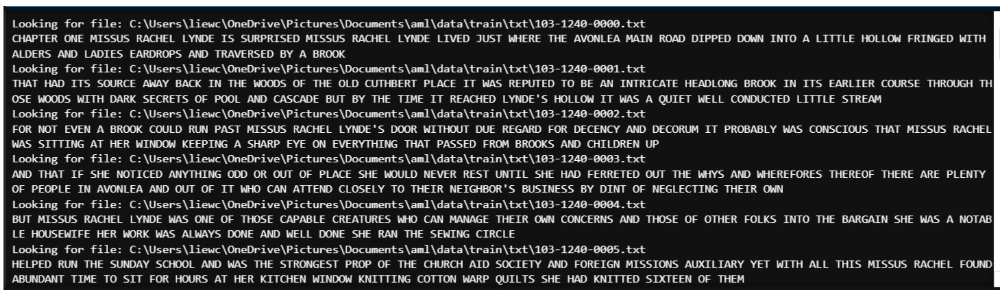
After that, the next step is to analyze the duration of audio files in the dataset to assess the variability and distribution. Mainly we will use descriptive statistics such as distribution (PDF) and cumulative distribution (CDF) of the audio durations.



```
min duration : 2.615
max duration : 16.925
average duration : 12.911411029411765
std duration : 3.503754314418558
Text(0.5, 1.0, 'CDF of durations')
```

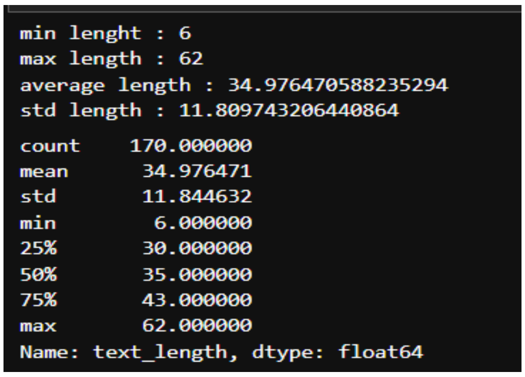
This step is to ensure outliers can be identified by inspecting the minimum and maximum values, and the mean and standard deviation will offer insights into overall variability.

Then we will conduct an iterative text file inspection, this is so that we can validate their content by reading and printing the text. We will also be constructing the full path to the transcript file using os.path.join with the target split_txt_dir.



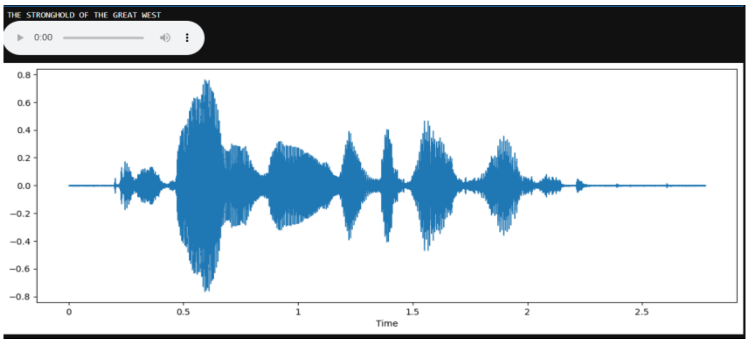
```
Looking for file: C:\Users\liewc\OneDrive\Pictures\Documents\aml\data\train\txt\103-1240-0000.txt
CHAPTER ONE MISSUS RACHEL LYNDE IS SURPRISED MISSUS RACHEL LYNDE LIVED JUST WHERE THE AVONLEA MAIN ROAD DIPPED DOWN INTO A LITTLE HOLLOW FRINGED WITH
ALDERS AND LADIES EARDROPS AND TRAVERSED BY A BROOK
Looking for file: C:\Users\liewc\OneDrive\Pictures\Documents\aml\data\train\txt\103-1240-0001.txt
THAT HAD ITS SOURCE AWAY BACK IN THE WOODS OF THE OLD CUTHBERT PLACE IT WAS REPUTED TO BE AN INTRICATE HEADLONG BROOK IN ITS EARLIER COURSE THROUGH TH
OSE WOODS WITH DARK SECRETS OF POOL AND CASCADE BUT BY THE TIME IT REACHED LYNDE'S HOLLOW IT WAS A QUIET WELL CONDUCTED LITTLE STREAM
Looking for file: C:\Users\liewc\OneDrive\Pictures\Documents\aml\data\train\txt\103-1240-0002.txt
FOR NOT EVEN A BROOK COULD RUN PAST MISSUS RACHEL LYNDE'S DOOR WITHOUT DUE REGARD FOR DECENCY AND DECORUM IT PROBABLY WAS CONSCIOUS THAT MISSUS RACHEL
WAS SITTING AT HER WINDOW KEEPING A SHARP EYE ON EVERYTHING THAT PASSED FROM BROOKS AND CHILDREN UP
Looking for file: C:\Users\liewc\OneDrive\Pictures\Documents\aml\data\train\txt\103-1240-0003.txt
AND THAT IF SHE NOTICED ANYTHING ODD OR OUT OF PLACE SHE WOULD NEVER REST UNTIL SHE HAD FERRETED OUT THE WHYS AND WHEREFORES THEREOF THERE ARE PLENTY
OF PEOPLE IN AVONLEA AND OUT OF IT WHO CAN ATTEND CLOSELY TO THEIR NEIGHBOR'S BUSINESS BY DINT OF NEGLECTING THEIR OWN
Looking for file: C:\Users\liewc\OneDrive\Pictures\Documents\aml\data\train\txt\103-1240-0004.txt
BUT MISSUS RACHEL LYNDE WAS ONE OF THOSE CAPABLE CREATURES WHO CAN MANAGE THEIR OWN CONCERNS AND THOSE OF OTHER FOLKS INTO THE BARGAIN SHE WAS A NOTAB
LE HOUSEWIFE HER WORK WAS ALWAYS DONE AND WELL DONE SHE RAN THE SEWING CIRCLE
Looking for file: C:\Users\liewc\OneDrive\Pictures\Documents\aml\data\train\txt\103-1240-0005.txt
HELPED RUN THE SUNDAY SCHOOL AND WAS THE STRONGEST PROP OF THE CHURCH AID SOCIETY AND FOREIGN MISSIONS AUXILIARY YET WITH ALL THIS MISSUS RACHEL FOUND
ABUNDANT TIME TO SIT FOR HOURS AT HER KITCHEN WINDOW KNITTING COTTON WARP QUILTS SHE HAD KNITTED SIXTEEN OF THEM
```

Next, we will analyze the length of transcript texts in the dataset by focusing on key descriptive statistics to validate the consistency and quality of the data. Outliers in text lengths are identified through the maximum and minimum values and set threshold for acceptable text lengths.



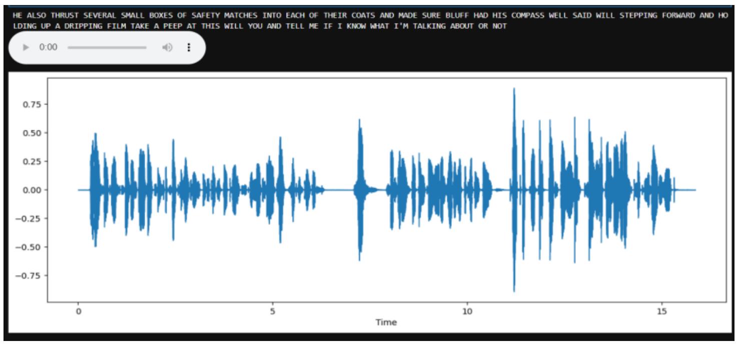
```
min lenght : 6
max length : 62
average length : 34.976470588235294
std length : 11.809743206440864
count    170.000000
mean      34.976471
std       11.844632
min        6.000000
25%       30.000000
50%       35.000000
75%       43.000000
max       62.000000
Name: text_length, dtype: float64
```

We will then be using the minimum text length in the dataset to visualize a specific sample with it to ensure the alignment of audio and text data. There is no anomalies and the audio signal is clean and correctly aligned



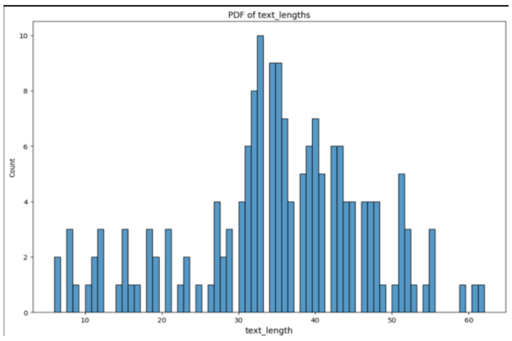
```
THE STRONGHOLD OF THE GREAT WEST
```

After the minimum, we will be testing the maximum text length sample and analyzing it. By analyzing its waveform plot, we ensure the audio signal is intact and corresponds to the transcript.



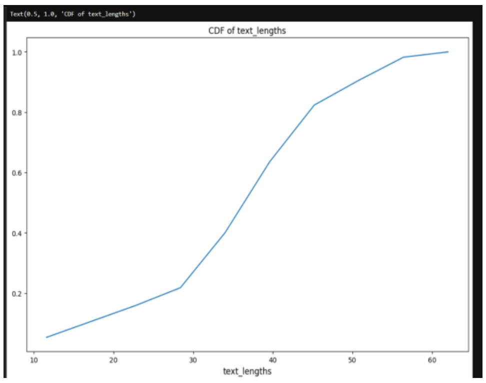
```
HE ALSO THRUST SEVERAL SMALL BOXES OF SAFETY MATCHES INTO EACH OF THEIR COATS AND MADE SURE BLUFF HAD HIS COMPASS WELL SAID WILL STEPPING FORWARD AND HO
LDING UP A DRIPPING FILM TAKE A PEEP AT THIS WILL YOU AND TELL ME IF I KNOW WHAT I'M TALKING ABOUT OR NOT
```

Moving on to the visualization of the probability distribution function of text_length, to provide insights into the variability of transcript lengths. The peak is observed at around text lengths of 30-40 words, suggesting that most transcripts are of that length.



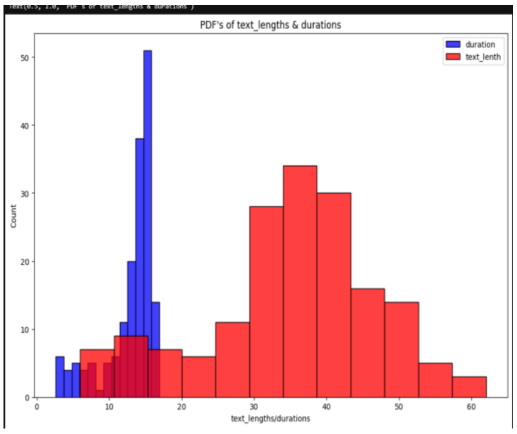
The outliers at both ends are minimal, indicating a dataset relatively consistent with transcript lengths.

After the PDF, we move on to the CDF of text lengths



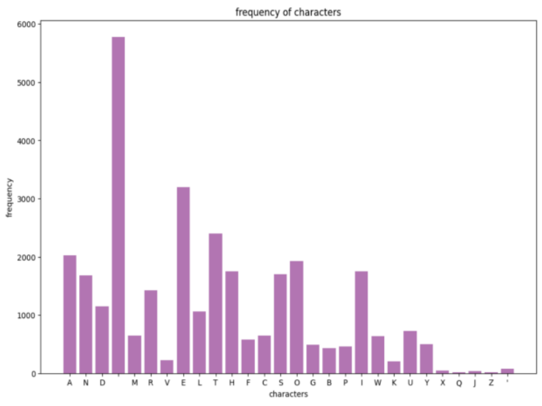
```
Text(0.5, 1.0, 'CDF of text_lengths')
```

Now let’s take a look at PDF of text lengths and audio durations. This will provide insights into the relationship between these two features. The analysis has validated the consistency of audio-text pairs and the suitability of the dataset for our speech-to-text task, where text length and audio duration typically correlate.

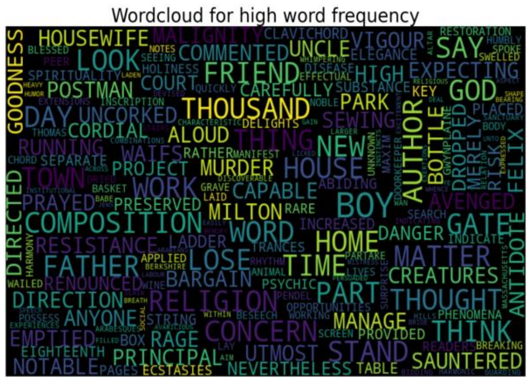


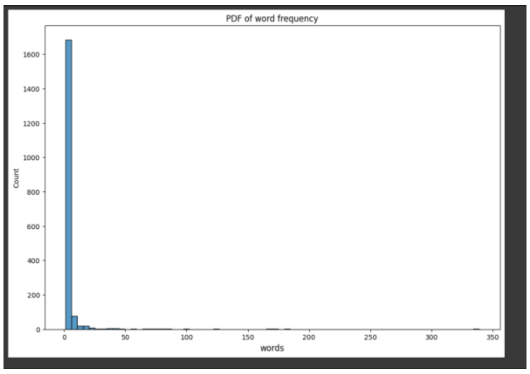
```
Text(0.5, 1.0, "PDF's of text_lengths & durations")
```

Frequency of characters in the dataset transcripts will also provide insights into character distribution and linguistic patterns of the text data, as expected the most commonly used characters are vowels and rarely used characters include punctuation X, Q, J, and Z.

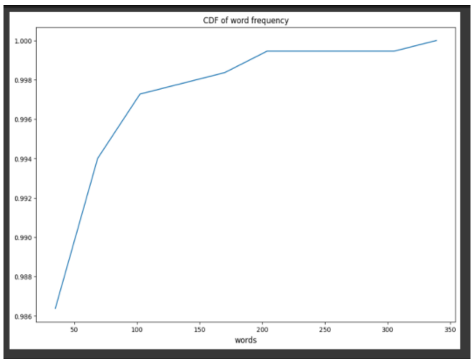


Word Cloud for High-Frequency Words where common words appear prominently (HOME, THOUSAND, COMPOSITION) reflecting their high frequency in the dataset and rare words are still visible but smaller ensuring the entire vocabulary is represented.

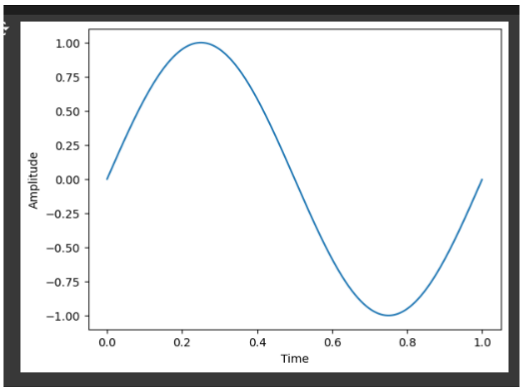




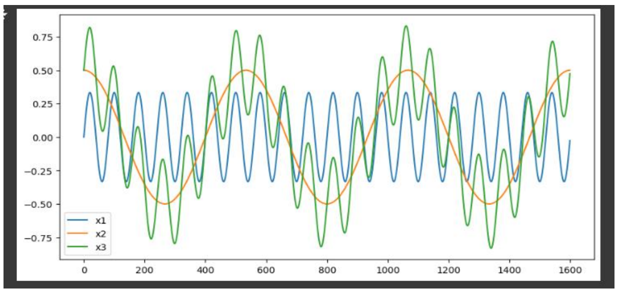
The majority of words appear less than 10 times in the dataset, indicating a long tail of infrequently used words. A few words dominate the dataset, likely representing common words or stopwords frequently used in the language.



The cumulative distribution confirms the rarity of most words in the dataset, with a small number of words accounting for the majority of occurrences. This insight is critical for downstream tasks, such as vocabulary optimization or stopword handling.

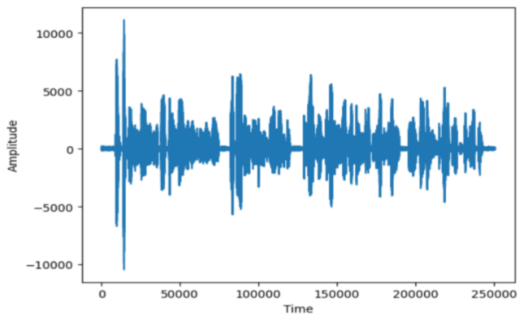


As you can see here, the sine wave successfully exhibits a smooth oscillation with an amplitude of 1 and a frequency of 1 Hz over one second. Then, The sample rate allows for high-resolution visualisation, confirming the wave's accuracy and completeness.



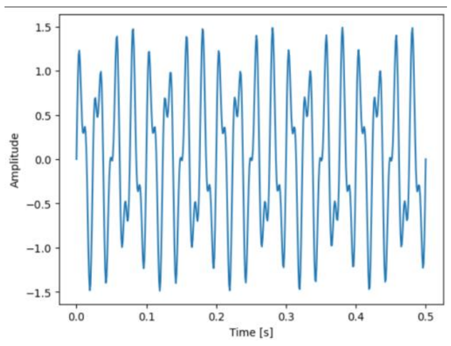
Here shown the wave 1 with a high-frequency sinusoidal wave with accurate amplitude and frequency properties. Then, there is a low-frequency cosine wave with correct amplitude and frequency characteristics in wave 2. Next, moving on to composite wave which successfully combine the properties of X1 and X2, by exhibiting both high-frequency oscillations and low-frequency modulation. For visualization, the plot effectively demonstrates the relationships and superimposition of the three waves.

Moving on to the waveform visualization which shows the amplitude variations over time which is a crucial step in verifying the integrity and quality of the audio data.

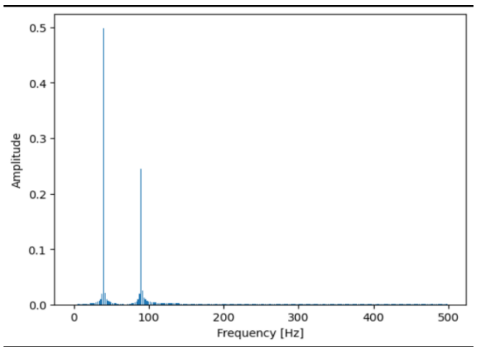


As shown, the audio signal is non-clipped and balanced with amplitudes distributed symmetrically around zero.

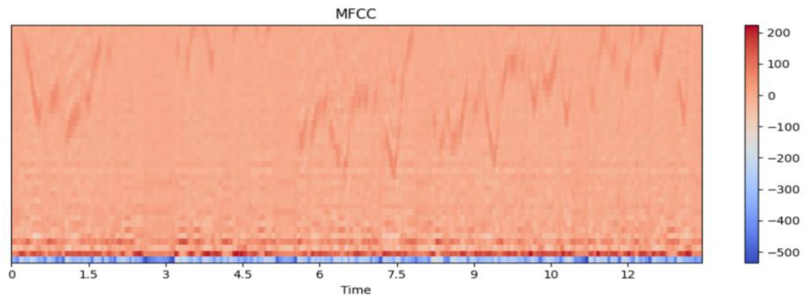
For the synthetic signal visualization,



It is composed of two sine waves with different frequencies and amplitudes. Peaks and troughs indicate constructive and destructive interference between the two waves, reflecting their relative amplitudes.



This plot visualizes frequency components of a synthetic signal using the FFT (Fast Fourier Transform) which is also a critical step in analyzing the frequency domain characteristics of a time-domain signal. As observed, the two prominent peaks correspond to the fundamental frequency (40 Hz) and the harmonic component (90 Hz) and the 40 Hz peak is higher than the 90 Hz peak suggesting that the relative amplitudes of the sin wave components in the time domain.



Lastly, the MFCC (Mel Frequency Cepstral Coefficients) of an audio signal is visualized. It is a compact representation of the spectral properties of audio. We can see higher MFCC values near the bottom of the plot meaning dominant lower-frequency components which is normal with typical speech signals.

Analysis of Training Metrics for Word Error Rate (WER) and Character Error Rate (CER)

The training process for the model provides a detailed description of performance over 54 epochs, highlighting notable patterns, advances, and obstacles. Metrics like Word Error Rate

(WER) and Character Error Rate (CER) provide important insights into the model's learning behaviour, demonstrating both improvements and regressions over time.

During the first four epochs, the model's error rates are significantly reduced. This significant decrease in both WER and CER demonstrates the model's ability to learn underlying patterns successfully. Therefore, the rapid improvements indicate that the learning rate and data representation are well linked in the early stages, allowing the model to capture key linguistic and structural properties.

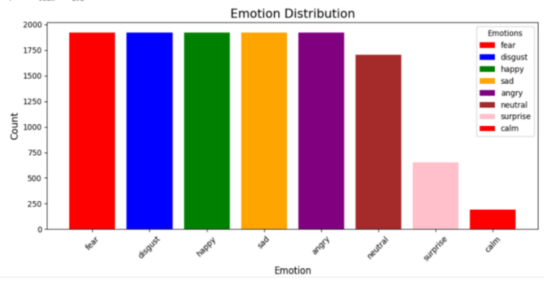
As training advances into the middle epochs such as 6 and 19 to 23, the measurements show inconsistencies, including noticeable spikes in WER and CER values. For example, in Epoch 6, WER grows to 52.925 and CER to 89.073, indicating a regression from the preceding epoch. These oscillations may suggest instability in the learning process, which could be caused by an imbalance in the learning rate, insufficient regularisation, or noisy training data. Such variations are normal in deep learning, but they emphasise the need for more resilient optimisation procedures.

In the later epochs, starting around Epoch 26, the model’s performance stabilizes significantly. WER values consistently fall below 15, with the best WER of 9.410 achieved at Epoch 29. This stabilisation indicates that the model has reached a point of consistent performance on word-level predictions. Despite the reduction in WER, CER remains disproportionately high, ranging from 80 to 90. This disparity between WER and CER indicates possible problems with character-level predictions, which could be caused by misalignment of metric aims or difficulties processing finer-grained text representations.

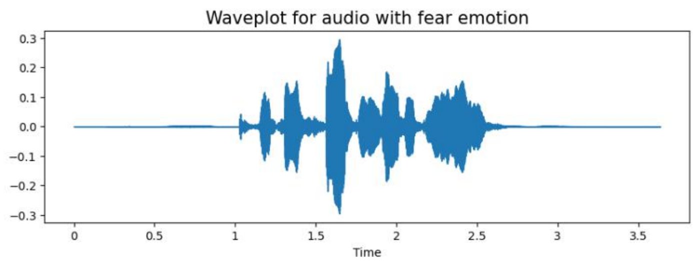
A major part of the training process is the approach for preserving the model when WER improves. This approach ensures that progress is captured at crucial milestones, which is consistent with machine learning best practices. By storing models at these points, the process prevents loss of progress due to potential regressions in subsequent epochs.

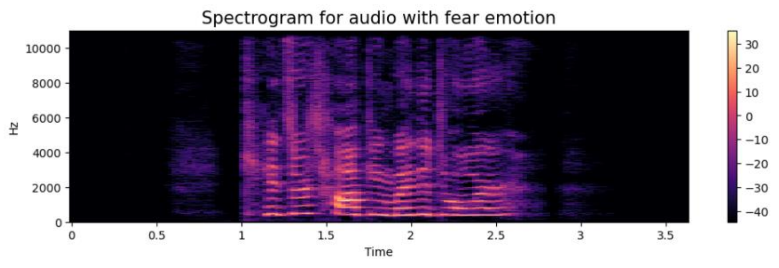
Now move on to our novelty part.

```
Emotion Distribution Table:
    Emotion  Count
0      fear   1923
1   disgust   1923
2     happy   1923
3       sad   1923
4     angry   1923
5   neutral   1703
6  surprise    652
7      calm    192
```

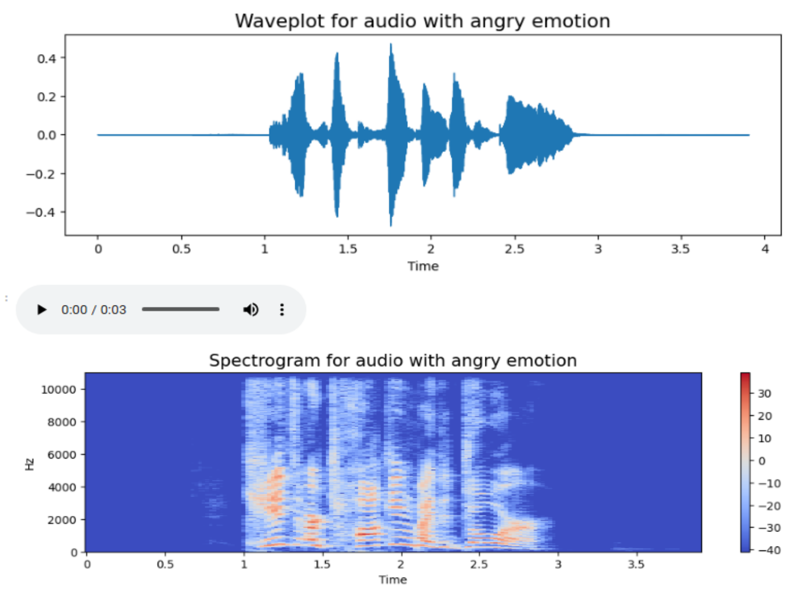


This chart and table show the distribution of emotions in the dataset. Emotions including "fear," "disgust," "happy," "sad," and "angry" are well-represented with over 1,923 samples each, while "neutral" has 1,703. However, "surprise" and "calm" are significantly under-represented, with just 652 and 192 examples, respectively. Thus, this mismatch could have an impact on the model's capacity to properly generalise for less common emotions.



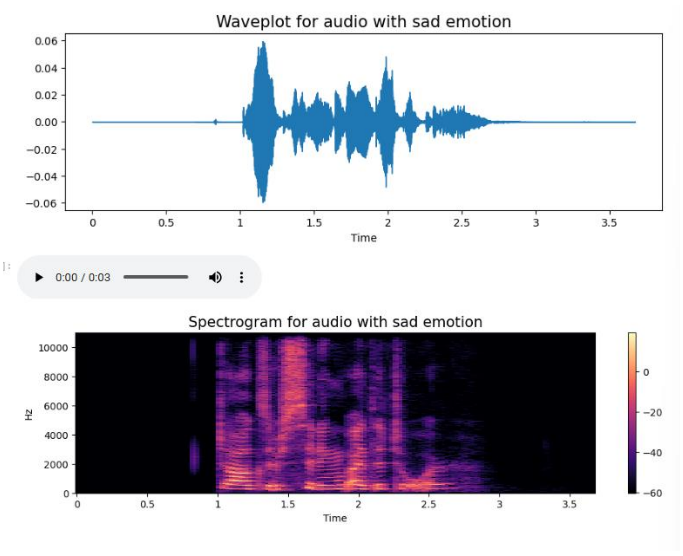


The visuals include a waveform and a spectrogram of an audio sample representing the emotion "fear." Then, the waveform shows the amplitude variations of the signal across time, indicating dynamic changes in the audio intensity. Next, the spectrogram is a time-frequency representation that shows energy distribution across frequencies throughout time. Bright regions imply increased energy, which corresponds to prominent acoustic elements associated with the "fear" emotion. In the end, these techniques aid in understanding the auditory patterns of emotions and evaluating data quality for feature extraction.



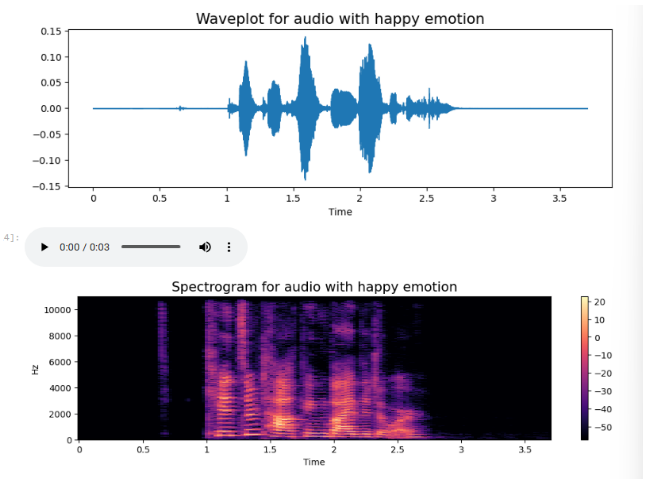
This visualisation shows the waveform and spectrogram of an audio clip displaying the emotion "anger."

The waveform shows significant amplitude variations, indicating powerful and sharp shifts in the audio stream, typical of "angry" speech. Besides that, the spectrogram shows the energy distribution across frequencies over time, with brighter parts corresponding to more energy, notably in the mid-to-high frequencies. These characteristics indicate the strong and aggressive tone patterns associated with wrath. Such visualisations aid in the analysis of auditory patterns for each emotion, ensuring that retrieved features match the expected characteristics.



This visualisation shows the waveform and spectrogram of an audio clip reflecting the emotion "sadness":

The waveform displays comparatively smaller amplitude changes, with smoother and softer transitions, which corresponds to the subdued quality of sad speech. Then, The spectrogram reveals a concentration of energy in lower frequencies, with less intensity and fewer sharp peaks than emotions such as rage or fear. The darker patches represent lower energy in higher frequencies, indicating a sombre tone. Thus, these patterns efficiently capture the acoustic intricacies associated with the feeling of melancholy, which helps to distinguish it from other emotions during feature extraction and classification.



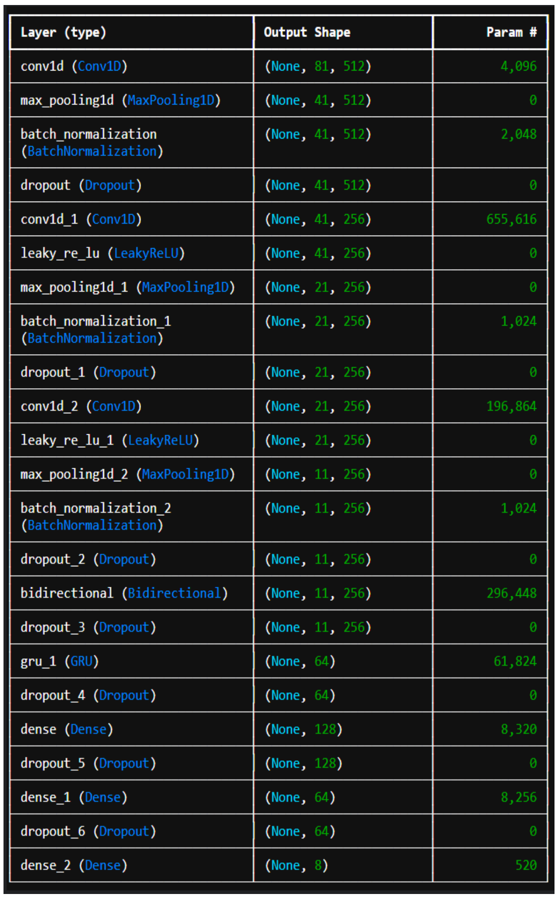
This visualisation depicts the waveform and spectrogram of an audio clip reflecting the emotion "happiness":

The waveform shows greater amplitude variations, indicating the energy and liveliness associated with cheerful speech.

The spectrogram shows concentrated energy over a wide frequency range, with prominent peaks and colourful patterns in higher frequencies. The bright parts represent the lively and dynamic tonal qualities of gladness.

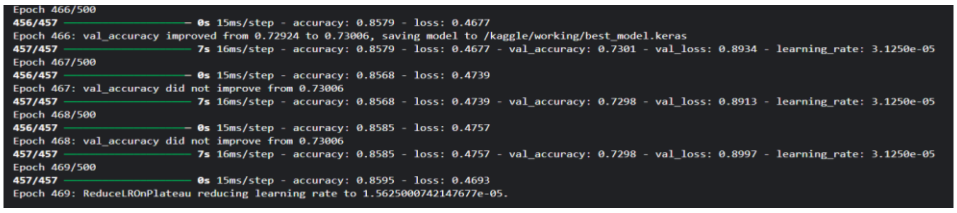
These acoustic properties help to capture the essence of happiness, allowing for more accurate emotion recognition during analysis and modelling.

Model summary:



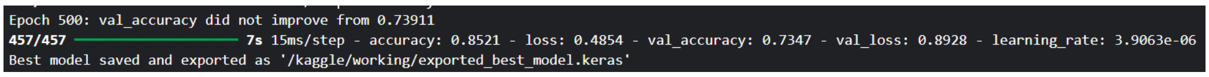
Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 81, 512)	4,096
max_pooling1d (MaxPooling1D)	(None, 41, 512)	0
batch_normalization (BatchNormalization)	(None, 41, 512)	2,048
dropout (Dropout)	(None, 41, 512)	0
conv1d_1 (Conv1D)	(None, 41, 256)	655,616
leaky_re_lu (LeakyReLU)	(None, 41, 256)	0
max_pooling1d_1 (MaxPooling1D)	(None, 21, 256)	0
batch_normalization_1 (BatchNormalization)	(None, 21, 256)	1,024
dropout_1 (Dropout)	(None, 21, 256)	0
conv1d_2 (Conv1D)	(None, 21, 256)	196,864
leaky_re_lu_1 (LeakyReLU)	(None, 21, 256)	0
max_pooling1d_2 (MaxPooling1D)	(None, 11, 256)	0
batch_normalization_2 (BatchNormalization)	(None, 11, 256)	1,024
dropout_2 (Dropout)	(None, 11, 256)	0
bidirectional (Bidirectional)	(None, 11, 256)	296,448
dropout_3 (Dropout)	(None, 11, 256)	0
gru_1 (GRU)	(None, 64)	61,824
dropout_4 (Dropout)	(None, 64)	0
dense (Dense)	(None, 128)	8,320
dropout_5 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dropout_6 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 8)	520

As shown below, the best model will be saved to best_model.keras based on the validation accuracy and the learning rate will be reduced if the validation loss has hit a plateau to avoid being stuck.



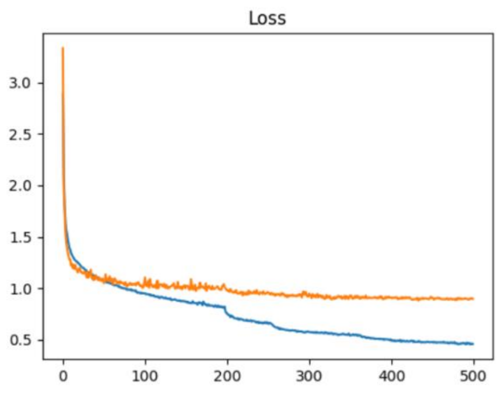
```
Epoch 466/500
456/457 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.8579 - loss: 0.4677
Epoch 466: val_accuracy improved from 0.72924 to 0.73006, saving model to /kaggle/working/best_model.keras
457/457 ━━━━━━━━━━━━━━━━━━━━ 7s 16ms/step - accuracy: 0.8579 - loss: 0.4677 - val_accuracy: 0.7301 - val_loss: 0.8934 - learning_rate: 3.1250e-05
Epoch 467/500
456/457 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.8568 - loss: 0.4739
Epoch 467: val_accuracy did not improve from 0.73006
457/457 ━━━━━━━━━━━━━━━━━━━━ 7s 16ms/step - accuracy: 0.8568 - loss: 0.4739 - val_accuracy: 0.7298 - val_loss: 0.8913 - learning_rate: 3.1250e-05
Epoch 468/500
456/457 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.8585 - loss: 0.4757
Epoch 468: val_accuracy did not improve from 0.73006
457/457 ━━━━━━━━━━━━━━━━━━━━ 7s 16ms/step - accuracy: 0.8585 - loss: 0.4757 - val_accuracy: 0.7298 - val_loss: 0.8997 - learning_rate: 3.1250e-05
Epoch 469/500
457/457 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.8595 - loss: 0.4693
Epoch 469: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
```

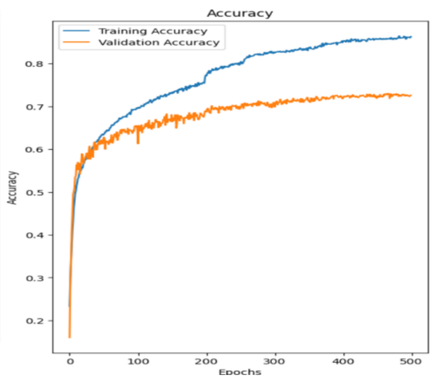
With our best validation accuracy of 0.73911 and lastly the saved best model will be exported for download.



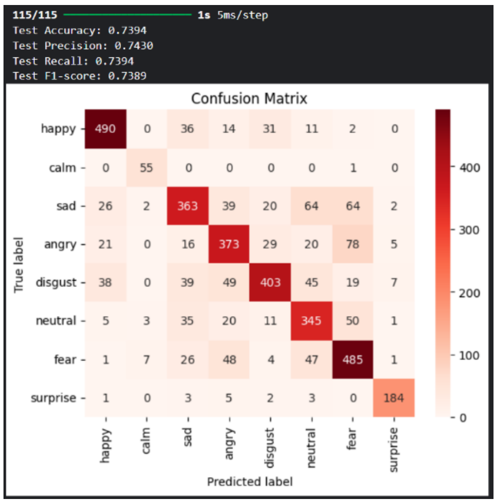
```
Epoch 500: val_accuracy did not improve from 0.73911
457/457 ━━━━━━━━━━━━━━━━━━━━ 7s 15ms/step - accuracy: 0.8521 - loss: 0.4854 - val_accuracy: 0.7347 - val_loss: 0.8928 - learning_rate: 3.9063e-06
Best model saved and exported as '/kaggle/working/exported_best_model.keras'
```

We will also plot the training and validation loss and accuracy to better visualize for underfitting or overfitting.





From above plotting, we can see that the validation loss is stuck at around 0.9 and with the training loss around 0.5 and with a trend of going lower. We can also observe validation accuracy converging around 0.74 with the training accuracy seeing an increase in trend. This is a sign of overfitting where the model is too fitted to the training data and is not generalizing well to unseen data



```
115/115 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step
Test Accuracy: 0.7394
Test Precision: 0.7430
Test Recall: 0.7394
Test F1-score: 0.7389
```

.

Above is the test accuracy, which is 0.7394, which is decent but can be improved. The test precision is 0.7430, test recall is 0.7394 and test F1-score is 0.7389. From the confusion matrix, we can see that the model has performed reasonably well, with some mistakes, such as predicting sad labels as neutral and fear, and struggling to predict fear labels as angry and neutral. The reason for this might be due to overlapping acoustic features between these emotions, such as similar pitch or tone variations.

Conclusion

In our project, we have proposed a novel hybrid CNN-GRU model speech emotion recognition in conjunction with our speech to text model. By using multiple datasets (RAVDESS, CREMA-D, TESS, and SAVEE), we extracted meaningful features and implemented preprocessing and augmentation techniques. Our approach has demonstrated the ability to classify seven emotions with an accuracy of 73% and detecting speech to text with a Word Error Rate of 9.410, which can be easily improved, given more computational resources and time. The analysis of the result has provided several strengths of the model, but it has also revealed areas for improvements, such as addressing the class imbalances and reducing the overfitting of our speech emotion recognition and we can improve the WER of our speech to text model by addressing the sparsity of dataset by having more diverse and more dataset as well as more computational resource. Despite these challenges, this project has laid the groundwork for more human-like interactions such as a court reporter recording a record of court proceedings in a courtroom or an interrogation officer recording an interrogation or in applications such as mental health monitoring where the words a patient said is important as well the tone and emotion he/ she said it in, and lastly customer service as well.

Future Work

Future work for our project aims to address the limitations we have faced and explore new techniques to enhance our models' performance.

First, utilising advanced computational resources such as GPUs or TPUs may greatly improve model training and inference times. This would allow for more comprehensive hyperparameter adjustment and experimentation with deeper, more complex neural network architectures, potentially resulting in higher accuracy and efficiency.

Second, improving dataset diversity and quality is critical to increasing the model's robustness. Using larger datasets with a diversity of accent, speech patterns, and environmental noise can assist reduce the Word Error Rate (WER) in voice-to-text operations. Furthermore, domain-specific datasets, such as those designed for courtrooms or mental health monitoring, could enhance the system's specialised and effective performance in real-world applications.

MARKING RUBRICS

Component Title	Final project Score and Descriptors					
No.		**Poor**	**Average**	**Excellent**	**Total Marks**	**Marks**
Component 1: Project Development						
		0 - 4	**5 - 7**	**8 - 10**	**10**	
1	Code quality (CLO5)	Codes are poorly structured. Not fully functional. Not documented.	Codes are sufficiently documented and mostly functional. Satisfactorily structured	Codes are fully functional and well structured and documented		
		0 - 4	**5 - 7**	**8 - 10**	**10**	
2	Implementation of algorithm related to ML (CLO2)	Incomplete development of the method.	Complete development of method.	Completed and enhanced the novel developed methods		
		0-2	**3**	**4-5**	**5**	
3	Completeness and complexity of your project (CLO5)	The functionalities that are implemented are non-functional	All functionalities are implemented and functional	All the functionalities are fully implemented and functional Implemented extra functionalities.		
				subtotal	25	
Component 2: Project Report + Project Demonstration						
		0 - 4	**5 - 7**	**8 - 10**	**10**	
1	Project Report (CLO2)	Poor writing quality Poor or no formatting /	Satisfactory writing quality, grammar and flow. Substantial	Good writing quality, grammar Well formatted and good		

		presentation Lack of technical content	technical content with used method.	presentation Demonstrate excellent technical content		
		0 - 4	**5 - 7**	**8 - 10**	**10**	
2	Presentation (CLO3)	Poor quality slides Poor time management Speech that is unclear	Satisfactory quality slides Speech that is satisfactory and understandable	High quality slides Good time management Speech that is clear and impactful		
		0-2	**3**	**4-5**	**5**	
3	Demonstration (CLO3)	Poor demonstration that is unclear The developed application/ research is not functioning/implemented fully	Satisfactory demonstration The developed application/ research is partially functioning/ implemented fully	Good demonstration The developed application/ research is fully functioning and logical		
				subtotal	**25**	
				TOTAL	**50**	

Note to students: Please include the marking rubric when submitting your coursework.