Boosting Customer Experience: Enhancing Speech Emotion Recognition for Improved Service Interactions

1. Introduction / Problem Statement/ Motivation **4**
2. Literature review **5**
2.1. Speech Emotion Recognition through Hybrid Features 5
2.1.1. Primary insights 5
2.1.2. Secondary insights 5
2.1.3. Databases used 6
2.1.4. Additional details of research 6
2.1.5. What can be done 6
2.2. Speech Emotion Recognition using Attention Model 7
2.2.1. Primary insights 7
2.2.2. Secondary insight 7
2.2.3. Additional insight 8
2.2.4. Databases used 8
2.2.5. What can be done 8
2.3. Speech emotion recognition using deep 1D & 2D CNN LSTM networks 9
2.3.1. Primary Insights 9
2.3.2. Secondary Insights 9
2.3.3. Databases used 9
2.3.4. Limitations/gaps in research 10
2.3.5. What can be done further 10
2.4. Implementation and Comparison of Speech Emotion Recognition System using Gaussian Mixture Model (GMM) and K- Nearest Neighbour (KNN) techniques 11
2.4.1. Primary insights 11
2.4.2. Secondary insights 11
2.4.3. Databases used 11
2.4.4. Additional details of research 11
2.4.5. What can be done 11
3. Datasets **12**
4. Methodology **13**
4.1. Data Pre-Processing 13
4.1.1. Data Integration 13
4.1.2. Exploratory Data Analysis 13
4.1.3. Data Augmentation 14
4.2. Train/Test Split 15
4.3. Resampling Techniques 17
4.4. Machine Learning Models 18
4.4.1. k-Nearest Neighbours (KNN) 18
4.4.2. Convolutional Neural Network (CNN) 20
4.4.3. Recurrent Neural Network (RNN) 22
4.5. Feature Selection Analysis 24
5. Results and Discussions **28**
5.1. Evaluation metrics 28
5.2. Project Phases 28
5.3. k-Nearest Neighbours (kNN) 29

5.4. Convolutional Neural Network (CNN) 40
5.5. Recurrent Neural Network (RNN) 52
6. Conclusion and Future Work **60**
6.1. Model Comparison 60
6.2. Limitations 62
6.3. Future Work 63
6.4. Takeaways 63
7. Reference **64**

1. Introduction / Problem Statement/ Motivation

In today's competitive business landscape, having good customer service interactions helps to bring in greater profits and foster customer loyalty. Reports show that businesses that focus on enhancing customer service interactions enjoy a revenue growth of 4% to 8% above their market average (Bain & Company, 2015). Conversely, 65% of customers consider switching to another brand with bad customer service interactions (Khoros, 2023).

However, many businesses failed to fully unlock the potential of Speech Emotion Recognition (SER) technology in delivering empathetic and timely customer interactions with a urgency ranking system that quickly deals with angry or sad customers who usually need more immediate assistance.

Therefore, we hope to modify the current SER approach, optimising its computational efficiency and urgency ranking. Using deep learning models like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) and K-Nearest Neighbors (KNN) , we will explore various techniques to reduce computational complexity while preserving accuracy.

2. Literature review

2.1. Speech Emotion Recognition through Hybrid Features

2.1.1. Primary insights

For the Emo-DB, SAVEE, and RAVDESS datasets, employing a hybrid MFCCT feature, which merges MFCCs with time-domain attributes, together with a streamlined CNN architecture (featuring three 1D convolutional layers, activation, dropout, max-pooling layers, and a fully connected layer) surpassed the performance of using either MFCCs or time-domain features alone, achieving accuracies of 97%, 93%, and 92%, respectively (Alluhaidan et al., 2023).

2.1.2. Secondary insights

The creation of hybrid MFCCT features from audio recordings unfolds in two primary stages: (1) the extraction of MFCC features, followed by (2) the integration of MFCC with time-domain features.

(1) Initially, MFCC features are extracted to encapsulate the characteristics of the vocal tract that are unique to each emotion. To minimise the loss of information, each speech utterance is segmented into 25ms frames, with a 10ms overlap between adjacent frames.

(2) Subsequently, to compile MFCCT features for Speech Emotion Recognition, the MFCC features are categorised into bins consisting of 1500 rows. From these, 12 time-domain features are extracted, and an algorithm is utilised to construct a feature matrix. In this matrix, the row count reflects the product of the number of bins and 12, while the column count corresponds to the number of audio files per emotion. This method is designed to bolster the model's accuracy.

The classification performance in all experiments was evaluated using weighted accuracy and AUCROC(Area Under the Receiver Operating Characteristics).

Five experiments evaluated the effectiveness of SER models using hybrid MFCCT features. The first experiment demonstrated that the CNN model surpassed other classifiers (KNN, RF, J48, NB, SVM), achieving weighted accuracies of 97%, 92.6%, and 91.4% across the Emo-DB, SAVEE, and RAVDESS datasets, respectively. The second showed MFCCT features improving weighted accuracy by approximately 50% over MFCC and time-domain features. The third identified a binning size of 1500 as optimal for maximising weighted accuracy with MFCCT features. The fourth found that using 12 time-domain features to generate MFCCT features resulted in the highest weighted accuracy, outperforming configurations with fewer time-domain features. Lastly, the fifth experiment established that the SER model employing MFCCT features and a CNN architecture exceeded baseline models in overall accuracy and efficiency across all tested datasets, despite a slight reduction in accuracy for specific emotions like boredom.

2.1.3. Databases used

The EMO-DB database contains 535 German audio clips, recorded by 10 professional speakers and categorised into seven emotions, with a 16kHz sampling rate. The RAVDESS dataset includes 1440 English audio files by 24 actors, covering eight emotions, recorded at a 48kHz sample rate. SAVEE includes 480 English utterances by 4 male speakers from CVSSP, across seven emotional categories.

2.1.4. Additional details of research

The research utilises silence removal and pre-emphasis techniques for data pre-processing. By increasing the power of high frequencies in speech signals and leaving low frequencies unchanged, the pre-emphasis technique can enhance the signal-to-noise ratio. This is achieved through the use of a high-pass filter, finite impulse response(FIR).

2.1.5. What can be done

Exploring the use of Recurrent Neural Networks (RNNs) in SER to leverage their ability to handle sequential data effectively. RNNs could potentially improve the model's performance by better capturing the temporal dynamics in speech data. Conducting a thorough comparison analysis of Speech Emotion Recognition (SER) methods that are based on deep learning (DL) approaches, employing a variety of datasets. This would provide insights into the relative performance and applicability of different DL models across diverse emotional speech datasets.

2.2. Speech Emotion Recognition using Attention Model

2.2.1. Primary insights

End-to-end unified models of different neural network architectures are able to achieve better accuracy than standard models such as GMM, SVM etc. The model comprises 2-dimensional Convolutional Neural Network for feature extraction, Long Short-term Memory network for sequential processing and Attention Mechanism for focusing on important features of the input (J. Singh et al., 2023) .

The model was able to achieve scores of at least 80% in terms of precision, recall and F1 score for 8 emotion labels as shown in **Image 1**.

Table 5. Overall emotional level score metrics.

Emotional Label	Precision (Percentage)	Recall	F1-Score
Angry	0.94	0.90	0.92
Calm	0.81	0.93	0.86
Disgust	0.90	0.94	0.92
Fear	0.93	0.86	0.89
Happy	0.88	0.88	0.89
Neutral	0.89	0.93	0.91
Sad	0.93	0.86	0.89
Surprise	0.89	0.93	0.91
Macro Average	0.89	0.9	0.9
Weighted Average	0.9	0.9	0.9
Accuracy			0.90

Image 1. Screenshot of emotion recognition scores from the research paper

In addition, it has an accuracy percentage of 90.19%, outperforming other models as shown in the **Image 2** below.

Table 6. Accuracy comparison with state-of-art studies.

Method/Model	Reference	Dataset Used	Accuracy Percentage
GMM	Kandali et al. (2008) [25]	Recorded	76.50%
SVM	Shen et al. (2011) [26]	Berlin	82.50%
SVM	Aljuhani et al. (2021) [28]	Custom	77.14%
CNN-1D	Li et al. (2019) [32]	RAVDESS	76%
1D-DCNN	Kwon (2021) [33]	EMO-DB	90%
CNN 1D + LSTM	Basu et al. (2017) [35]	EMO-DB	80%
CNN + Attention	Peng et al. (2020) [45]	IEMO-CAP	76.36%
CNN 3D + Attention	Yoon et al. (2018) [47]	MSP-IMPROV	55.70%
LSTM	Chopra et al. (2021) [46]	TESS, EMODB, SAVEE, RAVDESS	67%
LSTM + Attention + CNN-2D	PROPOSED	RAVDESS	74.44%
LSTM + Attention + CNN-2D	PROPOSED	SAVEE	57.50%
LSTM + Attention + CNN-2D	PROPOSED	TESS	99.81%
LSTM + Attention + CNN-2D	PROPOSED	Customized (RAVDESS + SAVEE + TESS)	90.19%

Image 2. Screenshot of accuracy comparison among various models from the research paper

2.2.2. Secondary insight

There are a variety of spectral features available in a speech signal for analysis, however, the paper demonstrated that using solely Mel-Frequency Cepstral Coefficient (MFCC) was sufficient to achieve high accuracy, rather than combining various features which will lead to increased computational requirements and complexity. Experiments were conducted with

different combinations of features and solely using MFCC, a 1 dimensional Convolutional Neural Network was able to achieve 85% accuracy. Even though it is not the highest, the accuracy did not deviate far from the highest accuracy score which is 86%.

2.2.3. Additional insight

Data augmentation seems to enhance the accuracy of the model as it can increase the variation between emotions in the dataset.

Attention Mechanism helps increase accuracy as it enables the model to identify small variations in speech features. This mechanism is especially useful as the spectrogram for neutral, surprise, sadness and fear are quite similar. In addition, as Attention Mechanism focuses on the relevant part of the input, it is less computationally heavy as it will disregard irrelevant parts of the input.

2.2.4. Databases used

A custom database was used. This database is a combination of different well known datasets namely TESS, SAVEE and RAVDESS. The 3 databases were combined as it provides more data, and diversity of speeches for model training.

2.2.5. What can be done

Usage of multi-model to identify emotion from both audio and visual inputs.

2.3. Speech emotion recognition using deep 1D & 2D CNN LSTM networks

2.3.1. Primary Insights

This research introduces the novel 1D and 2D CNN LSTM networks which outperform previous approaches in terms of speech emotion recognition (Refer to **Image 3**). The designed networks capture both local and global features unlike many other models that just concentrate on low-level features or a single emotion-related feature. These networks have strong generalisation abilities, suggesting potential applications in various fields, including healthcare (Jian-Feng et al., 2019).

Table 8. The comparison of average recognition accuracy of 2D CNN LSTM network conducted on Berlin EmoDB with other well-established feature representations and methods. The best performances are indicated in boldface.

Research work	Accuracy (speaker-dep)	Accuracy (speaker-indep)
Wu et al. [10]	91.6	85.8
Zhengwei Huang et al. [12]	88.3	85.2
Huang, Yongming, et al. [13]	75.5	–
Semiye Demircan et al. [14]	/	92.9
Our Work	**95.33**	**95.89**

Table 9. The comparison of recognition accuracy of 2D CNN LSTM network conducted on IEMOCAP database with other well-established feature representations and methods. The best performances are indicated in boldface.

Research work	Accuracy (speaker-dep)	Test Accuracy (speaker-indep)
W. Q. Zheng et al. [21]	/	40.02
Yelin Kim et al. [53]	73.78	/
Our Work	**89.16**	**52.14**

Image 3. Screenshot of Comparison Table: Accuracy of Speech Emotion Recognition by 1D and 2D CNN LSTM Networks Compared to Other Models

2.3.2. Secondary Insights

This study enhances the 1D CNN LSTM network by introducing a local feature learning block (LFLB) for extracting features from raw audio. Adding a LSTM layer enables capturing long-term dependencies within these features, allowing for the learning of emotion features from audio for the first time. Nevertheless, the newly introduced 2D CNN LSTM network outperforms the 1D network by capturing both local and global information from log-mel spectrograms, treating them as sequences processed by LSTM layers.

In conclusion, both networks effectively learn emotion features, with the 2D network achieving notably superior results by leveraging local and global information from log-mel spectrograms.

2.3.3. Databases used

This study employs the Berlin Emotion Database (EmoDB) and the Interactive Emotional Dyadic Motion Capture (IEMOCAP) database, comprising acted emotional speech data from various emotions and actor pairs, to recognize emotions and extract log-mel spectrograms.

2.3.4. Limitations/gaps in research

One limitation is the lack of explanation on how the networks recognize emotions, leaving a "black box" in their functioning. While efforts to understand this "black box" are ongoing, achieving higher accuracy in speech emotion recognition remains a challenge. Despite the 2D CNN LSTM network achieving around 95% accuracy, the quest for higher accuracy in speech emotion recognition persists as a challenge.

2.3.5. What can be done further

Further research could focus on enhancing the interpretability of CNN LSTM networks for speech emotion recognition and applying the outcomes to foster tangible social impact and improve individual well-being.

2.4. Implementation and Comparison of Speech Emotion Recognition System using Gaussian Mixture Model (GMM) and K- Nearest Neighbour (KNN) techniques

2.4.1. Primary insights

The GMM method outperforms KNN, achieving a remarkable 92% accuracy in identifying 'angry' emotions but only 25% for 'surprise'.

On the other hand, KNN showed a 90% accuracy in detecting 'happy' emotions but showed weaker performance for 'fear' and 'surprise.' It has difficulties differentiating 'happy' from 'neutral' emotions, a challenge also present in human perception under certain conditions. Nonetheless, KNN offers rapid processing, making it an attractive choice for time-constraint applications.

Yet, when evaluating based on Precision and F-measure, GMM proves superior, affirming its robustness and efficiency within the speech emotion recognition framework (Lanjewar et al., 2015).

2.4.2. Secondary insights

Pitch plays a critical role in influencing emotion recognition, with sentence-level pitch features outperforming voiced-level statistics in terms of accuracy and robustness. GMM surpasses HMM, KNN, and FFNN in the BES, achieving 76% accuracy, compared to 71%, 67%, 66% for HMM, KNN and FFNN respectively. To achieve better discriminative results, incorporating longer speaker prompts and natural speech into the database is beneficial. Notably, identifying simulated emotions proves easier than recognizing natural emotions.

2.4.3. Databases used

Berlin Emotion Speech Database (BES). This database comprises emotional content, compiled from audio recordings featuring ten actors across seven emotions.

2.4.4. Additional details of research

Employ Mel Frequency Cepstrum Coefficients (MFCC) for feature extraction, as preliminary studies suggest they exhibit lower noise sensitivity compared to other commonly used parameters. Extracting additional speech features like Pitch and Wavelet further enhances this process.

2.4.5. What can be done

New techniques can be discovered in which more classifiers can be fused together to serve the best recognition rates.

3. Datasets

In this project, we will develop machine learning models using three versions of the combined dataset sourced from three Kaggle repositories. The first version, termed "Non-Augmented Data," comprises the original integrated data without any modifications. The second version, termed "Augmented Data," will be generated after applying data augmentation techniques to the dataset. Finally, the third version, termed "Merged Data," will combine the non-augmented and augmented datasets to create a comprehensive dataset for model training and evaluation.

The three Kaggle repositories are as follows:

1. CREMA-D Dataset
This is an emotional multimodal actor dataset which comprises 7,442 original clips from 91 actors. These clips were from 48 male and 43 female actors between the ages of 20 and 74 coming from a variety of races and ethnicities. Actors spoke from a selection of 12 sentences. The sentences were presented using one of six different emotions (Anger, Disgust, Fear, Happy, Neutral, and Sad) and four different emotion levels (Low, Medium, High, and Unspecified) (Lok, E. J, 2019).

2. TESS Dataset
This set of 200 target words were spoken in the carrier phrase "Say the word _' by two actresses (aged 26 and 64 years) and recordings were made of the set portraying each of seven emotions (anger, disgust, fear, happiness, pleasant surprise, sadness, and neutral). There are 2800 data points (audio files) in total (Lok, E. J, 2019).

3. RAVDESS Dataset
The third dataset contains 1440 files: 60 trials per actor x 24 actors = 1440. The RAVDESS contains 24 professional actors (12 female, 12 male), vocalising two lexically-matched statements in a neutral North American accent. Speech emotions include calm, happy, sad, angry, fearful, surprise, and disgust expressions. Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression (Livingstone, S. R, 2019).

4. Methodology

4.1. Data Pre-Processing

4.1.1. Data Integration

Given that we are using 3 separate datasets, there was a need for us to consolidate the different data into a single file for our models to reference. The data that were present in each data source included the source audio file, in .wav format, and the respective emotion associated with each audio file. The integrated data file that we created contains a column containing the relative file path for each audio file and the emotion label tagged to that audio file. The relative file path is a new column added by the team as it would be necessary later on when using the Python library Librosa for extracting information from the audio files.

4.1.2. Exploratory Data Analysis

In the exploratory data analysis (EDA) phase for speech emotion recognition, we conducted feature extraction and investigated speech characteristics differences between male and female speakers.

Using the Librosa package, we extracted five crucial features essential for capturing emotional cues in audio files. Firstly, the Chroma feature detects pitch energy, revealing pitch variations that convey emotional nuances. Secondly, Spectral Contrast highlights fluctuations in loudness or intensity, enabling us to discern how specific frequencies prominently emerge within the audio spectrum. Thirdly, tonnetz captures the tonal relationships in speech, reflecting various aspects of emotional expressions. Fourthly, the MEL Spectrogram Frequency maps energy distribution across frequency bands, offering insights into the spectral composition of the audio. Lastly, the MFCC feature furnishes a comprehensive overview of the spectral envelope's shape, encapsulating the audio's tone colour and personality. While MFCC demonstrates a propensity for yielding commendable accuracy, we aim to explore other possible combinations of the features to unlock further avenues for refining emotion recognition capabilities.

Notably, certain features exhibit variations between male and female speakers, hinting at potential gender-specific speech patterns. For instance, males tend to have lower pitch frequencies than females, as shown by the Chroma feature. Tonal inflections differ between genders, as seen in tonnetz. Spectral contrast may indicate differences in loudness or intensity, reflecting speech dynamics. Further analysis of MEL Spectrogram Frequency and MFCC could reveal more about gender-specific spectral composition and tone colour. This preliminary investigation sets the stage for deeper analysis, enriching our understanding of speech emotion recognition across genders.

4.1.3. Data Augmentation

To facilitate the generation of models that are robust and generalisable, the team took several steps to augment the audio files:

Adding noise

The audio files were obtained by recording voice actors in a professional setting, meaning that they followed a fixed structure for each of the sentences that they spoke without any environmental influence to the audio captured. By adding random noise to each audio file, we are able to mimic the existence of background noise making the model more robust.

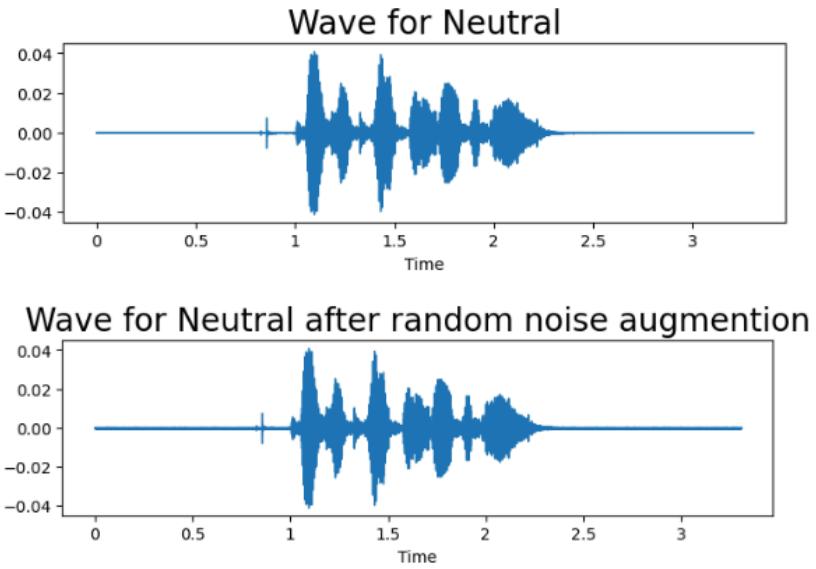


Image 4: Audio Sound Wave

Shifting audio

From the EDA, we noticed that the audio files followed a structure of having a one second pause at the start and end of the recording. This rigid structure of each sentence may be due to the professional recording environment which we do not want our models to focus on. Therefore, we introduced random shifts to the audio to either make it start earlier or later at random.

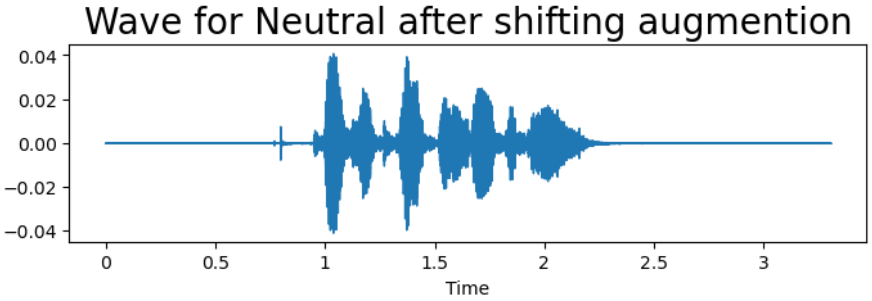


Image 5: Audio Sound Wave

Stretching audio

By stretching each audio file, we can mimic the sentence being spoken at a faster or slower rate. We realise that this method of augmentation could change the underlying meaning of each sentence which was why we fine tuned the parameter to minimise how the amount of stretch affects the audio files.

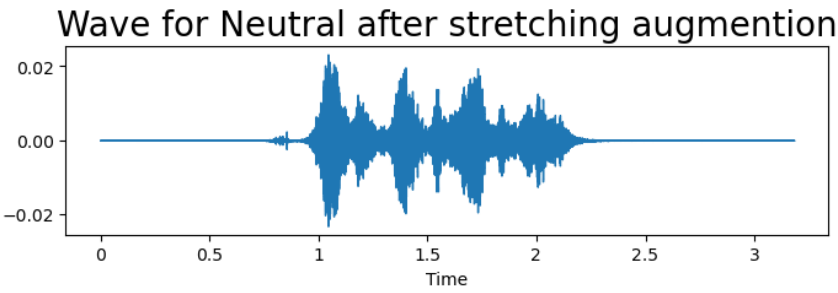


Image 5: Audio Sound Wave

4.2. Train/Test Split

The table below explains the terminologies that will be used throughout this section.

Table 1

Terminologies used for Train/Test split

Terminology	**Description**
X_train, X_test, y_train, y_test	Imbalanced, unencoded and unnormalized data split into 80% and 20% for training and testing respectively

In the splitting of our data into training and testing sets, we further split the training data following the same 80/20 split to create a validation set which would be used for validation of models and parameter finetuning.

From the EDA, we have also identified the labels that would be used as the target variable:

Table 2

Attribute descriptions with class encoding labels

<table>
<tr><th>Attribute Name</th><th>Data Type</th><th>Remarks</th></tr>
<tr><td>Emotion</td><td>Numerical</td><td>Attribute that we encoded based on labelling given from the dataset

Emotion labels according to:

<table>
<tr><th>Emotion</th><th>Mapped value of emotion</th></tr>
<tr><td>female_angry</td><td>0</td></tr>
<tr><td>female_calm</td><td>1</td></tr>
<tr><td>female_disgust</td><td>2</td></tr>
<tr><td>female_fear</td><td>3</td></tr>
<tr><td>female_happy</td><td>4</td></tr>
<tr><td>female_neutral</td><td>5</td></tr>
<tr><td>female_sad</td><td>6</td></tr>
<tr><td>female_surprise</td><td>7</td></tr>
<tr><td>male_angry</td><td>8</td></tr>
<tr><td>male_calm</td><td>9</td></tr>
<tr><td>male_disgust</td><td>10</td></tr>
<tr><td>male_fear</td><td>11</td></tr>
<tr><td>male_happy</td><td>12</td></tr>
<tr><td>male_neutral</td><td>13</td></tr>
</table>
</td></tr>
</table>

		male_sad	14
		male_surprise	15

4.3. Resampling Techniques

We explored various resampling techniques to address class imbalance in the dataset, which is a common challenge in machine learning. Imbalanced datasets are a serious problem that needs to be tackled because not doing so leads to poor generalisation of minority class and a bias towards the majority classes (Paul, 2018).

Altogether, we explored two different methods: Synthetic Minority Oversampling Technique (SMOTE) and Random Over-Sampling. SMOTE is a sophisticated approach that aims to balance class distribution by generating synthetic data for the minority class. Synthetic points are then created along the line segments connecting the chosen point to its neighbours. This method introduces diversity to the minority class and captures the underlying structure of the data (Satpathy, 2023). It is important to note that SMOTE can be applied to various machine learning models, including KNN, CNN and RNN.

Another resampling technique we utilised is Random Over-Sampling, which involves replicating randomly selected instances from the minority class. Unlike SMOTE, Random Over-Sampling does not consider the relationships between instances and simply duplicates existing minority class instances (Brownlee, 2021). While this approach is simpler and quicker to implement compared to SMOTE, it may result in duplicated instances that do not contribute much diversity to the minority class.

By leveraging these resampling techniques, we aim to identify which resampling technique will create a balanced dataset that facilitates the training of machine learning models capable of accurately classifying emotions in speech data, ultimately enhancing the performance and robustness of our speech emotion recognition system.

4.4. Machine Learning Models

4.4.1. k-Nearest Neighbours (KNN)

In our literature review presented in section 3.4, we identified the KNN classifier as a superior technique due to its comparative speed in computation against other models. This finding aligns with our research goals of selecting the most effective model while prioritising computational efficiency.

Additionally, KNN effectively accommodates non-linear relationships found in speech data features, such as pitch, through a distance metric to identify similar instances. This approach enables it to adeptly process the subtle variations in speech that convey different emotions. Consequently, we have chosen to explore the KNN model for this project.

We go through a 4-step process before finalising the KNN model.

After splitting the data into training and testing sets, we employ StandardScaler from the sklearn.preprocessing package. This step is crucial to prevent skewed outcomes since KNN heavily relies on distance. Scaling features to a common scale ensures that each feature contributes equally to the distance calculations, leading to a more precise and fair evaluation of data points.

Second, we explore the 2 different resampling techniques, namely, SMOTE and Random Over-Sampling on our training data. This step is essential to ensure that the neighbourhood does not get dominated by the majority classes, leading to inaccurate predictions for minority class samples.

Third, we tune the KNN model to find the optimal number of neighbours (K) that yields the highest accuracy for our speech emotion recognition task, using a layered evaluation process. We iterate over a range of K values, using 10-fold cross-validation (CV) for each K to assess the model's performance. Empirical evidence has shown that a good value of k-fold starts with 10 (Browniee, 2023). We first choose the best K based on its CV score. Second, we use a separate validation set to reevaluate the top-performing K value from the CV process. Here we will shortlist 4 K values based on its validation score. Last, we test the shortlisted K values on a separate test set and choose the optimal K value that yields the highest test accuracy.

Fourth, we will determine the final distance metric concurrently with the third step. Both Euclidean and Manhattan distances allow for easy interpretation of results, making them a solid foundation for initial experiments and benchmarks in speech emotion recognition. They are also more computationally efficient compared to more complex distance metrics like Dynamic Time Warping.

Euclidean distance metric:

$$dL2(p, q) = \sqrt{\sum_{k=1}^{n} (pk - qk)^2}$$

Manhattan distance metric:

$$dL1(p,q) = \sum_{k=1}^{n} |pk - qk|$$

Where n is the number of attributes, and pk and qk are the k^{th} attributes of data objects p and q, respectively. p and q are the two data to be searched for the distance.

Although it was proven by Asgar (Asgar et al., 2023) that the accuracy achieved with the Euclidean distance metric surpasses the Manhattan distance, this may be different for our model. Hence we will test both distance metrics for our KNN model.

Finally, we define the final KNN classifier using KNeighborsClassifier which is part of the scikit-learn package. Our 2 key parameters involved identifying the optimal number of neighbours in the third step and determining the best distance metric that yields the best accuracy in step four. We also need to identify the resampling technique that yields the best performance for our dataset.

4.4.2. Convolutional Neural Network (CNN)

Despite CNN models being traditionally associated with Image Processing tasks, CNN models are also extremely beneficial for Speech Emotion Recognition tasks because the model has demonstrated the capability to identify intricate patterns in speech data associated with diverse emotions and audio data with noise, yielding notably accurate prediction results (Onah & Ibrahim, 2023). As such, we chose to explore the use of the CNN model for our project.

Convolutional Layer - Conv1D

The Conv1D layer is a convolutional layer that operates on sequential data with one spatial dimension and it is often used for time-series and text data, while the Conv2D layer operates on 2D input data and it is often used in image classification (Hag, 2021). For our project, we used the Conv1D layer as we were handling time-series datasets.

A Conv1D layer does not have filters that stride through a two-dimensional matrix horizontally and vertically like how the Conv2D layer does. Instead, the filter only strides through one dimension horizontally.

How the Conv1D layer works is as such - if our kernel size is set to 3 and stride to 1, this means that the filter will stride 1 step horizontally to the right through 3 attributes (Refer to **Image 6 & 7**) until it completes striding through all attributes.

MFCC0	MFCC1	MFCC2	MFCC3
-274.081064	90.553742	-1.319409	28.355147
-282.730691	71.540738	34.301680	11.759998
-391.468786	86.299907	44.825176	37.441880
-471.930760	63.462643	13.026071	7.454551
-325.833235	65.936750	17.097764	19.950584

Image 6: First stride

MFCC0	MFCC1	MFCC2	MFCC3
-274.081064	90.553742	-1.319409	28.355147
-282.730691	71.540738	34.301680	11.759998
-391.468786	86.299907	44.825176	37.441880
-471.930760	63.462643	13.026071	7.454551
-325.833235	65.936750	17.097764	19.950584

Image 7: Second stride

After every convolution operation, the activation function, ReLu, is applied to the feature maps of each convolution operation, replacing all negative values with zeros and leaving positive values unchanged. This process introduces non-linearity into the neural network, allowing the network to learn complex patterns and relationships in the data.

Max Pooling Layer

While max pooling is widely used in CNN models to downsample the feature maps produced after every convolution operation, we did not implement max pooling layers into our model structure to prevent information loss.

Since max pooling reduces the dimensionality of the feature maps by retaining only the maximum value within each pooling region, this would mean that we will be discarding other information that might be crucial for our analysis, which is what we do not want to be doing considering the nature of our project.

Regularisation Technique - Early Stopping

As we compile the model for training, we set the number of epochs to 100. To prevent the model from training for too long and overfitting the model to the training data, we implemented the regularisation technique, Early Stopping (Refer to **Image 8**), to help us with that.

```
# Regularisation - Define early stopping callback to prevent overfitting
early_stopping = callbacks.EarlyStopping(monitor='val_loss', patience=5, verbose=1)

epochs = 200

history = model.fit(X_ros, y_ros, epochs=epochs, validation_split=0.2, callbacks=[early_stopping])
```

Image 8: Applying Regularisation (i.e., Early Stopping)

By including Early Stopping as a callback during the model training, we would have effectively regularised the training process by stopping it when the validation loss stops improving after 5 epochs (i.e., patience=5). This will help to achieve better generalisation performance on unseen data.

Hyperparameter tuning

Despite conducting initial experiments and training different CNN models *(which will be discussed in the "Results and Discussions" section)*, we realised that we were not achieving the most optimal results that we were looking for.

Understandably, creating the CNN model that will produce the best results requires multiple rounds of trial-and-error. Therefore, hyperparameter tuning is essential for CNN models as it helps to optimise the various parameters that significantly impact the model's performance and generalisation ability.

```
tuner = kt.RandomSearch(model_builder,
                        objective='val_accuracy',
                        max_trials=100,  # Adjust based on computational resources
                        executions_per_trial=1,  # Adjust based on computational resources
                        directory='model_hyperparameter_tuning_final',
                        project_name='audio_classification')

tuner.search(X_ros_train, y_ros_train, epochs=100, validation_split=0.2, callbacks=[tf.keras.callbacks.EarlyStopping('val_loss', patience=5)])

best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]
# Create the model with the best hyperparameters
model = model_builder(best_hps)
```

Image 9: Hyperparameter Tuning

Using the Keras Tuner (kt.), the tuner is initialised with "RandomSearch", which explores different hyperparameter combinations. The function, tuner.search(), is called with training data (X_ros_train and y_ros_train), along with other parameters like epochs, validation_split, and early stopping callbacks. The Keras Tuner iteratively builds and trains models with different hyperparameters, aiming to maximise validation accuracy (Refer to **Image 9**).

After tuning, the best set of hyperparameters is retrieved using the function, tuner.get_best_hyperparameters(). The model is then rebuilt using these best hyperparameters to create the best model to achieve the best performance.

4.4.3. Recurrent Neural Network (RNN)

We decided to use RNN as it is a commonly used machine learning model used in speech recognition. We made use of the Keras library to build a RNN model and specifically, we made use of the Long Short-Term Memory (LSTM) layer which is a variant of RNN layer. The LSTM layer is useful in our case as it helps solve one of the problems faced by traditional RNN, which is the vanishing or exploding gradient problem. The training input for this model is a data frame, which contains the extracted audio features values.

Long Short-Term Memory Layer

LSTM was designed to resolve the vanishing or exploding gradient due to long term dependencies. Normal RNN suffers when they need to remember information from much earlier time steps. This is due to the very small gradient update when the gradient is located very far back in the time step. When this occurs, layers with the small gradient update stop learning.

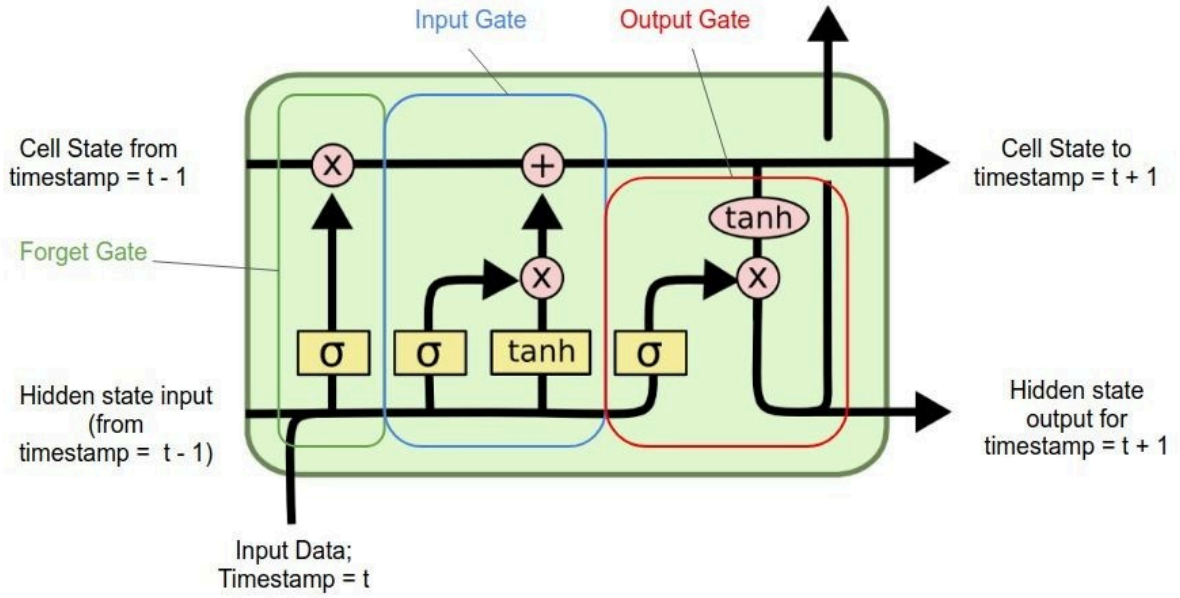


Image 10: LSTM cell architecture. J., R. T. J. (2024, March 6). *LSTMs explained: A complete, technically accurate, conceptual guide with keras*. Medium. https://medium.com/analytics-vidhya/lstms-explained-a-complete-technically-accurate-conceptual-guide-with-keras-2a650327e8f2

LSTM has 3 gates, forget gate, input gate and output gate. These gates determine which information is important and which are irrelevant. This ensures only important information gets passed down the sequence to improve performance of the model. Besides the gates, LSTM has an additional component compared to traditional RNN, called cell states, which serves as an additional memory but it connects to every cell in the sequence and brings the information down the sequence. This component is crucial as it helps bring information from earlier time steps to a much later time step. An LSTM cell still has the hidden state, similar to

a traditional RNN cell. Hidden state is a memory which stores the information from the previous time step.

The addition and removal of information to the cell state is determined by the 3 gates.

Firstly, information from hidden state and current output will go through the forget gate. The information will be passed into a sigmoid activation function to determine which information to drop.

Next, The input gate takes in the information from the hidden state and current input. The combined information will pass through 2 activation functions, tanh and sigmoid, before being multiplied together. The output, also known as "candidate" cell state, will determine what information to retain in the cell state. The "candidate" cell state is then multiplied with a forget vector where the LSTM will determine what information to drop before updating the cell state to create a new cell state.

Lastly, the hidden state determines what information the next hidden state will hold. The current input along with information in the hidden state will go through a sigmoid activation function to produce an output. This output is then multiplied with the output after passing the information in the new cell state through a tanh activation function. The output will be the next hidden state to the next sequence. In addition, the information in the cell state will be passed down to the next time step (Phi, 2020).

Gated Recurrent Unit Layer

Aside from the LSTM layer, we also looked into Gated Recurrent Unit (GRU) layers in our RNN model which just like LSTM aims to solve the problem of having vanishing gradients. Unlike LSTM which makes use of three gates: Forget, Input, and Output gates, GRU makes use of only two gates: Reset gate and Update gate. In its simplest form, these two gates are vectors that help the model decide what information is needed for the output and what should be discarded.

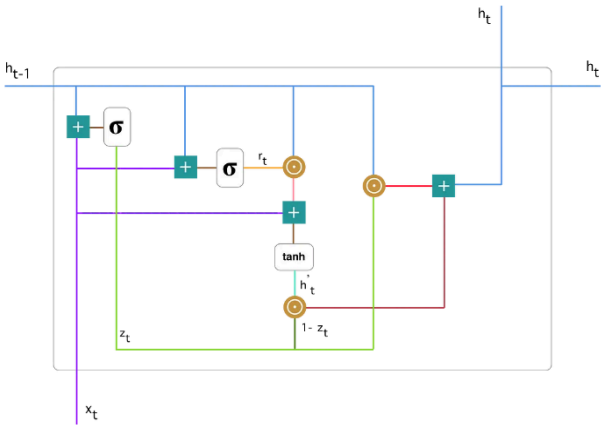


Image 11: Structure of Gated Recurrent Unit. Kostadinov, S. (2019, November 10). *Understanding GRU networks*. Medium. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

The image above shows the basic GRU which contains the “plus” operation, a Sigmoid and Tanh activation function and a “Hadamard product” operation. The Hadamard product operation helps to do element-wise multiplication and division whenever necessary for the matrices that are passing through the unit.

For each timestep, the update gate takes in its own weight, and multiplies it with the input. The output from the previous step is also multiplied by the weight of the current unit and both results are added together to be passed through a sigmoid function. By taking both the information from the current timestep and the previous timestep, the update gate is able to help the model determine how much of the past information is needed to be retained and used for future steps.

The reset gate is the opposite of the update gate. Instead of deciding what past information to retain, the reset gate helps the model decide what information should be forgotten. The formula of the reset gate is the same as the update gate with the only difference being the weights that would be used for each multiplication.

After passing through both the update and reset gates, we would do the element-wise product between the reset gate’s output and the output from the previous timestep. This is essentially the model deciding what needs to be removed from the previously stored information.

The final step of the GRU is to determine the information that will be passed on to the next unit. To do this, we bring back the output from all the gates that were previously used including the output of the update gate, reset gate and perform the element-wise multiplication to both of the outputs before summing up the results and passing it on to the next unit.

4.5. Feature Selection Analysis

Selection of data is important for any machine learning task. As we are doing audio classification, we wanted to identify which subsets of features would be the best for our different models. In order to carry out the feature selection analysis, we decided to use Recurrent Neural Network to perform the analysis. Hyperparameter tuning was done to identify the optimal hyperparameters. The hyperparameters we are testing are: Batch Size and Optimizer Algorithm.

We also wanted to test if normalising the data would help in the model performance, non-normalised or normalised data. For the test, we decided on 2 types of normalisation, which is min-max scaling and z-score.

The constant hyperparameter here would be the epoch which was set at 100.

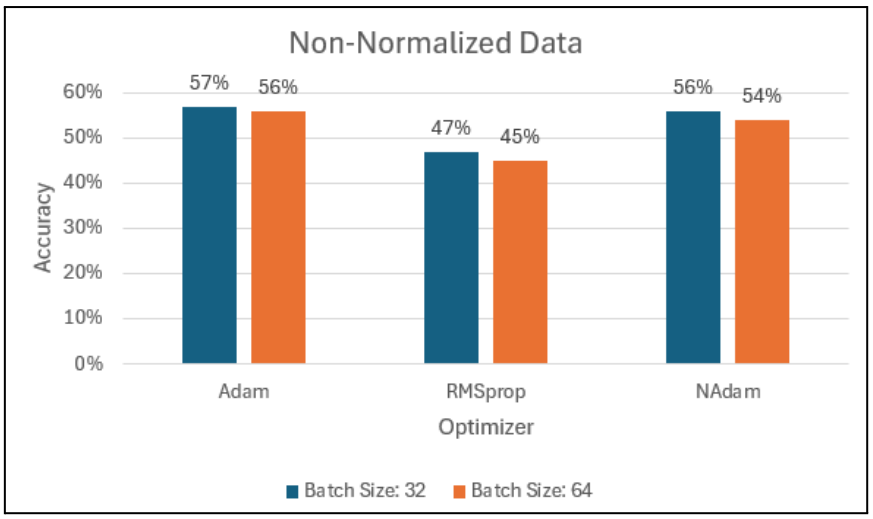


Image 12: Experiment with no normalisation

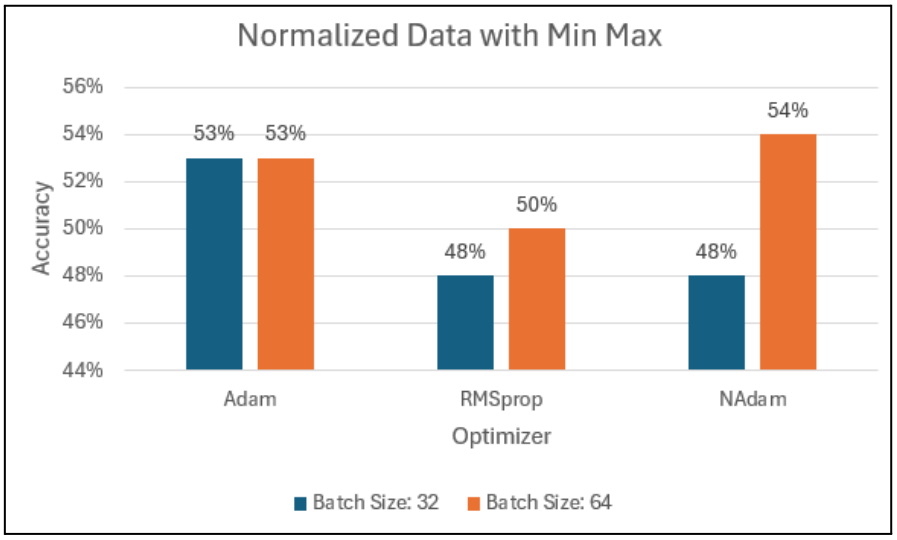


Image 13: Experiment with Min-Max Normalisation

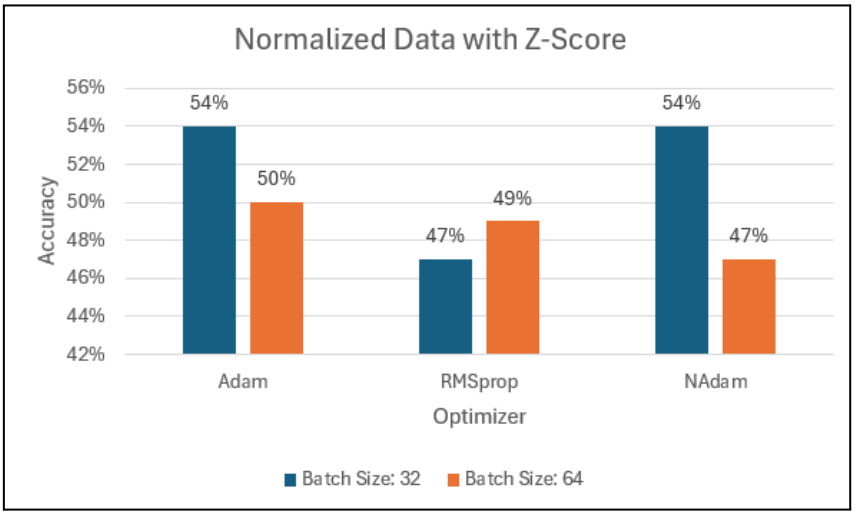


Image 14: Experiment with Z-Score Normalisation

Based on the results obtained, batch size 32 with Adam optimizer produced the best results. In addition, normalising the data resulted in a slight drop in performance. In conclusion, we decided to use batch size 32, Adam optimizer and non-normalised data to conduct the feature selection analysis.

Different feature subsets were created from the 5 audio features we have extracted, mainly: Mel-Frequency Cepstrum (MFCC), Spectral Contrast, Chroma, Mel Spectrogram (MEL) and Tonnetz. 31 subsets were created with subset size ranging from 1 to 5.

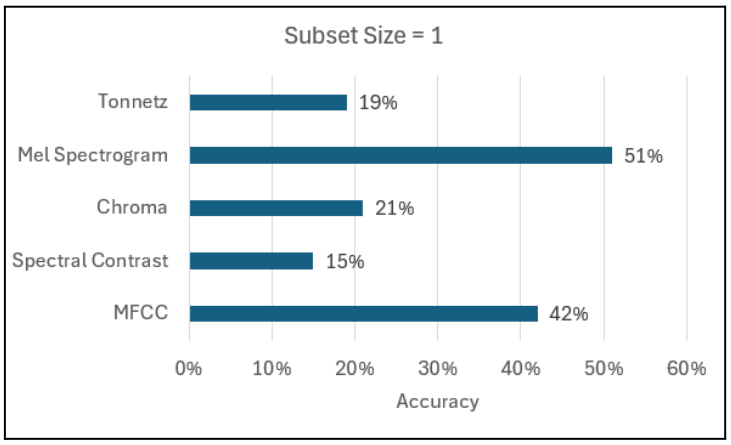


Image 15: Subset Size of 1

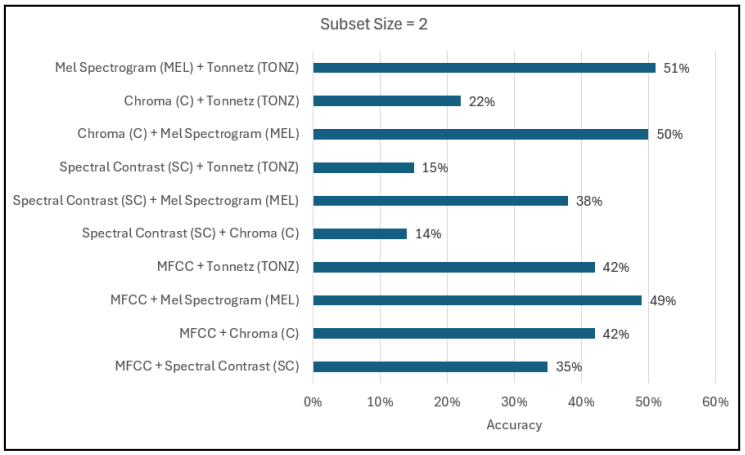


Image 16: Subset Size of 2

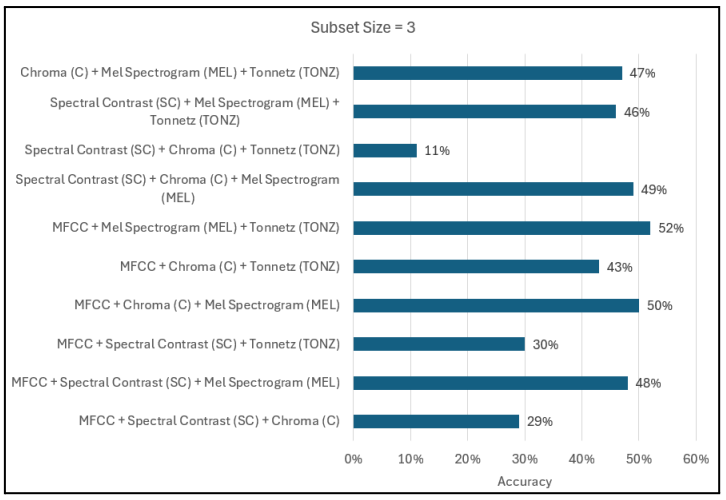


Image 17: Subset Size of 3

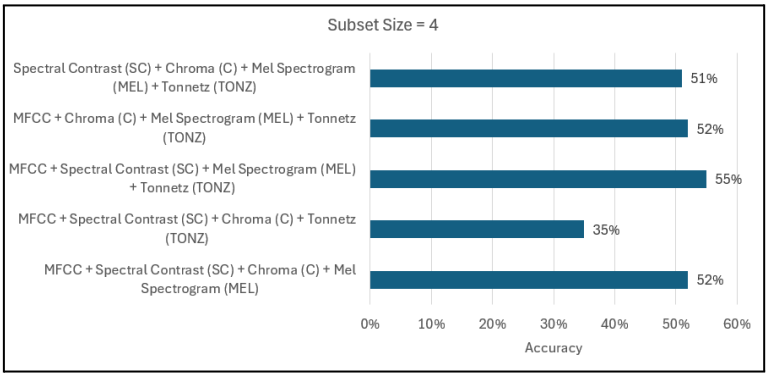


Image 18: Subset Size of 4

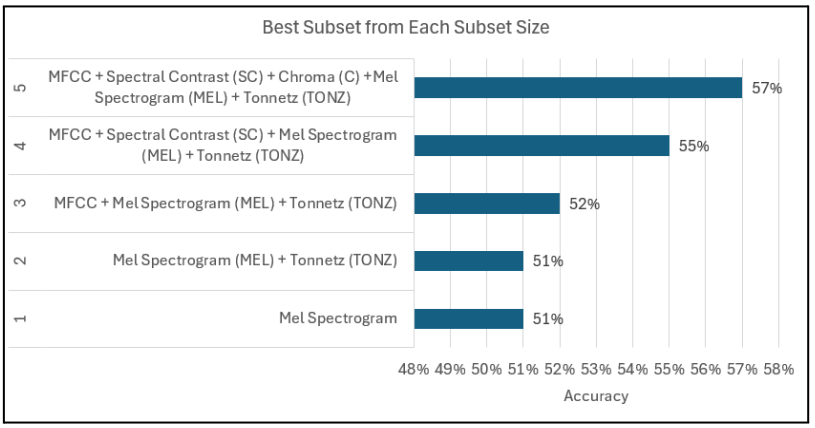


Image 19: Subset accuracy analysis

Image 19 shows the accuracy of the best subset from each subset size and based on the analysis, using all 5 features allows us to achieve 57% accuracy, the highest among all the other feature subsets.

Our findings contradict one of the literature reviews which states that solely using MFCC would result in higher accuracy. However, we can't completely prove the theory wrong as the authors used different models and datasets for their analysis.

Moving forward, we will utilise all 5 features for the 3 models for training.

5. Results and Discussions

5.1. Evaluation metrics

We look into a variety of metrics to evaluate our speech emotion recognition model. Our evaluation metrics include accuracy, precision, recall and f1-score.

We also look into the confusion matrix to provide a detailed visualisation of the model's performance across all emotion classes, showing True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). This granularity allows for an in-depth analysis of the model's behaviour, and an understanding of the model's strengths and weaknesses in classifying different emotions.

Accuracy is the simplest and most intuitive performance measure. It calculates the proportion of correct predictions (both true positives and true negatives) out of all predictions. However, accuracy alone may not fully capture our model's efficacy, because we have an imbalanced dataset and with that, certain emotions are underrepresented or overrepresented.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

Precision measures the accuracy of the positive predictions. It is the ratio of true positives to all positive predictions. In speech emotion recognition, precision would indicate how many of the instances classified as a particular emotion are that emotion.

$$Precision = \frac{(TP)}{(TP + FP)}$$

Recall is measured as the ratio of correctly predicted instances to the total actual instances of that emotion in the dataset. It evaluates the model's ability to capture and correctly classify all relevant cases of each emotion.

$$Recall = \frac{(TP)}{(TP + FN)}$$

The f1-score is a harmonic average that is derived from precision and recall. It helps maximise the model's recall without sacrificing precision.

$$F1 = \frac{2*Precision*Recall}{Precision + Recall}$$

5.2. Project Phases

Throughout the project, we have 2 phases. In the first phase, we wanted to test if the non-augmented or augmented data would be better for our models. For each model, we used a single variant to train on the 2 datasets and evaluate their performance. The main metric used to evaluate is accuracy. After evaluation, the accuracy of all 3 models ranges between 58% to 74%.

We noticed that the validation accuracy on non-augmented and augmented dataset had more to be desired, which could hint to our models being overfitted or that there wasn't enough data to train on. To circumvent this, we decided to combine both datasets into 1 and found out

that the merged dataset improved the accuracy of all 3 models. From this, we finalised the dataset that we would use and moved on to phase 2, where we created different variants of each model to find the best-performing variant.

In phase 2, the main focus was to identify the optimal model for each model. Hence, different variants were created for CNN and RNN models, with 2 and 4 variants respectively. In addition, fine tuning is also done for all 3 models. The results for the different phases and different variants will be discussed below.

Here, we found out that using Random Oversampling (ROS) works better for all models. Hence, ROS was used as the sampling method to balance the training data for model training. ROS was done after the train test split in order to avoid data leakage whereby test data might appear in the training dataset, affecting the evaluation when we use the test data. We want to ensure the model evaluates the actual data, instead of synthetic or duplicated data, providing a more accurate performance measure metrics.

5.3. k-Nearest Neighbours (kNN)

The performance of the kNN model was evaluated through 2 phases and 3 sets of experiments. In phase 1, we test the performance of our non-augmented and augmented dataset through 2 experiments. Our first experiment focused on determining the ideal number of neighbours. We utilised a 10-fold cross-validation technique and explored two distinct distance metrics alongside 2 resampling methods, all applied to our non-augmented dataset. In the second experiment, we extended our analysis to our augmented dataset, comparing the validation accuracy produced by different distance metrics and resampling techniques and also how it compares to our non-augmented data.

After completing phase 1, we advanced to phase 2 to evaluate our model's performance on the merged dataset. In this phase, we carried out an experiment to determine which distance metrics, number of neighbours and resampling techniques significantly improve the validation accuracy. We also aim to assess whether our merged dataset boosts the performance of our kNN model compared to both our augmented and non-augmented datasets.

The 2 phases are designed to identify the best number of neighbours, distance metric, and resampling technique for maximising kNN model performance across different datasets in speech emotion recognition.

Phase 1:

Results from Experiment 1:

From **Table 3** below, based on the 10-fold cross validation score, we observed that our model consistently identified 1 as the optimal number of neighbours, maintaining this selection across both distance metrics and resampling techniques applied. We note that the combination of using Manhattan distance, applying SMOTE for resampling and setting the number of neighbours to 1 yield the highest validation accuracy, which stands at 52.65%.

Table 3

Comparison of resampling technique and distance metric using 10-fold CV to find the optimal number of neighbours, utilising the non-augmented dataset

Dataset	Resampling Technique	K-fold CV	Distance Metric	Number of Neighbours	Cross - Validation Accuracy	Validation Accuracy
Non-augmented data	SMOTE	10	Euclidean	**1**	0.7660	**0.5088**
	Random Over-Sampling			**1**	0.7486	**0.5067**
Non-augmented data	SMOTE	10	Manhattan	**1**	0.7748	**0.5265**
	Random Over-Sampling			**1**	0.7575	**0.5201**

However, from **Image 20** below, we observed that for lower K values (eg. 1), training accuracy is high while the validation accuracy remains low due to the overfitting of the training data. We aim to find the optimal K value by identifying the sweet spot on the graph using a separate validation set. The sweet spot exists where the gap between the training and validation scores is minimal while ensuring both scores are as high as possible. Hence, we further explore the training and validation accuracy for K values between 1 and 10 inclusive.

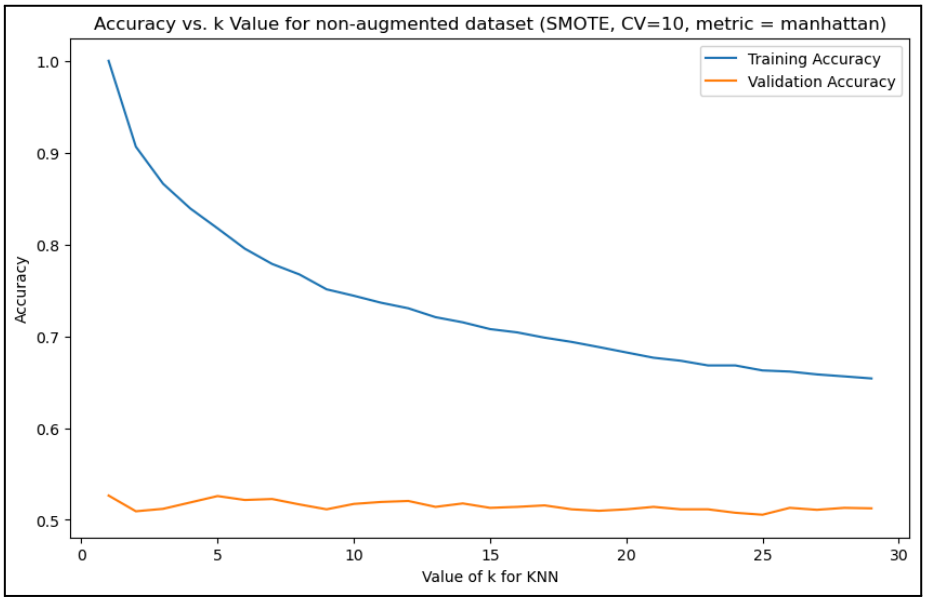


Image 20. Graph of Accuracy vs K value for our non-augmented dataset using SMOTE, CV = 10, metric = Manhattan

From **Image 21** below, we noticed that the top validation accuracies are when K = 1, K = 5, K = 7, and K = 10. Given the tendency that K = 1 leads to overfitting the model, we will

explore the 3 other K values as our ideal number of neighbours for our non-augmented dataset.

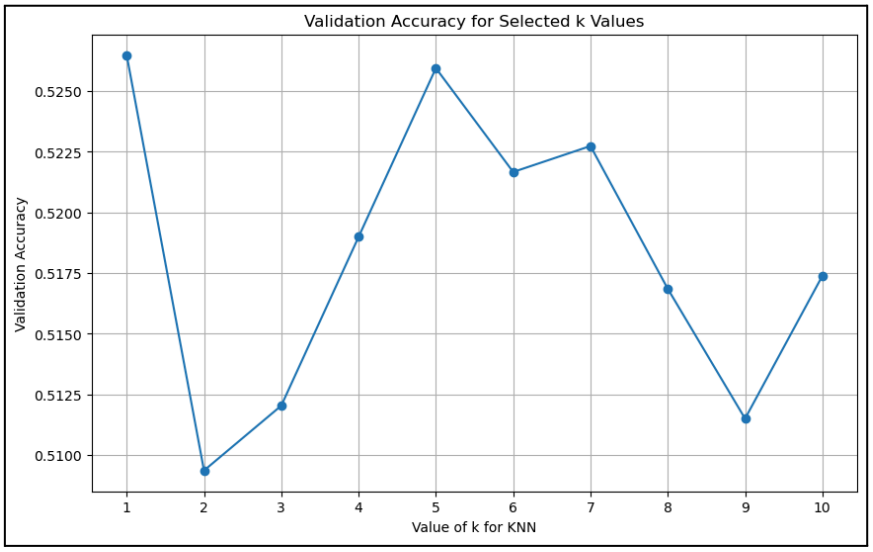


Image 21. Graph of validation accuracy for selected K value for our non-augmented dataset using SMOTE, CV = 10, metric = Manhattan

From **Table 4** below, we can conclude that the optimal number of neighbours is 7 for our non-augmented data as it yields the highest test accuracy of 51.52%.

Table 4

Comparison of the number of neighbours, utilising the non-augmented dataset

Dataset	Resampling Technique	Distance Metric	Number of Neighbours	Validation Accuracy	Test Accuracy
Non-augmented data	SMOTE	Manhattan	**1**	0.5265	**0.5143**
			5	0.5259	**0.5053**
			7	0.5227	**0.5152**
			10	0.5174	**0.5131**

Results from Experiment 2:

From **Table 5** below, based on the 10-fold cross validation score, we also observed that our model consistently identified 1 as the optimal number of neighbours, maintaining this selection across both distance metrics and resampling techniques applied. Similar to experiment 1, we note that the combination of using Manhattan distance, applying SMOTE for resampling and setting the number of neighbours to 1 yields the highest validation accuracy, which stands at 47.46%.

Table 5

Comparison of resampling technique and distance metric using 10-fold CV to find the optimal number of neighbours, utilising the augmented dataset

Dataset	Resampling Technique	K-fold CV	Distance Metric	Number of Neighbours	Cross - Validation Accuracy	Validation Accuracy
Augmented data	SMOTE	10	Euclidean	**1**	0.7492	**0.4564**
	Random Over-Sampling			**1**	0.7335	**0.4532**
Augmented data	SMOTE	10	Manhattan	**1**	0.7551	**0.4746**
	Random Over-Sampling			**1**	0.7396	**0.4741**

Likewise, as seen from **Image 22** below, setting the number of neighbours as 1 leads to overfitting. Hence, we will explore the training and validation accuracy for K values between 1 and 10 inclusive using a separate validation set to find the sweet spot.

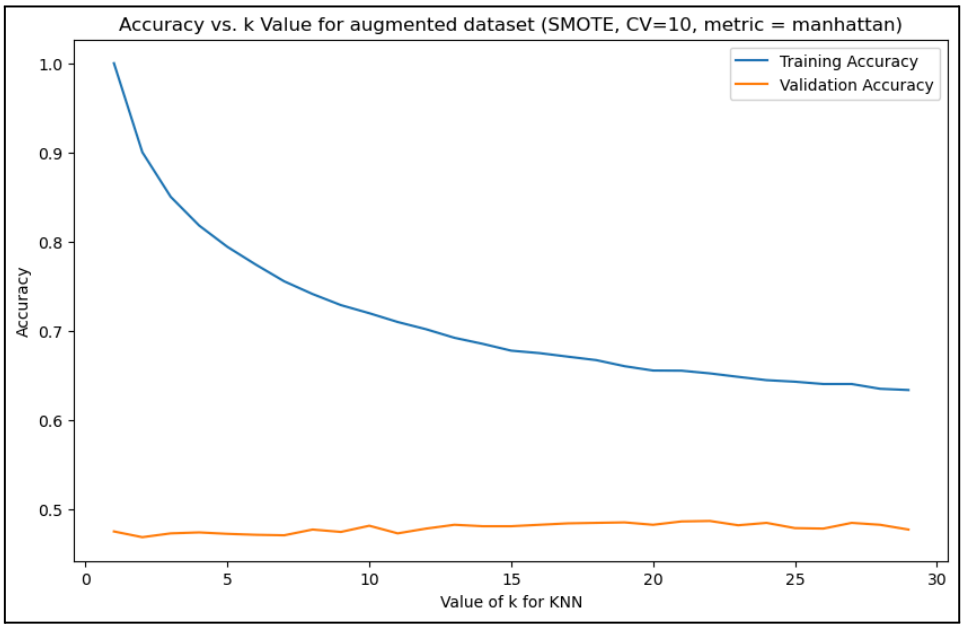


Image 22. Graph of Accuracy vs K value for our augmented dataset using SMOTE, CV = 10, metric = Manhattan

From **Image 23** below, we noticed that the top validation accuracies are when K = 1, K = 8, K = 9, and K = 10. Given the tendency that K = 1 leads to overfitting the model, we will explore the 3 other K values as our ideal number of neighbours for our augmented dataset.

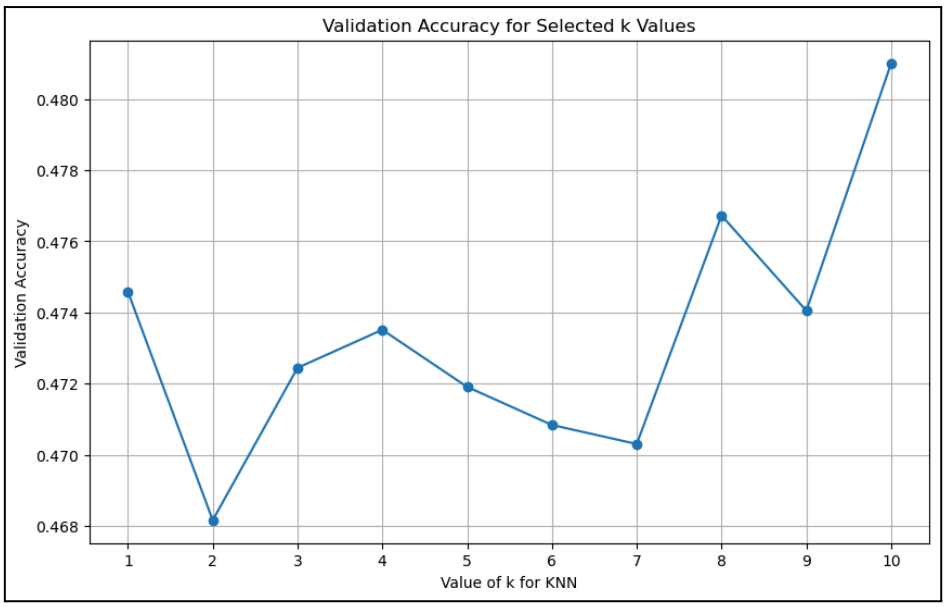


Image 23. Graph of validation accuracy for selected K value for our augmented dataset using SMOTE, CV = 10, metric = Manhattan

From **Table 6** below, we can conclude that the optimal number of neighbours is 9 for our augmented data as it yields the highest test accuracy of 48.18%.

Table 6

Comparison of the number of neighbours, utilising the augmented dataset

Dataset	Resampling Technique	Distance Metric	Number of Neighbours	Validation Accuracy	Test Accuracy
Augmented data	SMOTE	Manhattan	**1**	0.4746	**0.4733**
			8	0.4767	**0.4758**
			9	0.4741	**0.4818**
			10	0.4810	**0.4771**

Phase 2:

Results from Experiment 3:

Similarly, from **Table 7** below, based on the 10-fold cross validation score, we observed that our model consistently identified 1 as the optimal number of neighbours, maintaining this selection across both distance metrics and resampling techniques applied. We note that the combination of using Manhattan distance, applying ROS for resampling and setting the number of neighbours to 1 yields the highest validation accuracy, which stands at 53.30%.

Table 7

Comparison of resampling technique and distance metric using 10-fold CV to find the optimal number of neighbours, utilising the merged dataset

Dataset	Resampling Technique	K-fold CV	Distance Metric	Number of Neighbours	Cross - Validation Accuracy	Validation Accuracy
Merged data	SMOTE	10	Euclidean	**1**	0.7769	**0.5173**
	Random Over-Sampling			**1**	0.7576	**0.5154**
Merged data	SMOTE	10	Manhattan	**1**	0.7829	**0.5282**
	Random Over-Sampling			**1**	0.7667	**0.5330**

As seen from **Image 24** below, setting the number of neighbours as 1 leads to overfitting. Hence, we will explore the training and validation accuracy for K values between 1 and 10 inclusive using a separate validation set to find the sweet spot.

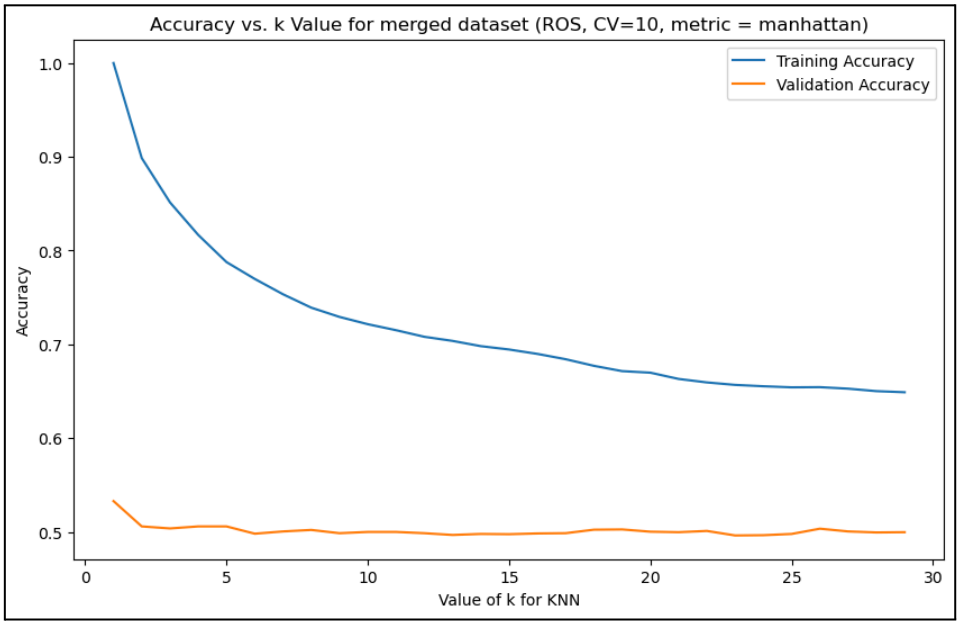


Image 24. Graph of Accuracy vs K value for our merged dataset using ROS, CV = 10, metric = Manhattan

From **Image 25** below, we noticed that the top validation accuracies are when K = 1, K = 2, K = 4, and K = 5. Given the tendency that K = 1 leads to overfitting the model, we will explore the 3 other K values as our ideal number of neighbours for our augmented dataset.

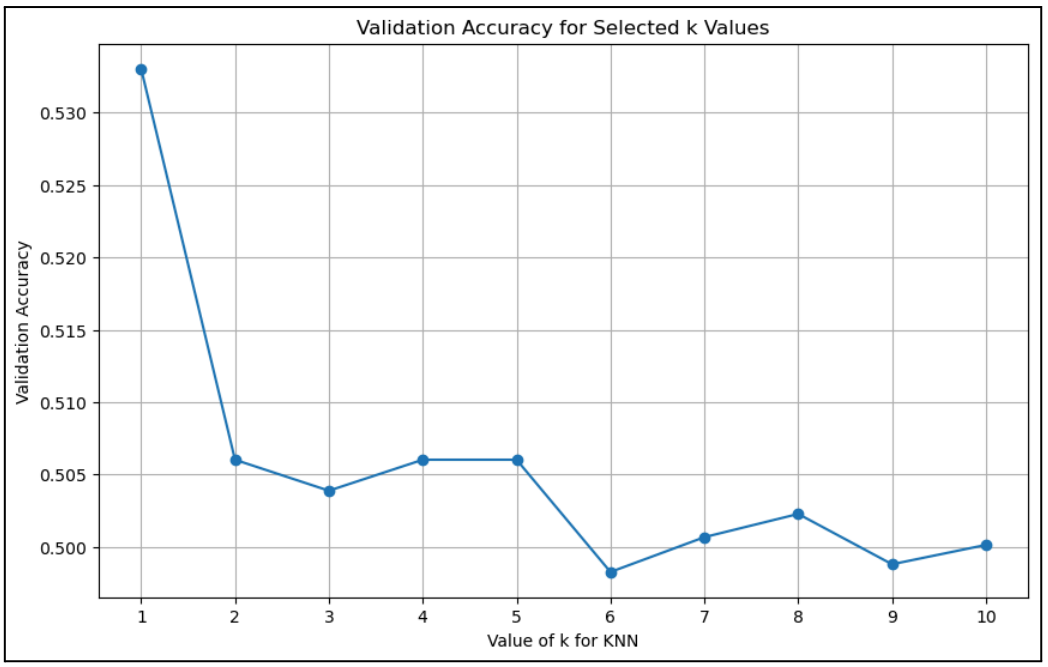


Image 25. Graph of validation accuracy for selected K value for our merged dataset using ROS, CV = 10, metric = Manhattan

From **Table 8** below, we can conclude that the optimal number of neighbours is still 1 for our merged data as it yields the highest test accuracy of 53.03%.

Table 8

Comparison of the number of neighbours, utilising the merged dataset

Dataset	Resampling Technique	Distance Metric	Number of Neighbours	Validation Accuracy	Test Accuracy
Merged data	Random Over-Sampling	Manhattan	**1**	0.5330	**0.5303**
			2	0.5060	**0.5136**
			4	0.5060	**0.5112**
			5	0.5060	**0.5166**

Summary for KNN model:

Upon evaluating the performance across all three experiments and all three datasets as seen from **Table 9** below, it was observed that the merged dataset outperformed the others, achieving the highest validation and test accuracies, 53.30% and 53.03% respectively. The optimal parameter settings for the merged dataset included using a single neighbour (K = 1), employing Manhattan distance as the metric, and utilising random over-sampling as the resampling technique.

Table 9

Comparison of all 3 datasets on validation and test accuracy

Dataset	Resampling Technique	Distance Metric	Number of Neighbours	Validation Accuracy	Test Accuracy
Non-augmented data	SMOTE	Manhattan	7	0.5227	0.5152
Augmented data	SMOTE	Manhattan	9	0.4741	0.4818
Merged data	Random Over-Sampling	Manhattan	1	0.5330	0.5303

Our minimal goal is for the KNN model to accurately learn and identify angry, calm, and sad emotions. This capability is crucial for effectively classifying these emotional states in the context of the customer service sector. The confusion matrix and classification report in **Image 27** and **Image 28** below depict the highest misclassification rates observed for male_sad at 62.33%, followed by male_angry at 46.97%, female_sad at 43.46%, female_calm at 33.33%, female_angry at 27.29% and male_calm at 17.65%. Despite these challenges, the model performed relatively better in identifying 'calm' emotions, particularly for male_calm. However, female_calm still had a higher misclassification rate, suggesting room for improvement.

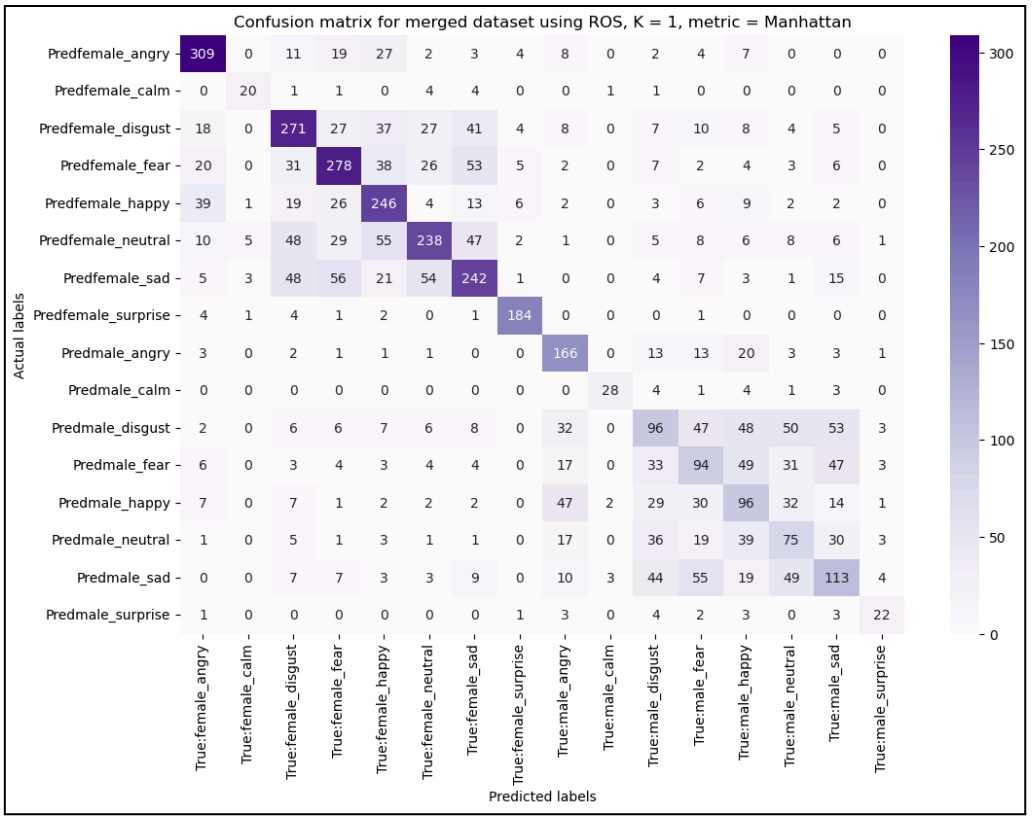


Image 27. Confusion matrix for the 16 emotion classes for the merged dataset (ROS, Number of neighbours = 1, metric = Manhattan)

Classification report on merged dataset on KNN model:

	precision	recall	f1-score	support
female_angry	0.78	0.73	0.75	425
female_calm	0.62	0.67	0.65	30
female_disgust	0.58	0.59	0.58	463
female_fear	0.59	0.61	0.60	457
female_happy	0.65	0.55	0.60	445
female_neutral	0.51	0.64	0.57	372
female_sad	0.53	0.57	0.55	428
female_surprise	0.93	0.89	0.91	207
male_angry	0.73	0.53	0.61	313
male_calm	0.68	0.82	0.75	34
male_disgust	0.26	0.33	0.29	288
male_fear	0.32	0.31	0.31	299
male_happy	0.35	0.30	0.33	315
male_neutral	0.32	0.29	0.31	259
male_sad	0.35	0.38	0.36	300
male_surprise	0.56	0.58	0.57	38
accuracy			0.53	4673
macro avg	0.55	0.55	0.55	4673
weighted avg	0.54	0.53	0.53	4673

Image 28. Classification report for the 16 emotion classes on the merged dataset (ROS, Number of neighbours = 1, metric = Manhattan)

Image 29 below visualises the clustering of 6 distinct emotion classes in a 2D t-distributed Stochastic Neighbour Embedding (t-SNE) reduced space. t-SNE is beneficial in this scenario as it excels at reducing high-dimensional data into two dimensions while preserving local structure. **Image 29** supports the previously mentioned misclassification rates, illustrating that male_calm emotions (represented by purple) form tighter clusters, while the male_sad, and male_angry classes (represented by brown and lime green) display a wider dispersion. This spatial distribution likely contributes to the higher misclassification rates.

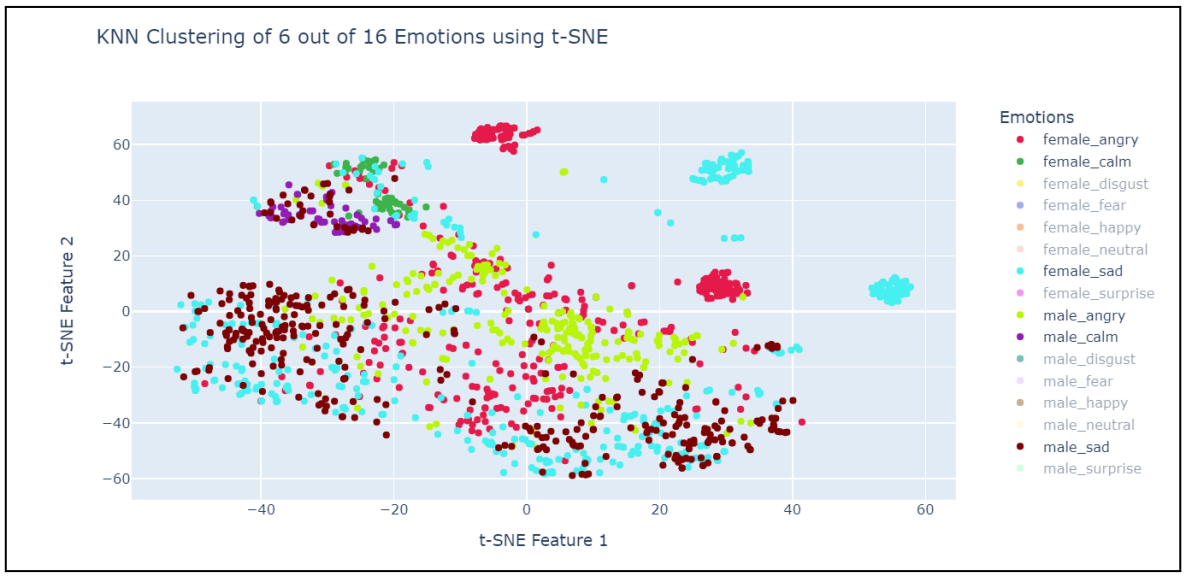


Image 29. 2D graph of KNN clusters for female_angry, female_calm, female_sad, male_angry, male_calm, male_sad

Image 30 reveals that among the six emotion classes we're concentrating on, the merged dataset achieves the highest precision rate for four of them (namely, female_angry, female_calm, male_angry and male_calm). The precision rate for female_sad and male_sad are the lowest among the six classes (52.61% and 34.66% respectively).

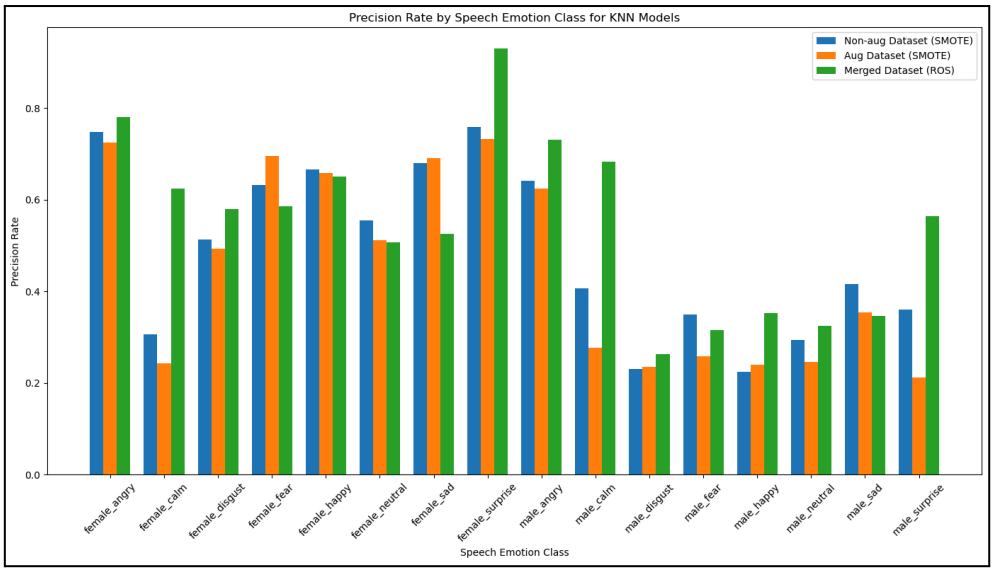


Image 30. Precision rate of the 16 emotion classes for the 3 datasets

Image 31 below reveals similar results to the precision scores above. The merged dataset achieved the highest f1 scores for the same four classes (namely, female_angry, female_calm, male_angry and male_calm). The f1 scores for female_sad and male_sad are the lowest among the six classes (54.50% and 36.10% respectively).

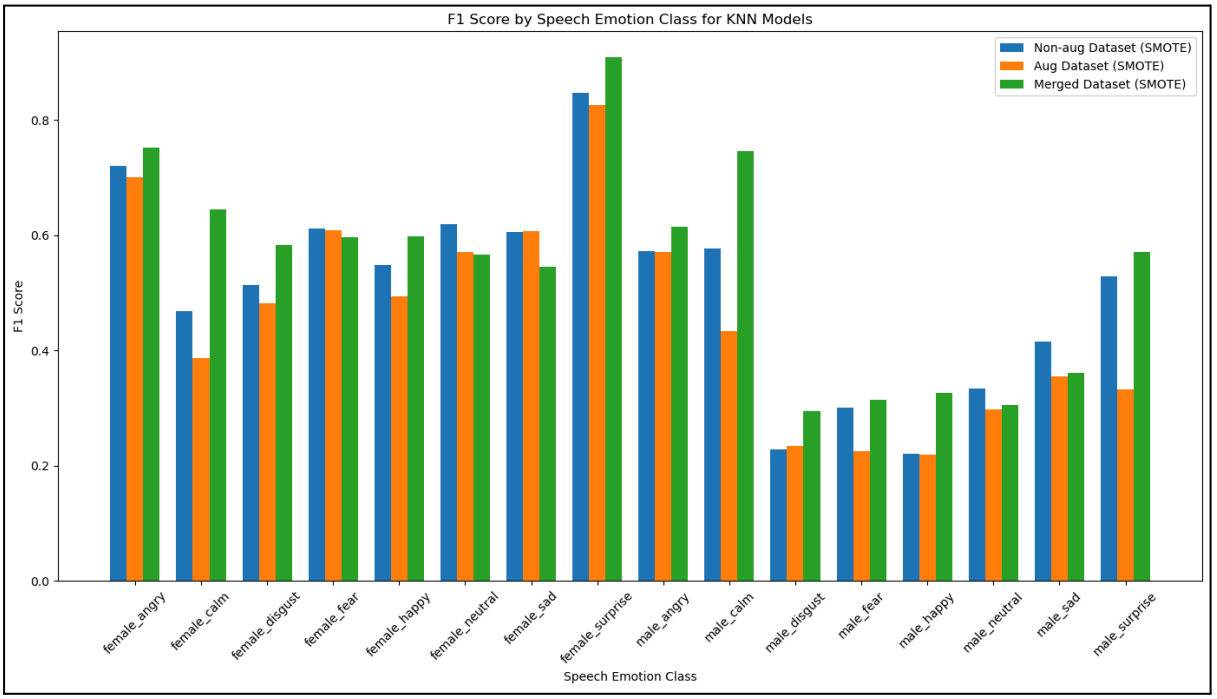


Image 31. F1 Score of the 16 emotion classes for the 3 datasets

Ultimately, although the KNN model using the merged dataset outperforms the other two datasets, it does not uniformly excel across all six targeted emotion classes, notably in cases like female_sad and male_sad. The overall test accuracy is also not deemed satisfactory. Furthermore, KNN does not work well when there are too many features because of the curse of dimensionality. This could be the main reason why our model constantly selects 1 as the best number of neighbours (using CV). To potentially surpass these constraints, exploring neural networks, as discussed below, could offer more valuable insights into enhancing model effectiveness across these varied emotion classifications.

5.4. Convolutional Neural Network (CNN)

The performance of the Convolutional Neural Network (CNN) model was evaluated through two phases. The first phase was to find the optimal dataset augmentation and the second phase was to find the optimal configurations for the CNN model. Ultimately, these two phases aim to enhance the CNN model performance.

Phase 1: Dataset Optimization

Our focus in this phase is to understand the impact of data augmentation techniques on model performance and determine the most effective dataset for our CNN model. We have conducted a comparative analysis using three datasets, non-augmented dataset, augmented dataset, and the merged dataset

Experiment 1: Non-Augmented Dataset

For Experiment 1, we used a 10-layered CNN model to evaluate the performance of the model on the original dataset without any form of augmentation. **Image 32** shows the training history metrics.

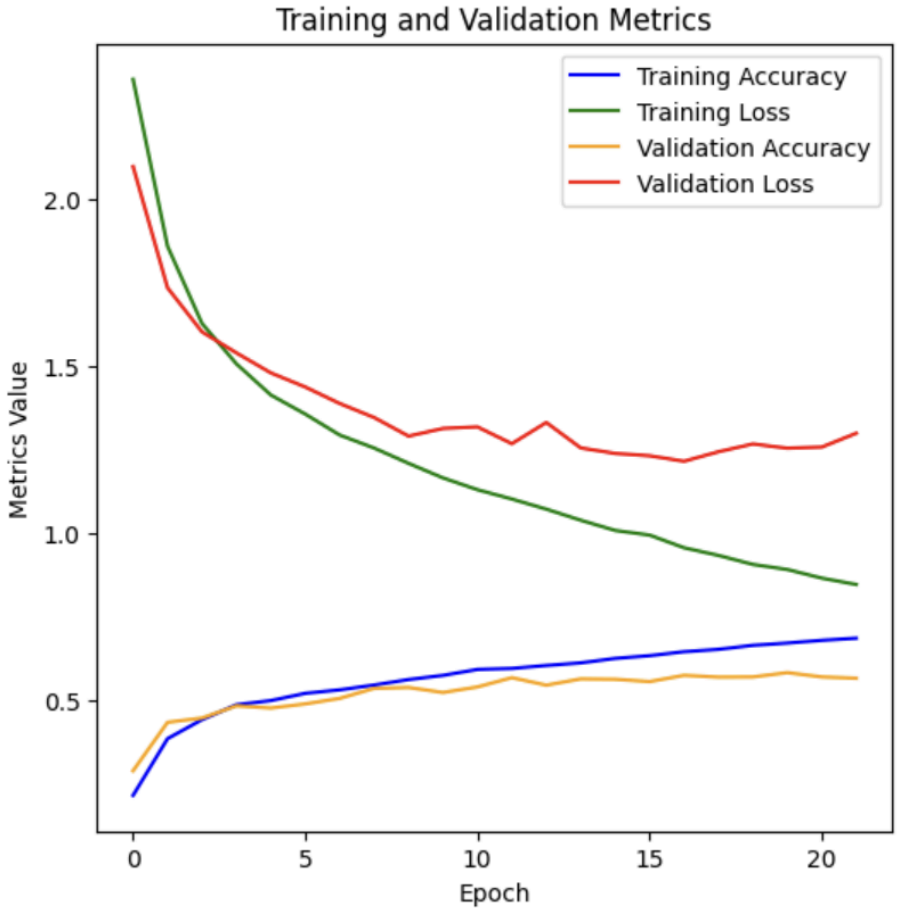


Image 32: A summarised training history of a 10-layered CNN model using a non-augmented dataset.

Experiment 2: Augmented Dataset

For Experiment 2, we used the same 10-layered CNN model to test the performance of the model on the augmented dataset. The augmented audio dataset was created by introducing variations such as adding noise, shifting audio timing, and stretching audio time to the original audio dataset. **Image 33** below shows the training history metrics.

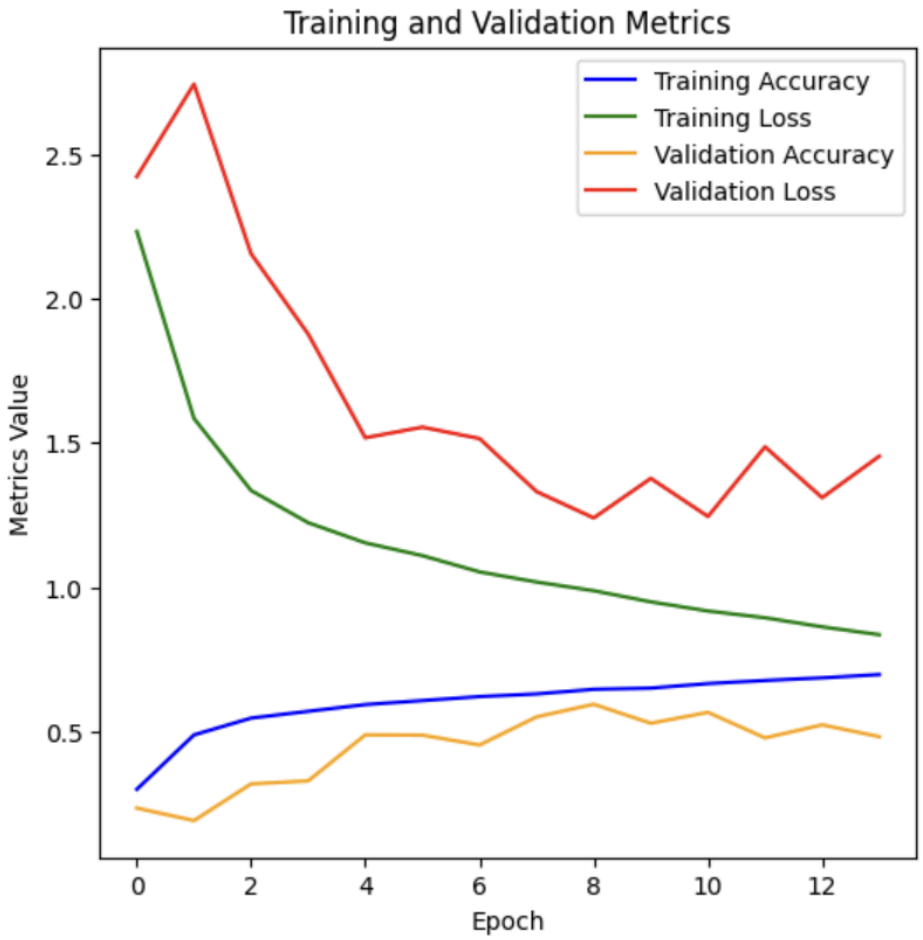


Image 33: A summarised history of a 10-layered CNN model using an augmented dataset.

Experiment 3: Merged Dataset

For Experiment 3, we used the same 10-layered CNN model to test the performance of the model on the merged dataset. The merged dataset was created by combining the non-augmented and the augmented datasets from Experiment 1 and Experiment 2 respectively. **Image 34** below shows the training history metrics.

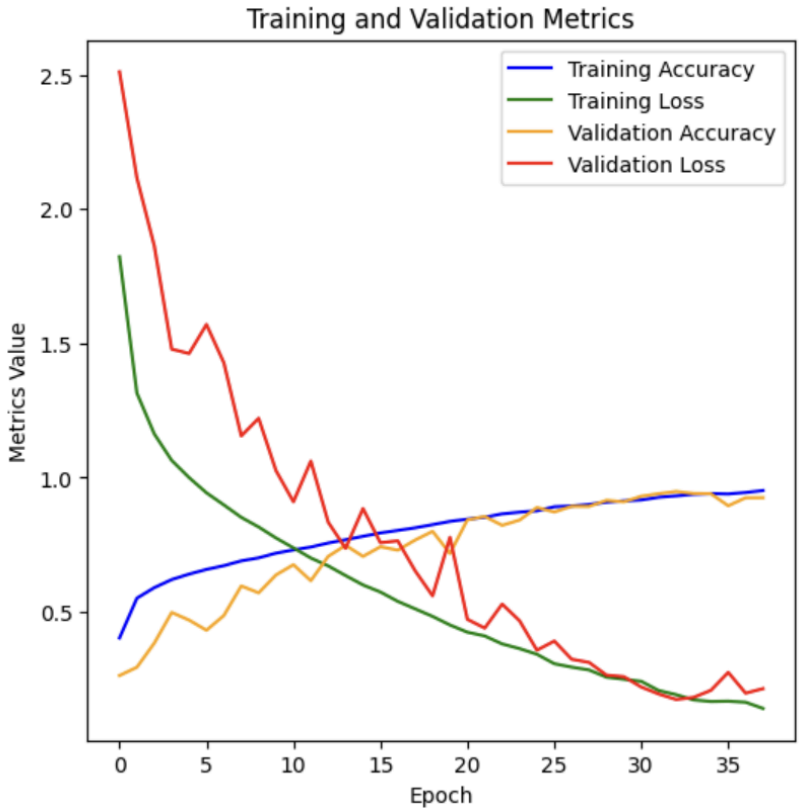


Image 34: A summarised history of a 10-layered CNN model using a merged dataset.

Table 10 shows the training results of the CNN model across the different datasets:

Table 10

Training results of the CNN model across non-augmented, augmented and merged datasets

	Validation Accuracy	Validation Loss
Non-Augmented Dataset	56.61%	1.2997
Augmented Dataset	48.22%	1.4534
Merged Dataset	92.53%	0.2145

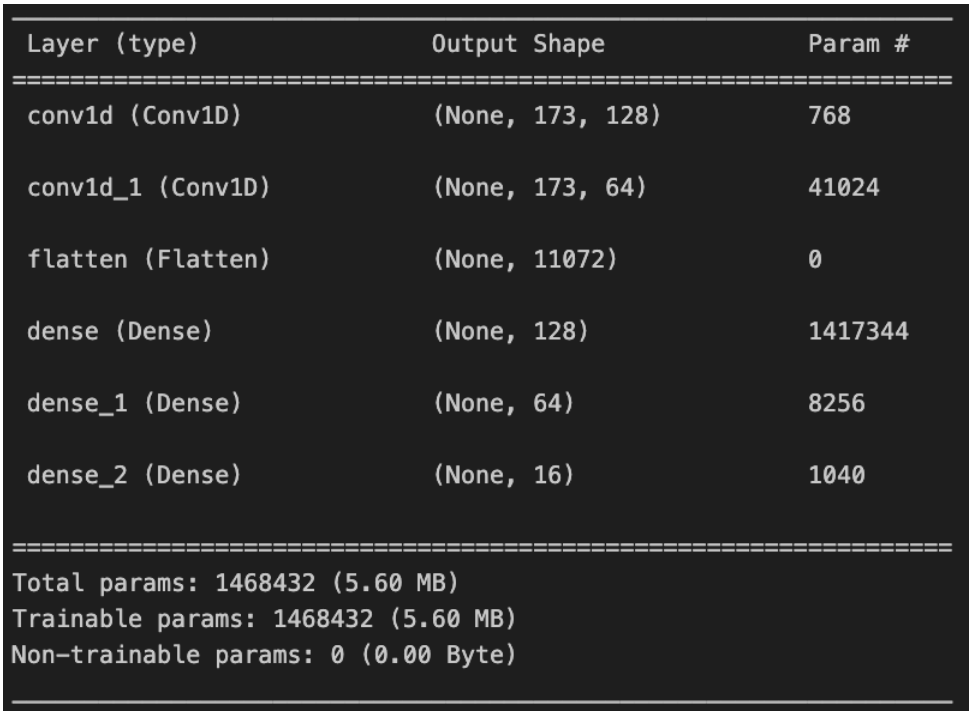
Based on the training results of the CNN model across different datasets from **Table 10,** the merged dataset outperforms the non-augmented and the augmented datasets. Training on the merged dataset yields the highest validation accuracy of 92.53% and the lowest validation loss of 0.2145. As such, we should proceed to enhance the model's performance using the merged dataset for hyperparameter tuning and adjustments to the model's architecture.

Phase 2: Model Variant Exploration

Beyond dataset optimization, we want to explore how the CNN architecture can affect the model performance. In this phase, we have experimented with different numbers of convolutional layers and hyperparameters. Ultimately, we want to identify the optimal CNN model configurations that improve the model's performance. We are using the merged dataset in this phase.

Experiment 1: Lesser Number of Convolutional Layers

We started the model variant exploration by using a simple CNN model with a few convolutional layers. **Image 35** shows the model's architecture that we experimented with. It is a 6-layered CNN model that consists of 2 Conv1D layers, 1 Flatten layer and 3 Dense layers.



Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 173, 128)	768
conv1d_1 (Conv1D)	(None, 173, 64)	41024
flatten (Flatten)	(None, 11072)	0
dense (Dense)	(None, 128)	1417344
dense_1 (Dense)	(None, 64)	8256
dense_2 (Dense)	(None, 16)	1040

Total params: 1468432 (5.60 MB)
Trainable params: 1468432 (5.60 MB)
Non-trainable params: 0 (0.00 Byte)

Image 35: A model summary of a CNN model with a lesser number of convolutional layers

Image 36 below shows the training history metrics of the CNN model with a lesser number of convolutional layers. It highlights the point of convergence for the training validation loss which allows us to identify when to stop the training of the model to prevent overfitting.

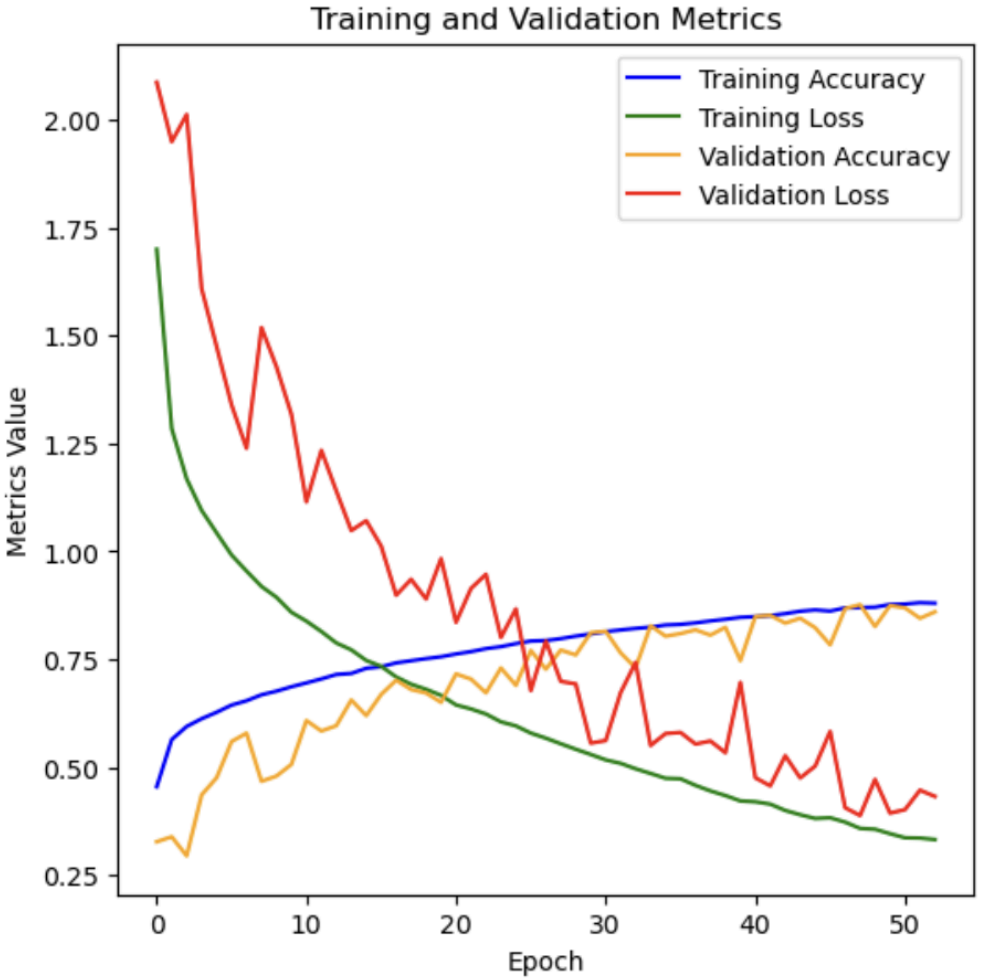


Image 36: A Summarised history of 6-layered CNN model

Experiment 2: Higher Number of Convolutional Layers

For this experiment, we increased the number of convolutional layers of the CNN model to see whether it will improve the model's performance. **Image 37** shows the model's architecture that we experimented with. It is a 10-layered CNN model that consists of 4 Conv1D layers, 1 Flatten layer and 5 Dense layers.

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 173, 256)	1,536
conv1d_1 (Conv1D)	(None, 173, 128)	163,968
conv1d_2 (Conv1D)	(None, 173, 64)	41,024
conv1d_3 (Conv1D)	(None, 173, 32)	10,272
flatten (Flatten)	(None, 5536)	0
dense (Dense)	(None, 256)	1,417,472
dense_1 (Dense)	(None, 128)	32,896
dense_2 (Dense)	(None, 64)	8,256
dense_3 (Dense)	(None, 32)	2,080
dense_4 (Dense)	(None, 16)	528

Total params: 1,678,032 (6.40 MB)
Trainable params: 1,678,032 (6.40 MB)
Non-trainable params: 0 (0.00 B)

Image 37: A model summary of a CNN model with a higher number of convolutional layers

Image 38 below shows the training history metrics of the CNN model with a higher number of convolutional layers. It highlights the point of convergence for the training validation loss which allows us to identify when to stop the training of the model to prevent overfitting.

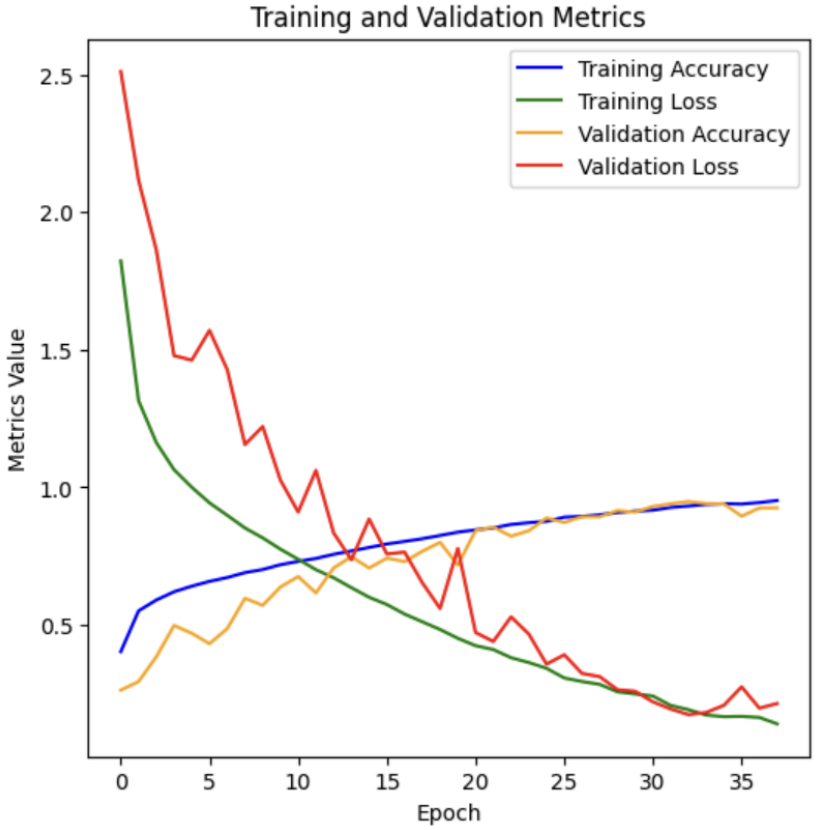


Image 38: A summarised history of the 10-layered CNN model

Table 11 shows the training results of the CNN models with different number of convolutional layers:

Table 11

Training results of the CNN models, 6-layered CNN vs. 10-layered CNN

	Validation Accuracy	Validation Loss
6-layered CNN	86.00%	0.4328
10-layered CNN	92.53%	0.2145

Based on Table 11, it is evident that the 10-layered CNN with more convolutional layers performs better than the 6-layered CNN with fewer convolutional layers. The 10-layered CNN model gives a higher validation accuracy of 92.53% and a lower validation loss of 0.2145. As such, Experiment 1 and Experiment 2 highlight that increasing the number of convolutional layers does improve the CNN model performance.

Experiment 3: Convolutional Layers (Hyperparameters Tuning)

Building on the foundational knowledge acquired from the initial experiments in Phase 2, the final experiment, Experiment 3, focuses on hyperparameters tuning within the convolutional layers of the 10-layered CNN model. Specifically, we are curious whether the number of neurons at each layer can improve the CNN model performance.

We undergo a random search of 100 iterations of varying the number of neurons in the convolutional layers. **Image 39** shows the best hyperparameters identified after the 100 iterations. It is a 10-layered CNN model that consists of 4 Conv1D layers, 1 Flatten layer and 4 Dense layers.

Layer (type)	Output Shape	Param #
conv1d_4 (Conv1D)	(None, 173, 288)	1728
conv1d_5 (Conv1D)	(None, 173, 192)	276672
conv1d_6 (Conv1D)	(None, 173, 96)	92256
conv1d_7 (Conv1D)	(None, 173, 32)	15392
flatten_1 (Flatten)	(None, 5536)	0
dense_5 (Dense)	(None, 480)	2657760
dense_6 (Dense)	(None, 128)	61568
dense_7 (Dense)	(None, 128)	16512
dense_8 (Dense)	(None, 32)	4128
dense_9 (Dense)	(None, 16)	528

Total params: 3126544 (11.93 MB)
Trainable params: 3126544 (11.93 MB)
Non-trainable params: 0 (0.00 Byte)

Image 39: Model Summary of the best CNN model hyperparameters identified after the 100 iterations

Image 40 below shows the training history metrics of the CNN model with a higher number of convolutional layers. It highlights the point of convergence for the training loss which allows us to identify when to stop the training of the model to prevent overfitting.

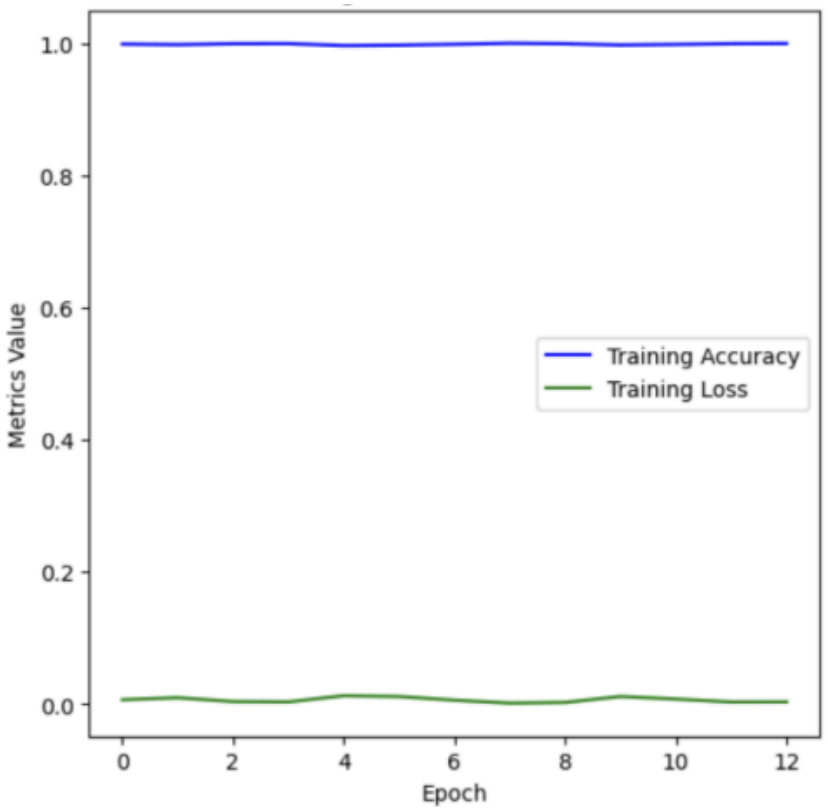


Image 40: A summarised history of the 10-layered CNN model after hyperparameters tuning

Table 12 shows the training results of the CNN models before hyperparameters tuning and after hyperparameter tuning:

Table 12

Training results of the CNN models, Before Hyperparameters Tuning vs. After Hyperparameters Tuning

	Validation Accuracy	Validation Loss
Before Hyperparameters Tuning	92.53%	0.2145
After Hyperparameters Tuning	99.75%	0.0074

Based on Table 12, it is evident that the 10-layered CNN after hyperparameters tuning is better than before hyperparameters tuning. The 10-layered CNN model after hyperparameters tuning exhibits a higher validation accuracy of 99.75% and a lower validation loss of 0.0074.

As such, Experiment 3 highlights that hyperparameter tuning can improve model performance.

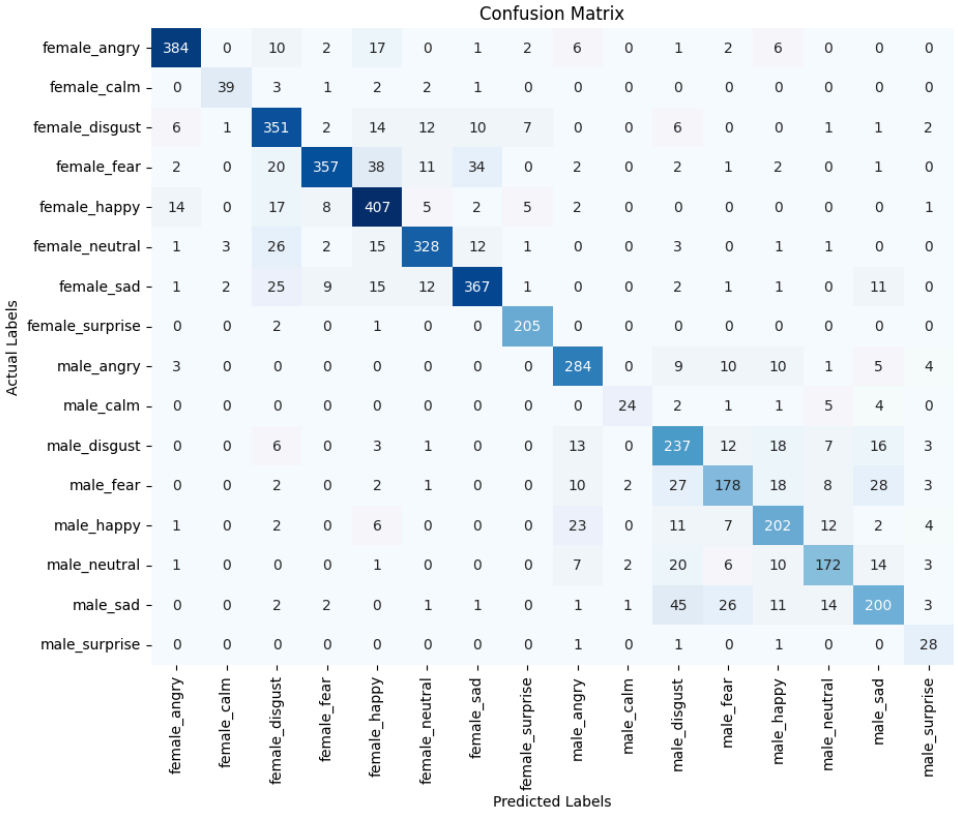


Image 41: Confusion matrix for 10-layered CNN after hyperparameters tuning prediction on test data

We then use the 10-layered CNN after hyperparameters tuning to test on the testing dataset to evaluate the model performance. Based on **Image 41**, the 10-layered CNN after hyperparameters tuning gives an overall good prediction, where a clear darker shade of diagonal line from the top left to the bottom right can be observed from the confusion matrix. This is a good sign because these values represent the true positives.

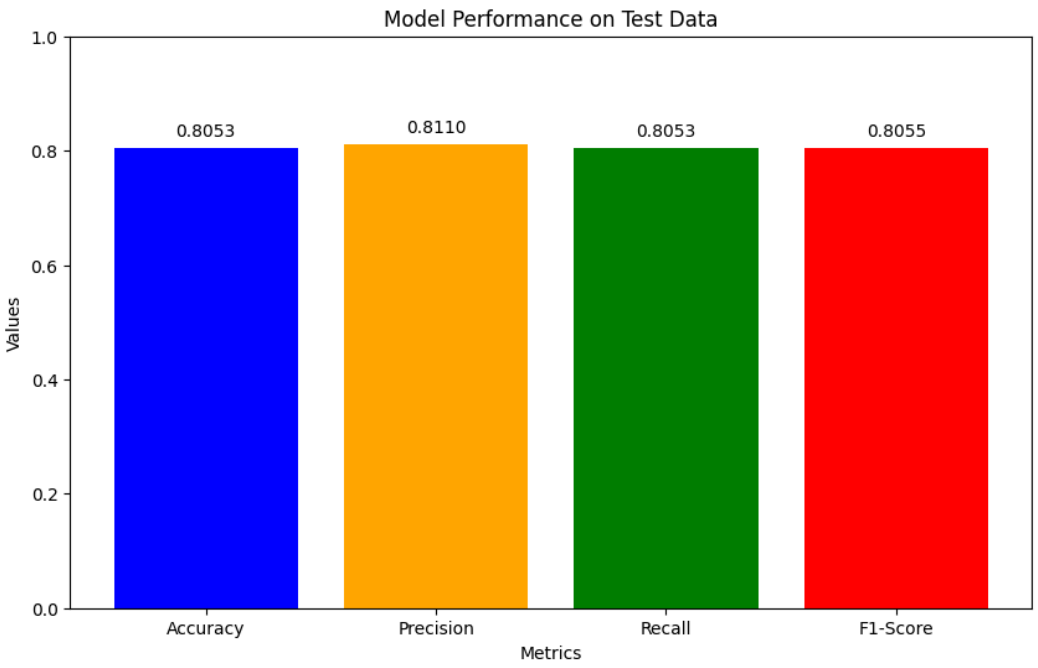


Image 42: Model Performance Metrics for 10-layered CNN after hyperparameters tuning prediction on test data

The model performance of the 10-layered CNN after hyperparameters tuning on the testing dataset is also commendable. The model has achieved high performance metrics, with accuracy, precision, recall, and F1-score all sitting in the upper 80% range.

5.5. Recurrent Neural Network (RNN)

When coming up with the RNN models, we broke up our work into two phases. In the first phase, we tested our model on the non-augmented and augmented dataset to find out which dataset works best for our model.

Phase 1: Dataset Optimization

Experiment 1: Non-Augmented dataset

For experiment 1, we used a 4-layered RNN model to test the performance of the model on the non-augmented dataset. The image below shows the training history metrics.

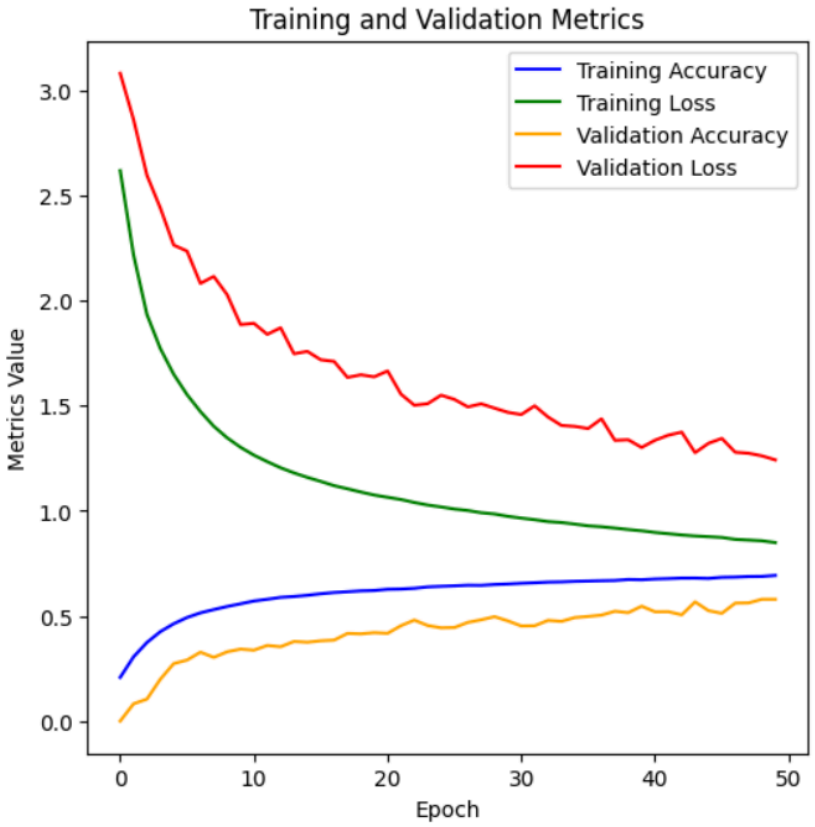


Image 43: Summarised history of 4-layered RNN model using Non-Augmented dataset.

Experiment 2: Augmented dataset

For experiment 2, we used the same 4-layered RNN model to test the performance of the model on the augmented dataset. The image below shows the different metrics that we use when evaluating the results of the different datasets.

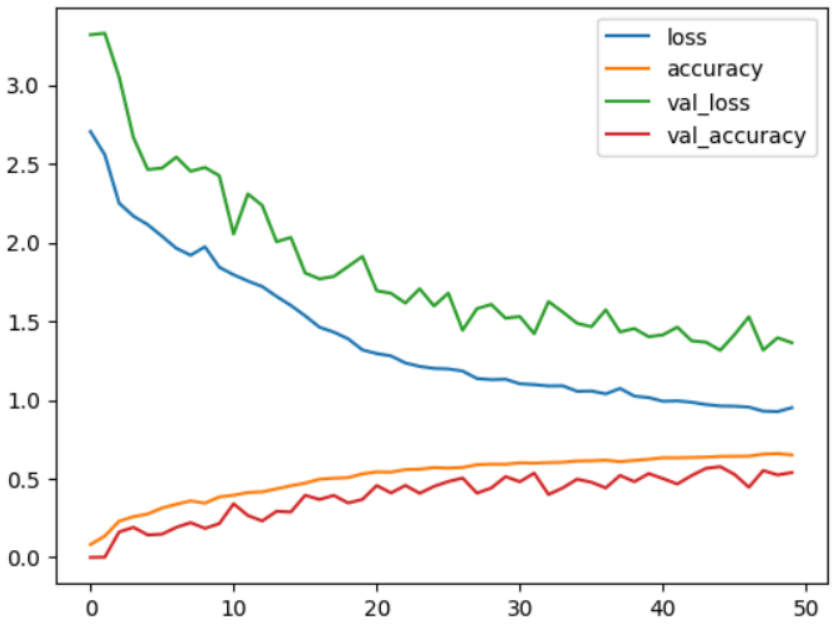


Image 44: Summarised history of 4-layered RNN model using Augmented dataset.

Experiment 3: Merged dataset

Moving to experiment 3, the merged dataset is used. The image below shows the training and validation metrics during the training process.

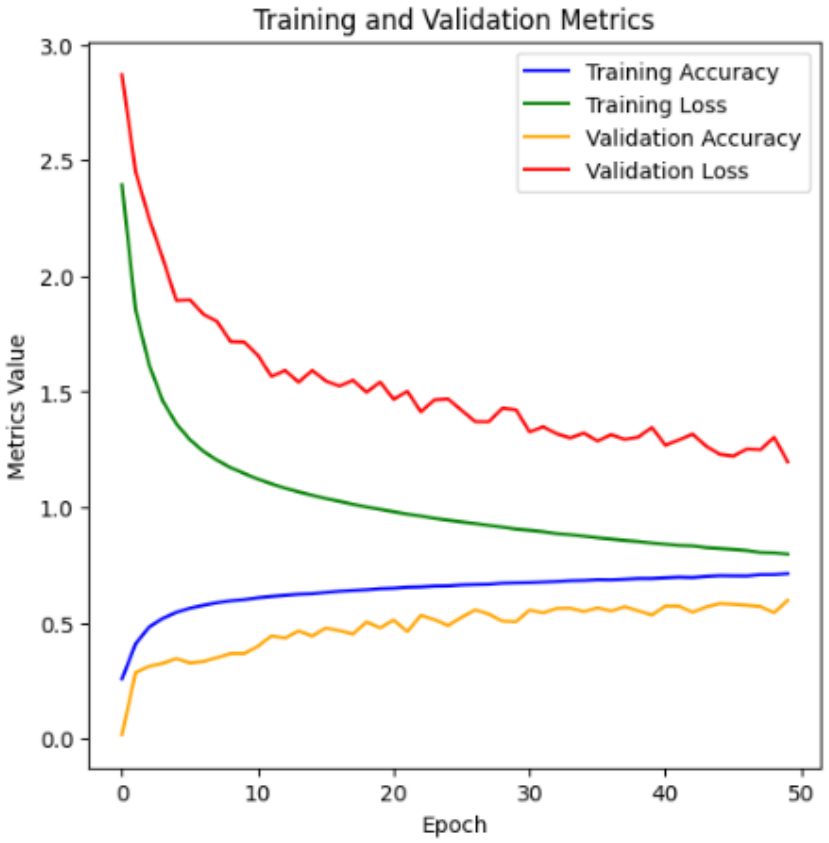


Image 45: Summarised history of 4-layered RNN model using Merged dataset.

The table below shows the result of the different dataset:

Table 13

Results from Non-Augmented, Augmented and Merged dataset

	Train Accuracy	Train Loss	Validation Accuracy	Validation Loss
Non-Augmented dataset	0.6926	0.8557	0.5800	1.2429
Augmented dataset	0.6509	0.9507	0.5388	1.3637
Merged dataset	0.7132	0.8008	0.5986	1.1967

Based on the results obtained in Phase 1, we can conclude that the model performed slightly better using the Merged dataset with validation accuracy of 59.86%. This could be the increase in the number of data for training or more variations in the dataset as it includes both Non-Augmented and Augmented data.

Phase 2: Model Variant Exploration

Now that we have determined the dataset to use, we will further different RNN variants to identify the best performing variant to use in the final evaluation.

Experiment 4: LSTMv1

The image below shows a version of the model that we experimented with. It is a 5-layered RNN model that consists of 2 LSTM layers and a Dense output layer.

```
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_20 (LSTM)              (None, 128, 128)          66560

 lstm_21 (LSTM)              (None, 128, 128)          131584

 lstm_22 (LSTM)              (None, 128)               131584

 dense_10 (Dense)            (None, 16)                2064

=================================================================
Total params: 331792 (1.27 MB)
Trainable params: 331792 (1.27 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Image 46: LSTMv1 Model Summary

The image below shows the history of the LSTM model and it also shows the point of divergence for the validation loss which allows us to identify when to stop the training of the model to prevent overfitting.

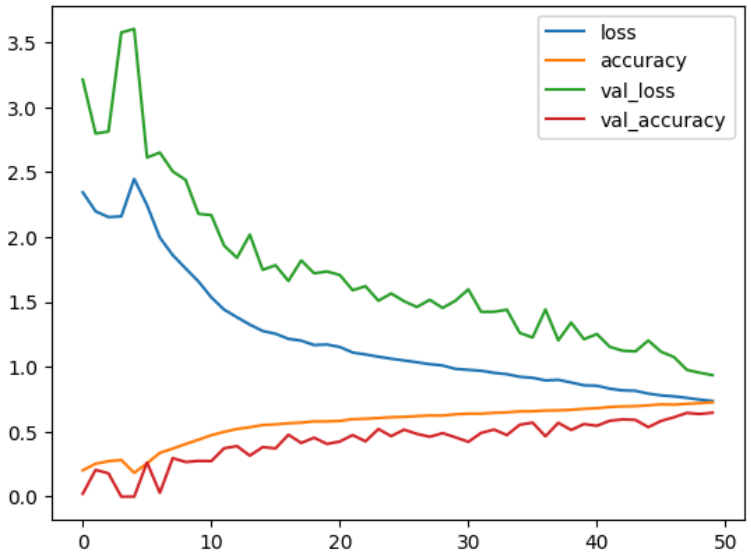


Image 47: Summarised history of LSTMv1 Model

Experiment 5: LSTMv2

The image below shows another variant of the LSTM model. For this model, it uses 3 types of layers, namely LSTM layer, Dense layer and a Dense output layer. After the input layer, there are 3 LSTM blocks. Each block starts with a LSTM layer, followed by BatchNormalization, LSTM layer, BatchNormalization and finally a dropout. The output of the last LSTM block will be passed into the 2 fully connected dense layers before being passed into the output layer.

Batch normalisation is used to normalise the net outputs from the previous layer's activation function for a more stable training which can help mitigate vanishing or exploding gradient issues.

Dropout is a regularisation technique that disregards certain nodes during training to reduce overfitting.

Layer (type)	Output Shape	Param #
lstm_28 (LSTM)	(None, 1, 1024)	4,907,008
batch_normalization_28 (BatchNormalization)	(None, 1, 1024)	4,096
lstm_29 (LSTM)	(None, 1, 1024)	8,392,704
batch_normalization_29 (BatchNormalization)	(None, 1, 1024)	4,096
dropout_22 (Dropout)	(None, 1, 1024)	0
lstm_30 (LSTM)	(None, 1, 1024)	8,392,704
batch_normalization_30 (BatchNormalization)	(None, 1, 1024)	4,096
lstm_31 (LSTM)	(None, 1, 1024)	8,392,704
batch_normalization_31 (BatchNormalization)	(None, 1, 1024)	4,096
dropout_23 (Dropout)	(None, 1, 1024)	0
lstm_32 (LSTM)	(None, 1, 1024)	8,392,704
batch_normalization_32 (BatchNormalization)	(None, 1, 1024)	4,096
lstm_33 (LSTM)	(None, 1, 1024)	8,392,704
batch_normalization_33 (BatchNormalization)	(None, 1, 1024)	4,096
dropout_24 (Dropout)	(None, 1, 1024)	0
dense_15 (Dense)	(None, 1, 64)	65,600
dropout_25 (Dropout)	(None, 1, 64)	0
dense_16 (Dense)	(None, 1, 64)	4,160
dropout_26 (Dropout)	(None, 1, 64)	0
dense_17 (Dense)	(None, 1, 16)	1,040

Image 48: LSTMv2 Model Summary

The image below shows the history of the model training using the merged dataset.

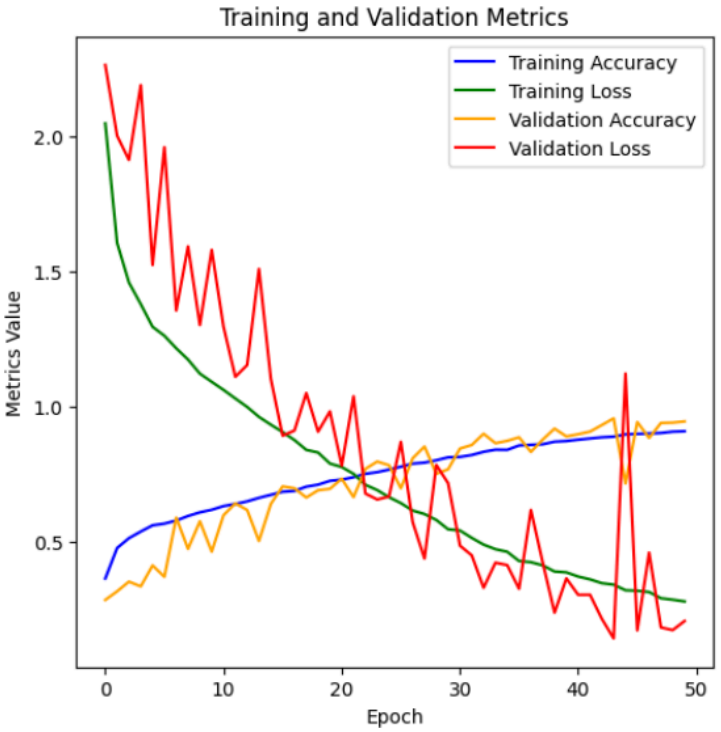


Image 49: Summarised history of LSTMv2 Model

Experiment 6: GRU

The image below shows a version of the model that we experimented with. It is a 5-layered RNN model that consists of GRU layers and a Dense output layer.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 128, 128)          50304

 gru_1 (GRU)                 (None, 128, 128)          99072

 gru_2 (GRU)                 (None, 128)               99072

 dense_6 (Dense)             (None, 16)                2064

=================================================================
Total params: 250512 (978.56 KB)
Trainable params: 250512 (978.56 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Image 50: GRU Model Summary

The image below also shows the history of the GRU model along with the point of divergence for the validation loss which allows us to identify when to stop the training of the model to prevent overfitting.

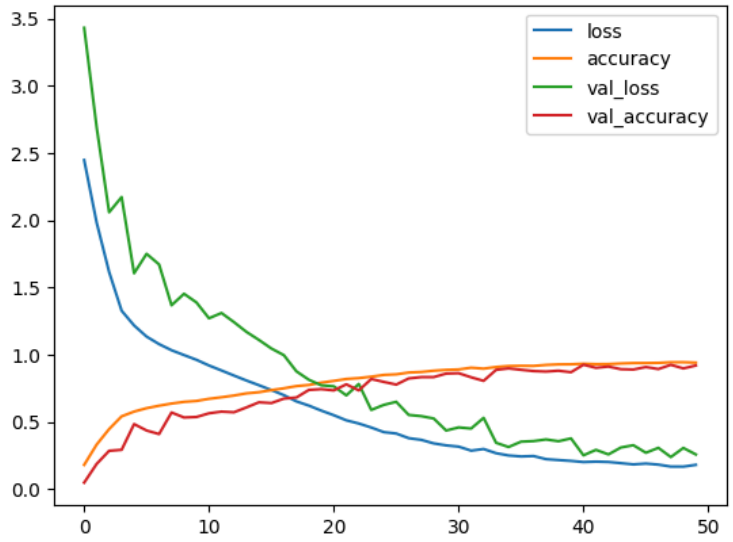


Image 51: Summarised history for the GRU Model

Experiment 7: GRU-LSTM

The image below shows a version of the model that we experimented with. It is a 5-layered RNN model that consists of two LSTM layers, a GRU layer and a Dense output layer.

Layer (type)	Output Shape	Param #
lstm_23 (LSTM)	(None, 128, 128)	66560
lstm_24 (LSTM)	(None, 128, 128)	131584
gru_4 (GRU)	(None, 128)	99072
dense_11 (Dense)	(None, 16)	2064

Total params: 299280 (1.14 MB)
Trainable params: 299280 (1.14 MB)
Non-trainable params: 0 (0.00 Byte)

Image 52: GRU-LSTM Model Summary

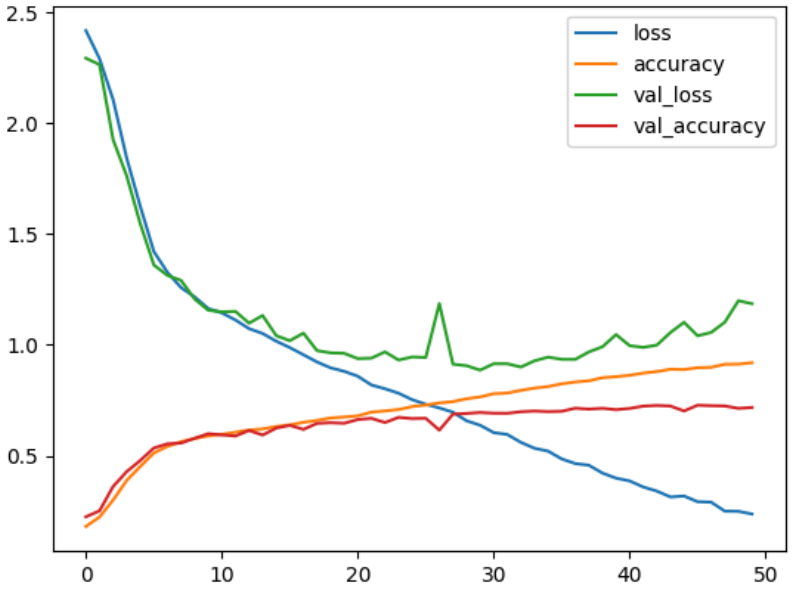


Image 53: Summarised history of GRU-LSTM Model

Looking at the results from the different experiments, we have decided to go with the LSTMv2 model that makes use of LSTM layers with normalisation and regularisation. It was able to obtain 94.44% in terms of validation accuracy, the highest among the 4 variants.

Table 14

Results from different models on the merged dataset

	Train Accuracy	Train Loss	Validation Accuracy	Validation Loss
LSTMv1 Model	0.7266	0.7368	0.6465	0.9362
LSTMv2 Model	0.9088	0.2750	0.9444	0.1732
GRU Model	0.9427	0.1669	0.9236	0.2376
GRU-LSTM	0.9381	0.2145	0.9231	0.2145

6. Conclusion and Future Work

6.1. Model Comparison

After finalising the 3 models, we will compare the models and determine the best model. The table below shows the performance measures of the model, which are test accuracy, precision, recall and F1 score.

Table 15

Performance measures for each model

	Test Accuracy	Precision	Recall	F1-Score
KNN	53.03%	0.5404	0.5303	0.5326
CNN	80.53%	0.8110	0.8053	0.8055
RNN (LSTMv2 Model)	76.46%	0.7682	0.7646	0.7648

From the table above, it is evident that CNN has the highest performance measures. However, we will choose RNN as the best model. RNN has the ability to process sequential data and is commonly used for audio classification tasks. In addition, the performance measures for RNN did not deviate far from CNN, with a 4% difference.

For this project, as the business goal is to enhance call centre customer service by prioritising customers based on their emotions, the key emotions we are targeting are, angry, sad and calm. Customers who are angry or sad will have higher priority, and customers who are calm will have lower priority when deciding which customers to attend to. This helps to improve customer satisfaction and retain customers effectively.

The confusion matrix below shows how well the RNN model can predict these targeted emotions for females and males.

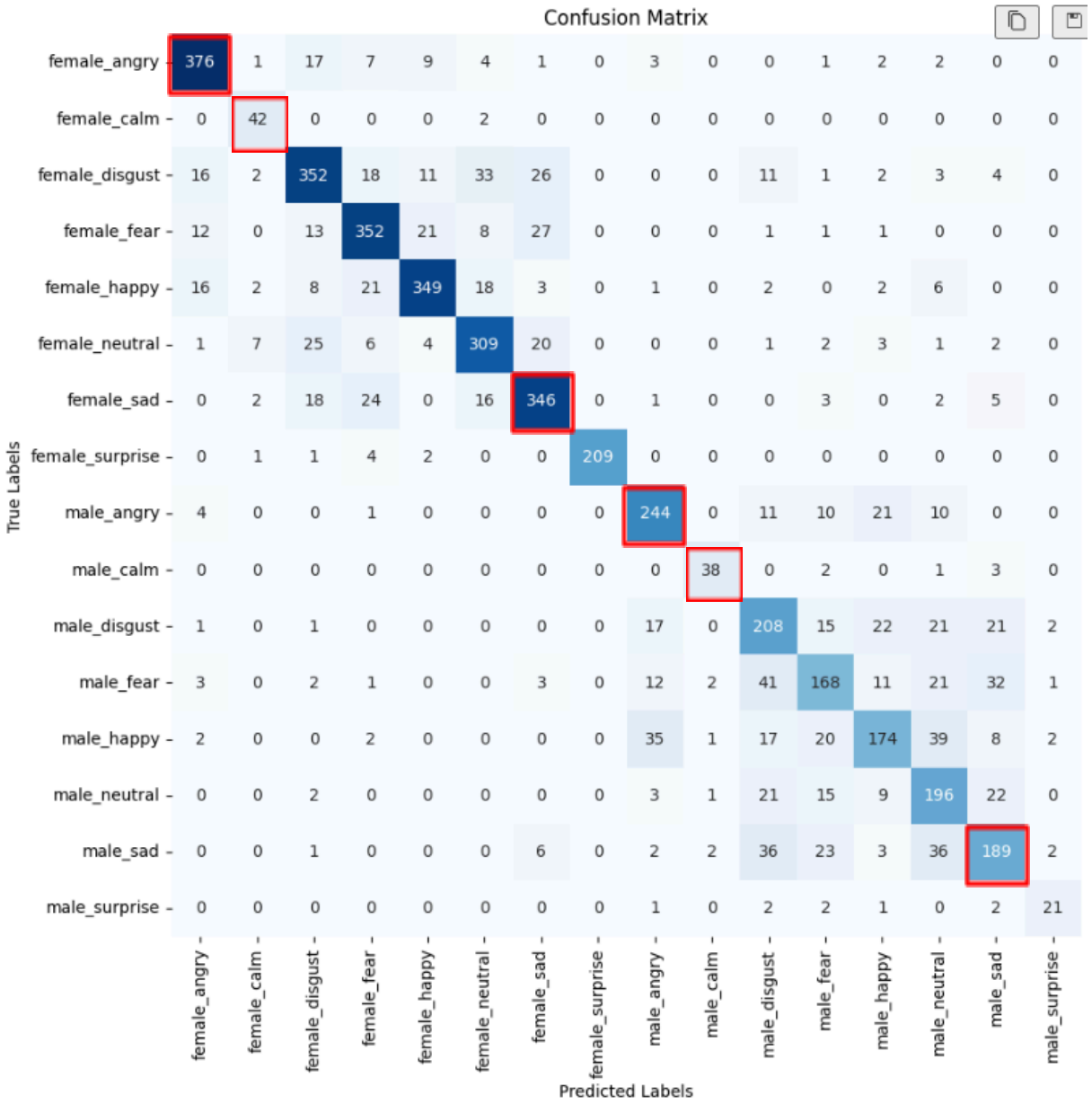


Image 54: Confusion matrix for RNN prediction on test data

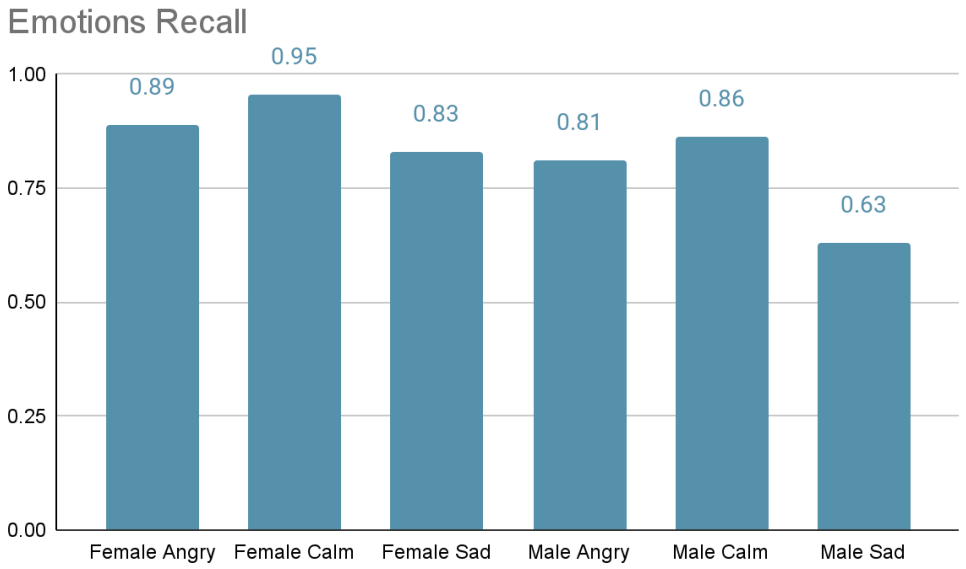


Image 55: Bar chart showing recall for the targeted emotions

The RNN performs well, with recall for most of the targeted emotions being over 0.8, except for Male Sad, which has a recall of 0.63.

6.2. Limitations

There are two limitations of our project. They are an imbalanced dataset and audio files produced by professionals.

Our dataset contains imbalance classes which is evident from the image below, which shows the class distribution for each emotion in our dataset.

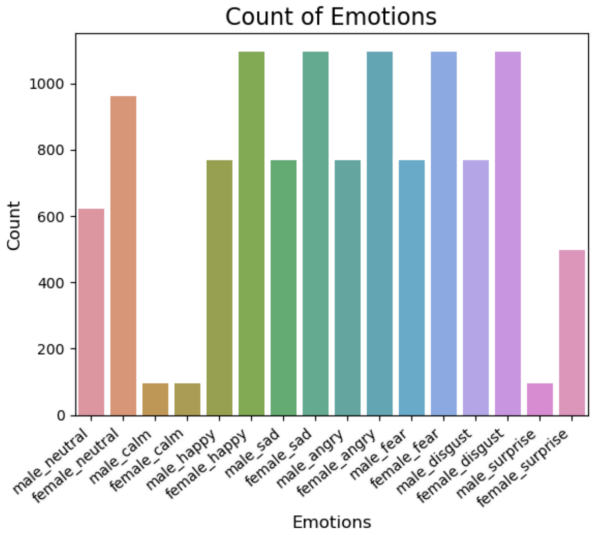


Image 56: Class distribution for each emotion

Imbalanced dataset can cause several issues when we are working on classification tasks. Firstly, the model will be biassed towards the majority class. The model objective during training is to minimise loss, and by focusing on majority classes, larger loss could be

minimised. Secondly, the model will generalise poorly to these minority classes as it did not have sufficient data for that class to train on. This also results in difficulty in detecting these minority classes. For our project, we tackled this issue using oversampling techniques like SMOTE or ROS. This method is effective when dealing with imbalanced data, but isn't the optimal option. The optimal solution to tackle this issue would be collecting more data for these imbalance classes to ensure our dataset has an equal number of data for each class.

The audio files used were produced by professional actors and this may not fully capture the natural variability and nuances of emotions in real-life. In addition, these are produced by a few people, which may not be representative for people in different cities or even countries. One solution for this is to obtain a more diverse dataset, which includes audio recordings of people in different demographics and different cultural backgrounds to make the model more generalizable for global use. These audio clips should also be captured in naturalistic settings.

6.3. Future Work

In terms of future work, we could first explore the use of attention mechanisms to improve the model's performance. Attention mechanism allows the neural network to selectively focus on important information in the input. It is similar to how our brains process information when we focus our attention on a specific information. One example is when we look at a photo. When we want to find how many cars are in the picture, our brain only focuses on cars and ignores every other information in the picture. The use of attention mechanisms helps improve model performance and computational efficiency. Attention mechanism has shown great potential in machine learning in these few years especially in tasks that deal with sequential data.

Next, a hybrid architecture could also be employed which combines CNN and RNN, leveraging the strengths of both models. Lastly, spectrograms can be used for CNN, which is an image of an audio wave. This may potentially help improve the performance of CNN in classifying emotions as CNN are designed for image related tasks.

6.4. Takeaways

From this project, we had 3 major takeaways. Firstly, we learnt how to develop our own machine learning models and understand the purpose of the different layers or hyperparameters during our fine-tuning process. This is crucial so that we can build and optimise our models that can potentially solve real world problems. Secondly, data is extremely important in any machine learning task as they are the information that the models learn from. We learnt the various issues and solutions in the data preparation process when we are researching and experimenting with the models. One great example would be the handling of imbalance data. Lastly, experiments are very important as it allows us to learn more as well as obtain actual results to prove or disprove hypotheses. This helps to reinforce our knowledge in machine learning and build a solid foundation for future machine learning projects.

7. Reference

Asgar, M. R. G., Hidayat, R., & Bejo, A. (2023b). Comparison Euclidean Distance and Manhattan Distance as Classification in Speech Recognition System. In Proceedings of the International Conference on Educational Management and Technology (ICEMT 2022) (pp. 454–463). https://doi.org/10.2991/978-2-494069-95-4_54

Alluhaidan, A. S., Saidani, O., Jahangir, R., Nauman, M. A., & Neffati, O. S. (2023, April 10). Speech emotion recognition through hybrid features and convolutional neural network. MDPI. https://www.mdpi.com/2076-3417/13/8/4750

Bain. (2015, April 8). Are you experienced? Bain & Company. https://www.bain.com/insights/are-you-experienced-infographic/

Bojanić, M., Delić, V., & Karpov, A. (2020, July 6). Call redistribution for a call center based on speech emotion recognition. MDPI. https://www.mdpi.com/2076-3417/10/13/4653

Brownlee, J. (2021, January 4). *Random oversampling and undersampling for imbalanced classification*. MachineLearningMastery.com. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

Brownlee, J. (2023, October 3). A Gentle Introduction to k-fold Cross-Validation. MachineLearningMastery.com. https://machinelearningmastery.com/k-fold-cross-validation/#:~:text=When%20a%20specific%20value%20for,learning%20model%20on%20unseen%20data.

Hag, A. (2021). *When will we use 1D convolution and when will we use 2D convolution in CNN? | ResearchGate*. ResearchGate. Retrieved March 29, 2024, from https://www.researchgate.net/post/When_will_we_use_1D_convolution_and_when_will_we_use_2D_convolution_in_CNN#:~:text=A%201D%20convolution%20is%20used,mostly%20used%20in%20Image%20classification

Jian-Feng, Z., Mao, X., & Chen, L. (2019). Speech emotion recognition using deep 1D & 2D CNN LSTM networks. *Biomedical Signal Processing and Control*, *47*, 312–323. https://doi.org/10.1016/j.bspc.2018.08.035

J., R. T. J. (2024, March 6). LSTMs explained: A complete, technically accurate, conceptual guide with keras. Medium. https://medium.com/analytics-vidhya/lstms-explained-a-complete-technically-accurate-conceptual-guide-with-keras-2a650327e8f2

Khoros. (2023, May 22). Must-Know customer service statistics of 2023 (so far). Khoros. https://khoros.com/blog/must-know-customer-service-statistics

Kostadinov, S. (2019, November 10). Understanding GRU networks. Medium. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

Lanjewar, R. B., Mathurkar, S. S., & Patel, N. (2015). Implementation and comparison of speech Emotion Recognition system using Gaussian Mixture Model (GMM) and K-Nearest Neighbor (K-NN) techniques. Procedia Computer Science, 49, 50–57. https://doi.org/10.1016/j.procs.2015.04.226

Livingstone, S. R. (2019, January 19). Ravdess emotional speech audio. Kaggle. https://www.kaggle.com/datasets/uwrfkaggler/ravdess-emotional-speech-audio

Lok, E. J. (2019b, August 30). Crema-D. Kaggle. https://www.kaggle.com/datasets/ejlok1/cremad/data

Lok, E. J. (2019a, August 24). Toronto emotional speech set (tess). Kaggle. https://www.kaggle.com/datasets/ejlok1/toronto-emotional-speech-set-tess

Onah, D. F. O., & Ibrahim, A. (2023). *Evaluating speech emotion recognition through the lens of CNN & LSTM deep learning models*. Retrieved March 29, 2024, from https://www.computer.org/csdl/proceedings-article/bigdata/2023/10386881/1TUOQOmliWQ

Paul, S. (2018, October 4). *Diving Deep with Imbalanced Data*. https://www.datacamp.com/tutorial/diving-deep-imbalanced-data

Phi, M. (2020, June 28). Illustrated Guide to LSTM's and GRU's: A step by step explanation. *Medium*. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Satpathy, S. (2023, November 17). *SMOTE for Imbalanced Classification with Python*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/

Singh J, Saheer LB, Faust O. Speech Emotion Recognition Using Attention Model. Int J Environ Res Public Health. 2023 Mar 14;20(6):5140. doi: 10.3390/ijerph20065140. PMID: 36982048; PMCID: PMC10049636.