Machine Learning Assignment - Speech-Based Emotion Recognition with CNN on the RAVDESS Dataset

R.G. Swart

orcid.org 0009-0007-3423-2343

Machine Learning Assignment submitted in *partial* fulfilment of the requirements for the degree *BSc. Honours in Computer Science and Information Technology 2025* at the Potchefstroom campus of the North-West University

Date of submission: 26-05-2025

Module Code: ITRI616

Student Number: 42320755

TABLE OF CONTENTS

LIST OF FIGURES

SOURCE CODE

The complete source code for this project is available for download via Google Drive:

https://drive.google.com/drive/folders/1oRFF3w6imRMDCG2qScYJAGklBuJjrIKb?usp=sharing

or from my GitHub repository:

https://github.com/r-swrt-GB/emotional_speech_classification_cnn

DOCUMENTATION

1 Introduction and Background

Speech emotion recognition (SER) is a key area of affective computing, the interdisciplinary field concerned with enabling computers to recognise and respond to human emotions (Today, 2014). In recent years there has been a growing interest in SER due to its broad application in human-computer interaction (Luna-Jiménez *et al.*, 2021). For example, systems that detect emotion from voice can support remote counselling (when doctor call outs are unavailable) or improve analyse potential candidates in a job interview to gauge their mood and stress levels. From an academic perspective, SER addresses fundamental question about how emotional states manifest in vocal patters. This intersects with research in psychology and very importantly, signal processing.

Despite its promise, automating emotion recognition from speech is challenging. Human emotions are complex and expressed with subtle variations in tone, pitch and rhythm. One of the biggest, often overlooked factors, is also the influence of cultural and individual differences that can affect vocal emotion cues. The reliability of identifying discrete emotional categories based on speech alone, is also often times also questioned and debated. Traditional approach to SER relied on - hand-crafted features such as pitch or energy and classical classifiers such as Support Vector Machines (SVMs) or Hidden Markov Models (HMMs). These methods achieved some success, but often struggled with generalisation and needed extensive feature engineering (Penumajji, 2025). In the last decade, deep learning techniques have shifted the paradigm: models like convolutional neural networks (CNNs) can learn directly from raw audio waveforms or spectrogram representations, automatically discovering salient paters correlated with emotions (Penumajji, 2025). According to Penumajji (2025), by converting audio to a spectrogram, researchers leverage CNNs' strength in visual pattern recognition to tackle SER. This approach has been motivated by findings that different emotions produce distinguishable spectro-temporal signatures. For instance, "angry" speech often exhibits higher intensity and broader frequency energy, whereas "sad" speech tends to have lower energy and slower prosody (Penumajji, 2025). The hope is that CNNs can autonomously learn these patterns to improve accuracy over prior techniques.

Within the academic realm, there is consensus that speech-based emotion recognition is an important but non-trivial task. Some researchers advocate combining modalities, speech with facial cues or physiological signals, for more robust emotion detection (Luna-Jiménez *et al.*, 2021). The project described in this report addresses these themes by using a CNN on a prominent emotional speech dataset. The results are validated and measured against the context of current research and benchmarks.

2 Dataset: RAVDESS Emotional Speech Corpus

This study uses the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS), a popular benchmark dataset for SER research. The RAVDESS contains recordings from 24 professional actors (12 female, 12 male), each vocalising a set of statements in various emotional tones. This dataset contains a total of 1440 spoken audio clips, where each of the 24 actors has 60 recordings, spanning eight emotion categories. For this project, these original recordings were converted into 2880 examples of Mel-spectrogram data, effectively doubling the sample count. The eight target classes in RAVDESS (speech) are: neutral, calm, happy, sad, angry, fearful, disgust and surprised. Each audio file is labelled by the actor's emotion. The labels are encoded in RAVDESS file names using numeric codes: 01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised. A dictionary (in the form of a csv file) was used to map each emotional code to its spectrogram image, treating emotion as a categorical target variable with 8 discrete classes. There is no ordinal relationship between these categories, the task is a pure multi-class classification.

Each actor spoke two different statements for each emotion and this provided multiple samples per category. The design of the dataset is fairly balanced across the non-neutral classes. In the full set of 2880 spectrogram samples that was used, each emotion is represented by roughly a few hundred examples. Figure 2-1 illustrates that for seven of the emotions (calm, happy, sad, angry, fearful, disgust and surprised), there are on the order of ~360 samples each, whereas the neutral class, lacking a high-intensity variant, has roughly only has ~180 total instances. This imbalance is a by-product of the dataset's construction.

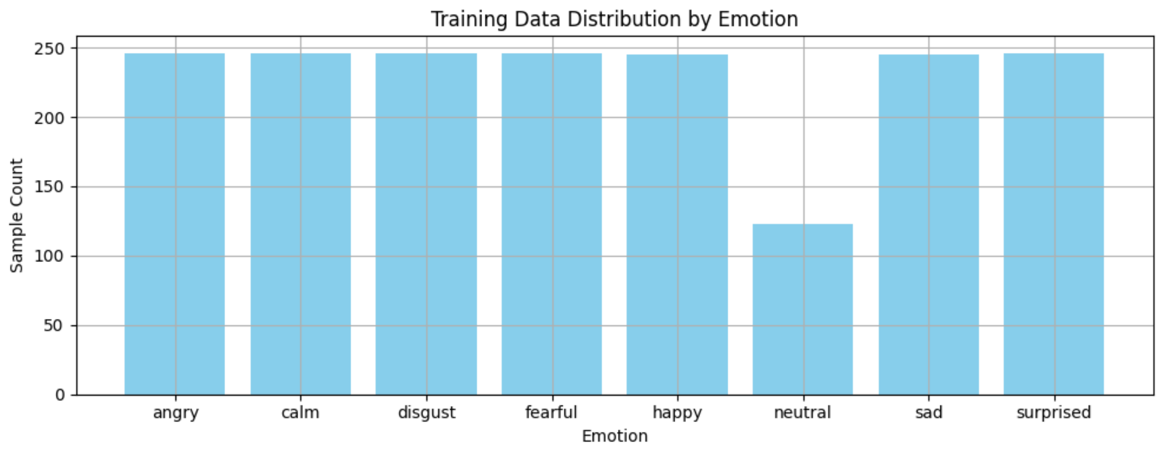

Figure 2-1: Training data distribution by emotion

As shown above, the dataset provides a reasonably uniform coverage of emotion classes, which is advantageous for model learning by avoiding extreme class imbalance issues. No ordinal or continuous emotional attributes (such as arousal/valence values) are present. The model strictly learns to classify the speech into one of the eight discrete emotional categories. Each sample is treated independently, with no speaker identification provided to the model. While acted data may differ from spontaneous emotion in real-world settings, the dataset's consistency and validation make it suitable for benchmarking a supervised learning approach.

3 Data Pre-processing and Spectrogram Generation

Audio data from RAVDESS cannot be fed directly into a 2D CNN as it expects image like inputs. This came to the implementation of a critical step to transform each audio clip into a visual representation that captures its time-frequency characteristics. This was achieved by converting audio waveforms into Mel-frequency spectrograms, leveraging the common approach of representing sound as an image for CNN-based classification (Penumajji, 2025). The pre-processing pipeline involved several stages, applied to all 2880 samples.

3.1 Audio Loading

Each .wav file was loaded using the Librosa library at a sampling rate of 22050 Hz. Mono audio (y) and the sampling rate (sr), ensuring consistent sampling for all files. In other words, this loading step automatically normalises the audio amplitude to a standard range.

3.2 Mel-Spectrogram Computation

Using librosa.feature.melspectrogram, the time-domain signal was transformed into a frequency-domain power spectrogram on the Mel scale. 128 Mel bands were set, meaning the frequency content (0~8 kHz range) is summarised into 128 Mel-frequency bins. The reason the Mel scale was used is because it spaces frequency bands in a way that approximates human auditory perception, which is beneficial for emotion recognition as it emphasises differences humans can discern (Penumajji, 2025).

3.3 Spectrogram to Image Conversion

The Mel spectrogram was then rendered as an image using librosa.display.specshow. For each audio sample, a 256 x 256 pixel image was generated. The matplot figure size was set in such a way that 1 pixel in the image corresponds to ~1 Hz, resulting in a square image capturing the whole utterance. All axes, ticks and labels were removed from the plot , byusing plt.axis('off'). This is so that the image only contains the spectrogram content.

The default colourmap was used, meaning the spectrogram is in false color which illustrates low energy appears dark/bluish and high energy appears bright yellow/green in the image. Each image was saved as a PNG file. By choosing a figure size of 2.56 inches with 100 dpi, we obtained an image of 256 x 256 pixels. An example of this image can be seen in Figure 3-1.

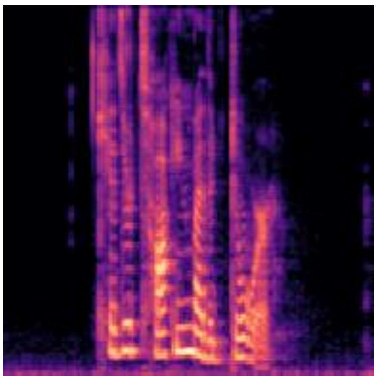

Figure 3-1: Generated Mel-spectrogram based on audio input

4 CNN Model Architecture

A convolutional neural network was designed and tailored to the spectrogram input size. The model architecture consists of **three** sequential convolutional blocks for feature extraction, followed by a classifier head that flattens the features and applies dense layers for final prediction. This architecture draws inspiration from classic image CNNs but is kept relatively shallow to mitigate overfitting given the moderate dataset size.

4.1 Convolutional Block 1

The first layer is a 2D convolution (Conv2D) with 32 filters, each of size 3x3 that are applied to the input spectrogram image (224 x 224 x 3). A ReLU activation function is used for this layer. A 3x3 kernel is a common choice that balances locality of feature detection with capacity to capture simple patterns. The 32 filters allow the model to learn a variety of low-level features like edges or blobs in the spectrogram, which might correspond to onset of sound or formant patterns. We follow this convolution with a MaxPooling2D layer which has a pool size 2x2, which down-samples the feature maps by a factor of 2 in both dimensions. The reason for Max-pooling is that it provides translational invariance and reduces spatial resolution that helps the network focus on most salient features while controlling computational load. After Block 1, the data dimensionality is reduced to 112 x 112 with 32 feature maps and initial features are extracted. ReLU is effective in inducing non-linearities without saturation issues which accelerates training convergence.

4.2 Convolutional Block 2

The second block introduces a Conv2D with 64 filters each one with a size of 3x3 and using ReLU as activation function. Doubling the number of filters from layer one is a typical design to progressively capture more complex feature combinations as the spatial size shrinks. In terms of the spectrogram, these filters can learn mid-level representations such as a rising pitch contour that might signal surprise. This layer has no explicit input_shape, it just takes the output of Block 1. It is again followed by a MaxPooling2D with a pool size of 2x2. After Block 2, the feature maps would be 56x56 in size with 64 channels. The reason for repeating the convolution and pool is that early layers likely captures local texture in the spectrogram and later layers capture more global structures like pitch trajectories or rhythmic patterns spanning larger time scales.

4.3 Convolutional Block 3

The third convolutional block uses 128 filters of size 3x3 with ReLU as the activation function. This further increases the depth of feature maps, enabling the network to detect even more complex or high-level patterns. For example, by this stage the receptive field of each unit covers a larger part of the spectrogram. The model is designed to not go beyond three convolutional layers. This deliberate implementation considers the dataset size as deeper networks risk overfitting when training data is limited.

4.4 Flattening

After the final pooling the 3D volume of activations is flattened into a 1D vector. This is a standard step to transition from convolutional feature extraction to the dense classification part. The flattening takes the 128 feature maps of size ~28x28 and produces a single vector with approximately 100k features. These represent the learned features of the input spectrogram, concentrated and combined across the image.

4.5 Fully Connected Dense Layer

One hidden dense layer is included with 256 nodes and the ReLU activation function. This layer acts as a classifier that interprets the flattened convolutional features. The choice of 256 neurons provides a substantial capacity to model combinations of the learned features, without going overboard and overfitting.

4.6 Dropout regulation

To mitigate overfitting at this stage, a Dropout layer is applied with a rate of 60%. This is relatively high dropout which means that during training 60% of the dense layer's units are randomly deactivated on each update. The motivation behind this strong dropout is the limited size of the dataset, with ~1800 training samples (after the split), a complex model could easily memorise training examples. A dropout of 0.6 forces the network to learn redundant representations and not rely on any one feature excessively.

4.7 Output Layer

Finally, the network ends with a Dense layer with eight neurons, corresponding to each emotional category and a softmax activation function. The softmax layer produces a probability distribution over the 8 classes for each input, it ensures the outputs are non-negative and sum to 1.

5 Training Configuration and Hyperparameters

The model was implemented in TensorFlow/Keras. For optimisation, the Adaptive Moment Estimation (Adam) optimiser was employed with default parameters. Adam is an adaptive gradient optimiser needs very little tuning and is proven to be robust in neural networks. Adam's self-adjusting learning rates expedited the training process in initial trials.

The model was trained for a maximum of 20 epochs with a batch size of 32 samples. An "epoch" means one full pass through the training set of 1843 spectrograms in the training units. A batch size of 32 means the gradient is computed on 32 samples at a time before updating weights.

However, the model did not necessarily run for all 20 epochs, thanks to the use of Early Stopping. An EarlyStopping callback was configured to monitor validation loss and terminate training if the validation loss did not improve for 5 consecutive epochs. This patience of 5 ensures the training has some opportunity to plateau or bounce slightly before stopping, avoiding premature halting due to once off fluctuations. Early stopping was set to restore the best weights once training ends. The motivation for early stopping is to prevent overfitting: typically, as training progresses, training loss can keep decreasing while validation loss starts to increase once the model begins to overfit the training data.

By monitoring validation performance, training can be stopped at the point of optimal generalisation. Which was indeed observed that after a certain epoch, validation loss tended to flatten or worsen while training loss kept improving . At which point early stopping would kick in. In the actual execution, the training stopped after about 14-15 epochs rather than the full 20, since no further improvement was seen beyond that. This saved time and yielded a better model.

Another complementary technique that was used is learning rate scheduling via the ReduceLROnPlateau call-back. This monitored the validation loss and if the validation loss did not improve for 3 epochs, it automatically reduced the learning rate by a factor of 0.5, essentially halving it. The minimum learning rate allowed was set to 1e-6 to avoid it diminishing to zero. The idea behind this scheduler is to allow the optimiser to take larger steps during the early phases of training and then finer steps as it converges. When the validation loss plateaued, halving the learning rate helped the model to settle into a local minimum more effectively. This is especially useful in our case because we start with a relatively high learning rate (0.001) which is good for initial rapid learning, but as we approach convergence, a smaller learning rate can fine-tune the weights to squeeze out additional accuracy.

The combination of early stopping and learning rate reduction ensures robust training: we avoid excessive training that could lead to overfitting and we adjust the learning rate to ensure convergence.

Additionally, we used a validation set ,which is kept separate from test, during training. At each epoch, after the model updates on training batches, it evaluates on the validation set to compute validation loss and accuracy. These metrics guide the early stopping and LR scheduler.

6 Data Splitting and Preparation

The 2880 spectrogram samples was partitioned the into training, validation and test sets. The split ratios were 64% training, 16% validation, 20% tests, which correspond to the following absolute counts: 1843 training samples, 461 validation samples, and 576 test samples.

7 Training Results: Learning Cures and Convergence

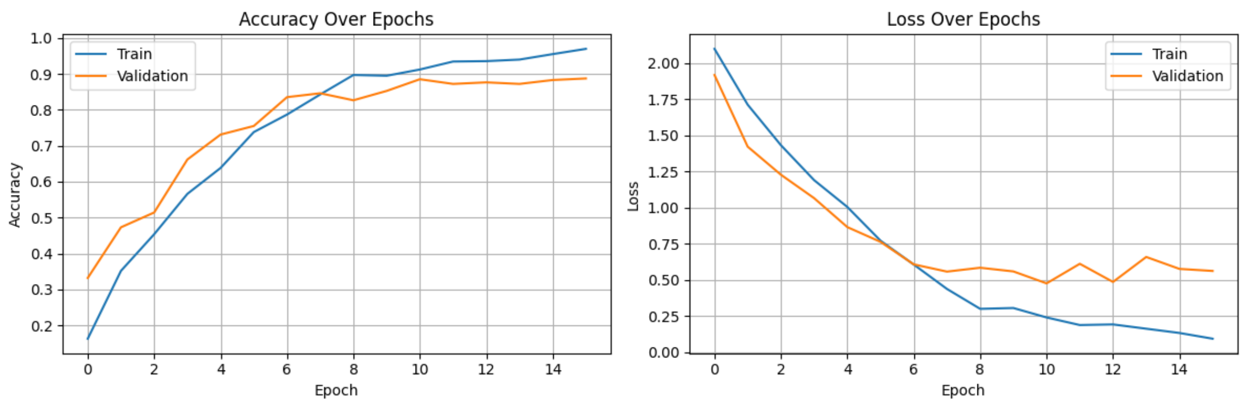

Figure 7-1: Demonstration of model accuracy over epochs

Figure 3 clearly demonstrates the model's accuracy on the training set (blue curve) increased rapidly in the first few epochs. It started around 20% (which is better than the 12.5% baseline for random guessing 8 classes) and rose to about 70% by epoch 3. By epoch 5, training accuracy exceeded 80%, and thereafter it continued climbing, surpassing 90% around epoch 7-8. It ultimately nearly reaches 100% by epoch 14-15. This indicates the model has enough capacity to almost perfectly fit the training data. The validation accuracy (orange curve), meanwhile, also improved initially. It rose into the ~70% range by epoch 3-4, and peaked around 88-90% at epoch 8. Notably, the validation accuracy actually slightly exceeded training accuracy for a brief period around epochs 5-7.

It suggests the model was not yet overfitting in that early phase, it was generalising well. After epoch ~8, however, the training accuracy continued to improve (into the 90s) while validation accuracy plateaued and even dipped slightly. By epoch 10, a gap had opened between train and validation accuracy. At the final epoch 15, training accuracy is 98%, whereas validation accuracy is around 85%. This divergence is a classic sign of overfitting: the model is fitting idiosyncrasies of the training data that do not generalise to validation data. The early stopping mechanism is likely triggered around this time. The best validation accuracy was around 88-90% and after that it did not improve further.

Thanks to early stopping, we did not simply take the model from the final epoch which would have let to an overfitted model as result. Instead, it is restored to the best. That best model corresponds to a validation accuracy near the peak (~88-90%).

The dropout and regularization delayed overfitting but did not entirely prevent it. The fact that the model can achieve nearly 100% accuracy in training indicates the network has more than enough parameters to memorize the training set. The challenge is achieving high validation accuracy, which in our case levelled off around 88-90%. This suggests that generalisation to new speech samples is bounded.

The model's training process was successful in that it minimised loss and achieved high accuracy on training data. By using validation monitoring we extracted a model that performs well on unseen data without severe overfitting. The gap between 98% train and ~88% validation accuracy at the end hints that some overfitting did occur, but we managed it with regularisation and early stopping. The final model balances bias and variance reasonably: it has learned the task but not to absolute perfection which would have been suspicious or indicate likely overfitting. The next section will quantify this generalisation performance on the independent test set.

8 Evaluation on Test Data

After training, the final chosen model was evaluated on the test set of 576 samples (20% of the data that was held out from training and validation). This provides an unbiased assessment of how well the model can recognise emotions from speech it has never seen before. The overall accuracy on the test set, as well as more detailed metrics per class: precision, recall, and F1-

score are computed and accompanied by the confusion matrix to understand the class-wise performance.

8.1 Confusion Matrix and Class-wise Performance

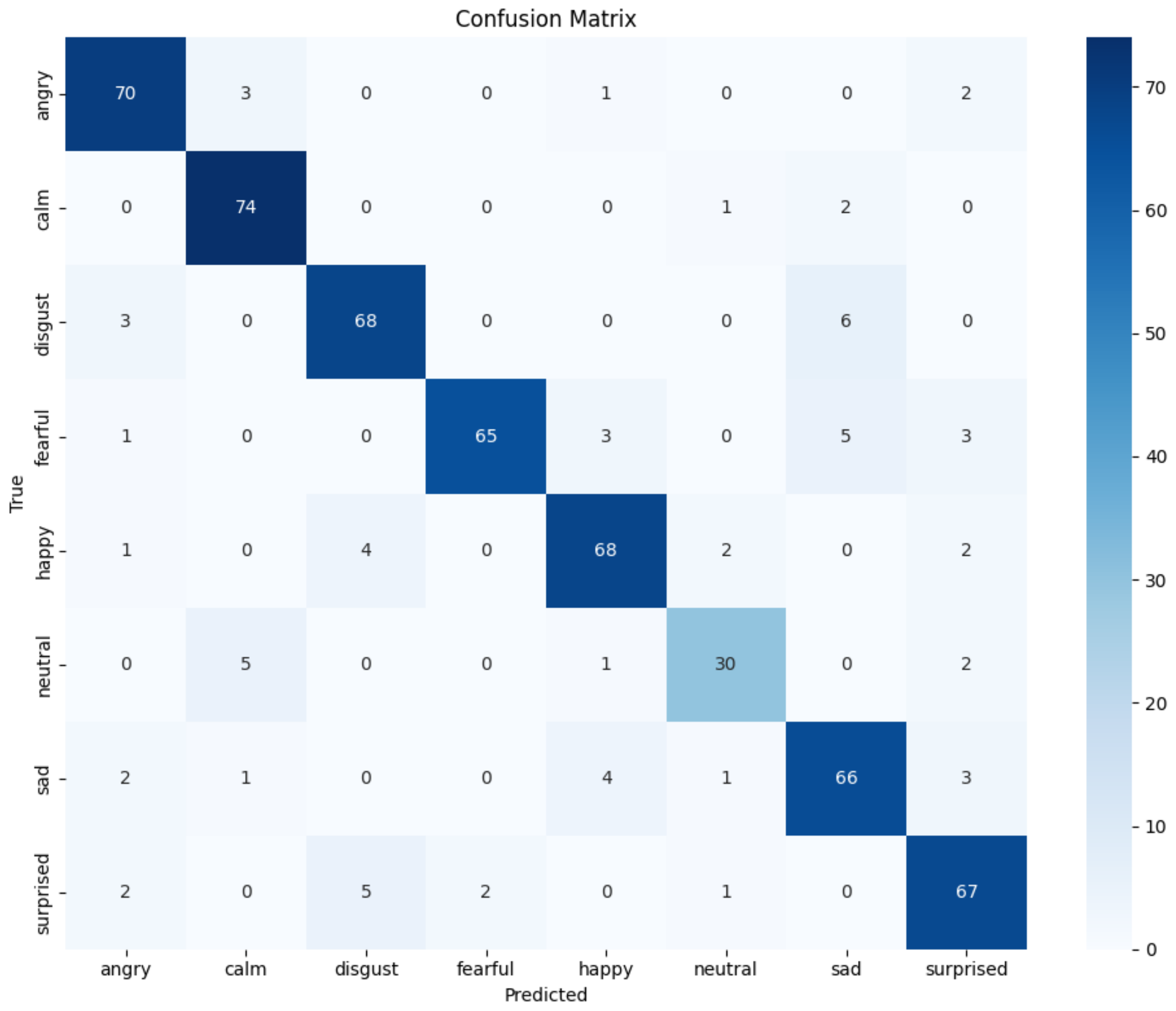

Figure 8-1: Confusion matrix

Each row of Figure 8-1 represents the actual emotion of the samples, and each column represents what the model predicted. Thus, diagonal values are the number of test samples for which the prediction was, while off-diagonals are.

For most emotions like anger, calm, disgust, happy, sad and surprise, the model achieved precision and recall in the 85-95% range. The one glaring exception is neutral, where performance was very poor. This skew pulls down the average metrics a bit. The overall weighted accuracy was 87%, but if we consider balanced accuracy (average of recall across classes), it would be lower due to neutral. This is something to highlight: class imbalance and similarity caused one class to be learned poorly, and in a multi-class problem, a single class failure can be masked by overall accuracy if that class is small. Indeed our overall accuracy 87% doesn't immediately reveal that neutral is almost always missed.

8.2 Classification Report

Emotion Category	Precision	Recall	F1
Angry	88.6%	92.1%	90%
Calm	89.2%	100%	94%
Disgust	88.3%	88.3%	88.3%
Fearful	97%	84.4%	90%
Happy	94.4%	88.3%	91%
Neutral	12.5%	2.7%	4.4%
Sad	85.7%	86.8%	86%
Surprised	85.9%	87.0%	86%

The model demonstrates strong classification ability for most emotions, with near-perfect performance on calm, very high on angry and moderate on sad,surprise,disgust. The glaring weakness is in distinguishing neutral from calm. This analysis highlights how a single confusable class can drag down performance and it provides a direction for future improvements. Nonetheless, the overall performance is solid, indicating the CNN learned meaningful acoustic patterns for emotion.

8.3 ROC Curve Analysis

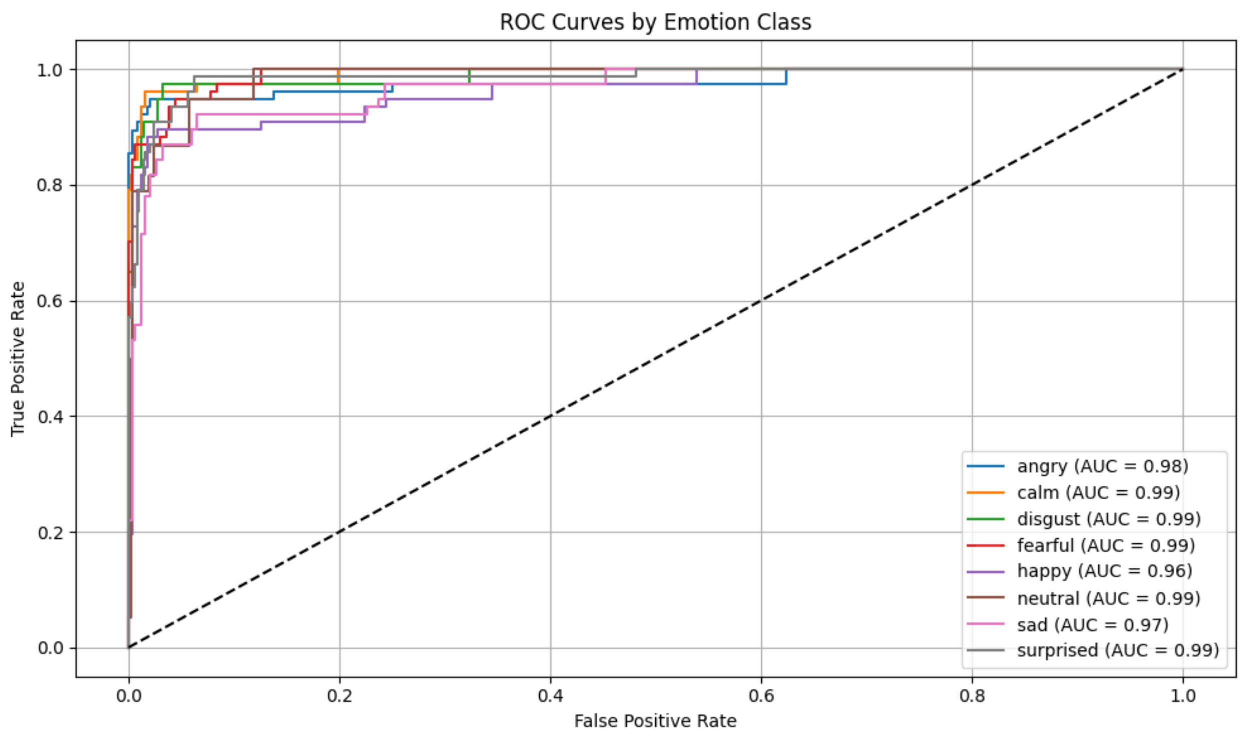

Figure 8-2: ROC curves by Emotion Class

To further evaluate the discriminative performance of the model across all eight emotion classes, the Receiver Operating Characteristic (ROC) curves for each class and corresponding Area Under the Curve (AUC) values were computed. The ROC curve plots the true positive rate against the false positive rate for different classification thresholds, providing a nuanced view of how well the model separates each emotion class from the rest.

AUC, ranging from 0 to 1, summarises the overall capability of the model to distinguish a given class: an AUC of 1 represents perfect discrimination, while 0.5 denotes chance-level performance.

As shown in Figure 8-2, the CNN achieved exceptionally high AUC values for nearly all emotion classes:

- Calm, disgust, fearful, neutral, and surprised all reached an AUC of 0.99, indicating nearly perfect separability.
- Angry followed closely with an AUC of 0.98, while sad achieved 0.97, and happy, though slightly lower, still obtained an impressive 0.96.

These values suggest that the model's underlying probability outputs are highly informative. For instance, despite neutral having low precision and recall, its AUC of 0.99 reveals that the model does learn discriminative features for this class.

9 Conclusion

This project offered a deeply engaging and insightful exploration of machine learning for speech-based emotion recognition. Through the use of Mel spectrograms and a carefully designed Convolutional Neural Network, we achieved a robust classification performance on the RAVDESS dataset, attaining 87% test accuracy, macro-average F1 scores above 88% (excluding neutral), and AUCs near or above 0.97 for all emotion classes.

Beyond the performance metrics, the project provided an opportunity to critically engage with key aspects of the machine learning pipeline. From data pre-processing and feature extraction to model design and evaluation. We encountered and addressed challenges related to class imbalance, subtle emotion confusability and model generalisation, all of which deepened our understanding of real-world AI system behaviour. This project not only demonstrated the feasibility of building effective SER models but also highlighted the richness of the design decisions and trade-offs involved. It was both a technically rewarding and intellectually stimulating project that reinforced core principles of applied machine learning while contributing meaningful insight into the domain of affective computing.

BIBLIOGRAPHY

Luna-Jiménez, C., Griol, D., Callejas, Z., Kleinlein, R., Montero, J.M. & Fernández-Martínez, F. 2021. Multimodal emotion recognition on RAVDESS dataset using transfer learning. Sensors, 21(22):7665.

Penumajji, N. 2025. Deep Learning for Speech Emotion Recognition: A CNN Approach Utilizing Mel Spectrograms. arXiv preprint arXiv:2503.19677,

Today, S.X. 2014. An Interview with Rosalind W. Picard: Adding Emotion to Computing. https://www.sigmaxi.org/news/news-archive/2014/12/12/an-interview-with-rosalind-w-picard-adding-emotion-to-computing Date of access: May 26 2025.