Emotion Classification on Audio Speech Data

STAT3007 Project Report[1][2]

Jordan Foss[3] **Yuzhe Jie**[4] **Yue Xiao**[5]

June 3, 2021

Abstract

Different emotions can be recognised through the variety of tones and pitches of the speaker. While a trivial task for a human, it is still a difficult and interesting problem in modern machine learning without referring to the linguistic meaning embedded in the speech. Through this report, we investigate various methods for processing and classifying audio data into a class of emotions. Various models architectures were trained and test, with a emphasis on Convolutional Neural Newtorks (CNNs) and on the mel-spectrograms - a perceptually relevant representation of human audio. We also proposed a decomposition methods for the mel-spectrograms - RGB decomposition, as well as a method of extracting time-dependent features using LSTM layers. Lastly, we integrated an autoencoder with a CNN to boost the performance of classification on noisy data.

[1] Github link for the project - `https://github.com/JordanFoss/STAT3007_Project`

[2] Google Drive - `https://drive.google.com/drive/folders/1n9xwoN4oa4teVaBLyc5bvzuJZ70zhhQk?usp=sharing`

[3] Student Number - 45302282

[4] Student Number - 44575986

[5] Student Number - 45307157

We consent for this to be used as a teaching resource.

Contents

1 Introduction

Speech comprehension is the ability to understand the contextual meaning from the speech signals. It is a well developed ability in humans, as it happens almost instantly and naturally despite the high complexity of the speech signals. In fact, a human infant less than 1 year old is able to recognise familiar voices [1]. To fully understand the meaning behind speech, the underlying neurological and psychological processes in the brain must process the information in three different levels: 1) recognising words, 2) constructing sentence meaning, 3) developing contextual understanding.

Developing contextual understanding from speech signals is not limited to the linguistic meaning inside the messages. It also includes other cues presented in the amplitude and tone of the speaker, as well as the speed of speaking. Recognising and categorising the information embedded in such cues might provide more information about the speech, reinforcing the development of contextual meaning and prompting appropriate responses. Recent Psychology studies in [2] suggested that human listeners can understand accurately the speaker's feeling at the time of communication from the emotional cues faster than extracting verbal contextual meaning, invariant across cultures. However, this is not necessarily the same for computers.

The advancement of natural language processing (NLP) coupled with the success of speech recognition or transcript generation provides a wide range of efficient industrial applications. Although using sentiment analysis on the transcripts can effectively extract emotional information presented in the linguistic information, the task of recognising emotions through audio speech signals has not yet entirely been solve and is only limited to a number of small scale applications [3]. Extracting emotional information from emotional cues isolated from linguistic information thus becomes an interesting problem in human-computer interactions.

With the vast amount of audio data being generated daily, solving this problem prompts additional potential industrial applications. One major example is the automatic user reviews from phone call customer support. As many companies already store all this audio data for review, it stands to reason that a method of automatically generating user reviews based on audio data would have a major impact of how companies provided their user experience. The state of the art audio analysis software - AWS's Connect [4], a commercial cloud contact center, only offers the feature of sentiment analysis based on customer transcripts through NLP techniques. It requires the handing picking keywords that represent negative/positive sentiments. Hence, the ability to generate reviews free of such manual tuning or transcript dependency thus becomes the main focus of improvement in this area. Moreover, many modern devices provide a virtual assistant that allows the user to interact using only their voice. While these virtual assistants respond very well to user speech meaning, there is potential to explore responding to the user based on how they talk. This opens up potentially using emotion recognition software in virtual assistants to provide a more personalised user experience. Finally, it can served as a tool to assist in identifying emotions for those who have difficulty in doing so (e.g. patients with Autism Spectrum Disorder(ASD)).

It has been showed that emotional recognition based on audio-visual data presented promising results. However, majority of such success is dependant on the correlation between the visual and the audio features[5]. Since real time analysis of visual data is significantly more expensive than audio data, this coupling limits the possibility of industrial applications. Hence, audio-only model is worthwhile to investigate in searching for a low-power alternative of the task.

In this report, we aim to explore different architectures for the task of emotion classification. Adhering to real-life speech signals, where the Audio data mostly comes with noises which makes the context and meaning of it incomprehensible, the problem is extended to classify emotions from noisy audio data. In this case, techniques for noise-invariant architectures are also explored.

2 Related Work

Speech Emotion Recognition (SER) has been an active research area over the past decade and comprehensive studies were conducted to investigate the task. Earlier studies for SER consisted of finding low-level acoustic speech parameters that are correlated to emotions with the use of standard classifiers such as Support Vector Machine (SVM), Gaussian Mixture(GMM) or fully-connected shallow Neural Networks [6]. However, the identification of the most representative acoustic features that can characterise different emotions is an elusive challenge using these conventional techniques. Moreover, these methods require prior knowledge of the "best features" and have shown inconsistencies between studies [6].

With the recent advancement and robustness of convolutional Neural Network (CNN), where the success of CNN architectures can extend to generalised 2-dimensional matrix arrays [7], a new research focus of SER is shifted to automatic low-level acoustic feature extraction procedures offered by CNNs. To utilise CNN architectures in analysing audio data, an additional pre-processing step is taken to convert the raw audio samples into a 2-dimensional array. This approach no longer adheres to the concept of an end-to-end framework, therefore, techniques of such feature transformations should be carefully implemented depending on the applications.

The details will be discussed in the our solution.

3 Dataset

The dataset used for the speech data is from The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) [8]. We selected the audio speech from RAVDESS dataset as it matches our goal in this project. It contains:

- 7356 audio files in total, consisting of
- 2 data channels - speech or song (only speech is used here)
- 24 actors (12 males and 12 females)
- 8 emotion classes (only 5 are used in this study)
- 2 statements
- 2 repetitions (only used 1 of them)
- normal or high intensity (only high intensity is used)

Each audio speech is the composition of 5 components, namely emotions (neutral, calm, happy, sad, angry,fearful, disgusted, surprised), emotional intensities (normal, strong), statement ("kids are talking by the door", "Dogs are sitting by the door"), repetition (first, second), and actor (from 01 to 24).

Because of the similarities in emotions and repetitions, the whole speech dataset still sounds like there are lots of redundancies so we need to do a further subset selection of somewhat distinct speeches. The finalised dataset contains only 5 emotions (calm, happy, sad, angry, surprised), strong emotional intensity, 2 statements, the second repetition and all 24 actors. Hence, each actor has 10 samples of audio speech.

In addition, we also selected a dataset for inducing realistic noises. That is, the Microsoft Scalable Noisy Speech Dataset (MS-SNSD) [9]. The form of noises consist of background sounds from object movements or machine operations. Any noises that resemble human voices (e.g. babbling) are not studied. Speech to Noise Ratio (SNR) is selected at 40.

4 Steps taken to solve the problem

To solve the problem, our attempts include the initial pre-processing of audio data before inputting into a CNN model for classification. We also investigated and compared the performance of various CNN architectures in this task.

4.1 Pre-processing

The pre-processing of raw audio aims to convert the signal data to mel-spectrograms, a 2-dimensional array on which our deep learning models are build for.

4.1.1 Noise Generation

Our noisy speech is the combination of clean speech and noise audio samples. It should be noted that the noises being background sound should be consistent and loud enough to be heard. Thus, we only picked 12 types of noise from the MS-SNSD. (Air-con, Copy Machine, Kitchen, Munching, Resturant, Shutting door, Squeaky chair, Station, Traffic, Typing, Vacuum cleaner, Washer/Dryer). Each type of noise has several versions and in total give us 79 noises. The noise and clean speech are combined under the same frequency (48kHZ, the same frequency for speech), keeping the quality of both the noise and speech. SNR of 40 is chosen to scale the noise to the level that is not too loud nor too soft. We combined each piece of clean speech with each noise, yielding 790 noisy speeches for each actor.

4.1.2 Silence Truncation and Downsampling

We have found that there are periods of silence before and after the speech in all the speech samples. This means truncation is needed as silence does not contribute any meaningful information. To accomplish this we cut off the first and the last connected components where the magnitude of the amplitude is under a certain threshold. In this case, we manually picked such a threshold as 0.001. We also down-sampled the sampling frequency of the audio to 16kHz after truncating, to produce audio samples that have a relatively longer duration of time and sufficient signal frequency resolution as per the Nyquist-Shannon sampling theorem [10]. In order to fully recover a speech signal produced by a human voice, which contains frequencies up to 8kHz, the sampling rate must be at least twice as much.

4.1.3 Normalisation

Normalisation is an important pre-processing step in machine learning as it makes data within the same scale while keeping their original distributions. In our scenario, the amplitude of the raw audio files is normalised to have zero mean and unit variance. That is,

$$\frac{X - \mu}{\sigma}$$

where X is the audio sample, μ is the mean and σ^2 is the variance.

4.1.4 Padding/Cropping

One problem with our speech signal data is that they have varying timestep lengths. Since in later pre-processings steps mel-spectrograms will be computed and serve as an input to the CNN, they are required to have fixed dimensions. Hence, it is necessary to make all the speech data the same length. In the case where the audio has duration longer than the desired length, we discard any audio samples past the cut-off. On the other hand, where the audio data has shorter duration, the audio data is padded by concatenating an array of zeros until the audio sample reaches the desired duration.

We chose concatenating the zero array before the audio (pre-padding), as it was shown in [11] that pre-padding results in a drastic improvement of performance for LSTM models, while having a negligible impact on CNNs.

The duration of the padded/cropped sample is fixed at 2 seconds, as this is the rounded average duration for the truncated samples. Therefore, setting it to the average length can preserve most of the speech information on average across our dataset, while avoiding too much redundancy induced by padding.

4.1.5 Mel-spectrogram

Raw audio waveforms only represents the loudness of the audio signal through an amplitude measure at each time-step. It does not offer any useful information about the compositions of different frequencies. This implies that certain feature transformations are needed as the emotional cues, which are heavily dependent on the tone or pitch of the speaker, is mostly representative by the change in the frequency composition.

The most effective transformation that reveals the frequency composition is via Short-Time Fourier Transform (STFT), which decomposes the amplitude waveforms into a frequency-time domain known as a log-spectrogram. It is a 2-dimensional array that shows the magnitude of each frequency wave at every time step, see Figure 1.

While the log-spectrogram provides frequency-time representation of the audio data and perceptually relevant amplitude information, it does not take into account the non-linearity of how humans perceive frequencies. For this task, which is about human perception of emotions through audio analysis, generating such perceptually relevant frequency information becomes a crucial pre-processing step. Hence, we consider a non-linear scaling for the log-spectrogram, known as the Mel-scale. It is a log-scaling consisting of the following function [12]:

$$Mel(f) = 2595 \log_{10}(1 + \frac{f}{700})$$

where f is the frequency in Hz.

The mel-scaling marginalises the difference in high frequency regions while leveraging the same difference in low frequency regions. This mimics the human perception of the change in frequencies [13]. The effect of the scaling is shown in Figure 1.

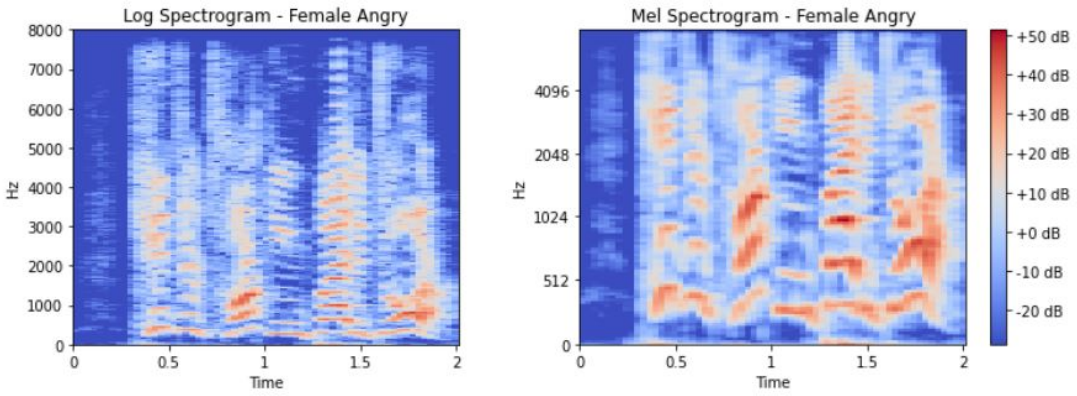


Figure 1: Frequency-time representation of an aduio signal for a female actor with angry emotion. Left: log-spectrogram of the signal. Right: log-spectrogram of the signal after mel-scaling

The complete pre-processing pipeline can be summarised in Figure 2.

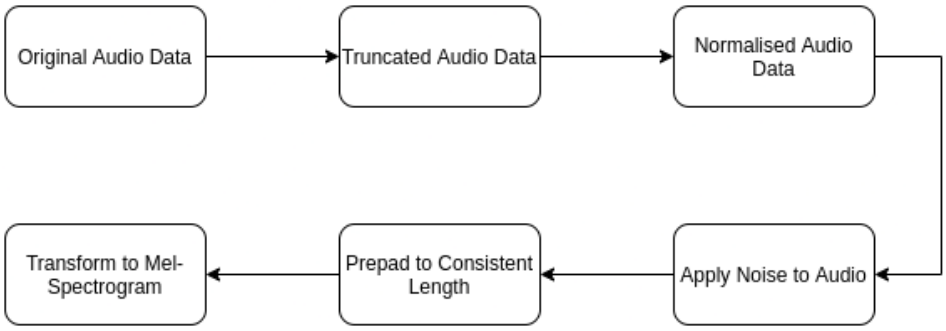


Figure 2: Full audio preprocessing pipeline

4.1.6 Training/Test Dataset

Considering the training time and limited computation resources we can access on Google Colab, only a portion of clean and noisy speeches are selected for training and testing: 430 out of 790 for noisy speech and corresponding clean speech for each actor, included a smaller number of versions for each noisy type. Moreover, we further split the dataset amongst actors, to prevent over-fitting on speaker-dependent features by selecting data from 16 actors (8 females 8 males, totally 6880 pieces of data) to be our training set, data from the rest of 8 actors (4 female and 4 male, totally 3440 pieces of data) to become our testing set.

4.2 Models

Using Mel-spectrograms as input data allows us to effectively utilise the CNN architectures for the task. Aside from the CNNs, we further investigated the performance of a common computer vision techniques - RGB decomposition. LSTM architecture was also considered to extracat time-dependent features of mel-spectrograms.

Additionally, as we wanted to construct a generalised model that adheres to real-world environments, we also investigated denoising techniques. This led us to implement a denoising autoencoder that later was stacked with various classifiers for exploration.

4.2.1 Baseline CNN

[14] has shown that a vanilla CNN is able to successfully extract features that can be used for classification. Extending on that, the first model we constructed and tested on the model was a pure CNN model with the structure shown in Figure 3. This model serves as the baseline model.

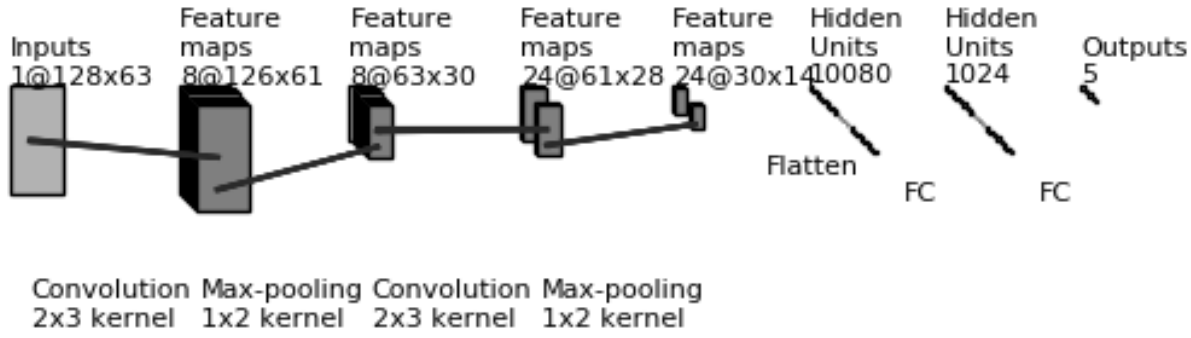


Figure 3: Baseline CNN Model

4.2.1.1 Filter size tuning

We focused our tuning on the size of filter kernels as well as max-pooling layer sizes. This is due to the fact that the scaling of the rows and columns of mel-spectrogram is different - one represents the increment in frequency while the other represents the increment of time. Unlike images, where both axis represent the spatial domain and are independent on the spatial coordinates, mel-spectrograms have a dependency on the time domain [15].

For this reason, as suggested in [15] which concluded that rectangular kernel/max-pooling sizes performed better for spectrograms, we have investigated the effect on different combinations kernel sizes and max-pooling sizes. This included any rectangular filters up to 4×4 for filtering and 3×3 for max-pooling. Our result suggested that a rectangular kernel size (2×3) and max-pooling size (1×2) achieved the best accuracy.

The rest of the hyperparameters are tuned through the usual grid search and are summarised in Table 1.

Hyper parameters	
Activation	ReLU
Learning Rate	0.001
Momentum	0.4
Batch Size	10
Optimiser	SGD
Dropout Layer	0.25
Loss Function	Cross Entropy Loss
epoch number	20

Table 1: A summary of the hyperparameters for the baseline CNN model

This model structure and hyperparameters were chosen after tuning all hyper parameters of the model. The graphs generated for this tuning can be found in Figure 19 in the Appendix.

4.2.2 CNN + LSTM

As mentioned earlier, the fundamental difference between image classification and SER using CNN lies in the fact that audio data is necessarily time-dependent. This still holds after the construction of mel-spectrograms.

To capture the time-dependent information while utilising the frequency information provided by the mel-spectrogram, we augmented a CNN model following by a LSTM model, as suggested in [16] and [17].

The input mel-spectrogram was split into 3 segments along its time axis. Each segment contains 21 time steps and served as an input into the CNN model. The output of the CNN model was then flatten, and inputted into the LSTM model. The LSTM model connects the information from each time segment to extract time-dependent information for classification.

The CNN implementation used for this model was the baseline model mentioned above. The model structure is demonstrated below in Figure 4.

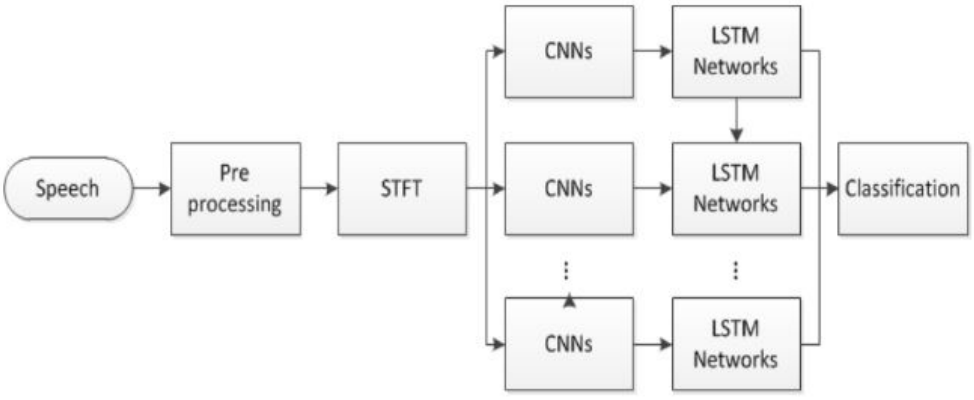


Figure 4: Model structure for CNN + LSTM. Source: [17]

The kernel sizes are inherited from the baseline model as it was effective in identifying frequency features. The LSTM model contains 2 layers which is also bi-directional to allow inference of frequency information from both past and future [17]. The rest of the hyperparameters are summarised in Table 2. See Figure 20 and 21 in the Appendix for more supplementary graphs regarding the grid search involved in the hyperparameter tuning.

Hyper parameters	
Activation	ReLU
Learning Rate	0.01
Momentum	0.2
Batch Size	10
Optimiser	SGD
Dropout Layer	0.25
Loss Function	Cross Entropy Loss
epoch number	40
LSTM layers	2

Table 2: : A summary of the hyperparameters for the CNN+LSTM model

4.2.3 CNN + RGB

The final augmentation we implemented to the CNN model was a RGB decomposition of the mel-spectrogram as an additional preprocessing step, before it was inputted in to the model. This approach was inspired by a paper that attempted the same method [6]. Additionally, the CNN for this model is the same as the Baseline CNN and thus has the same parameters1.

This method involved multiplying each pixel in the mel-spectrogram by one of 64 colours, based on its decibel value. These 64 colours were generated from the MATLAB jet colour map and show a range of colours from blue at low decibels to red at high decibels [18]. These RGB images were then inputted to the Baseline CNN model with the number of channels changed to 3 to accommodate the different colours.

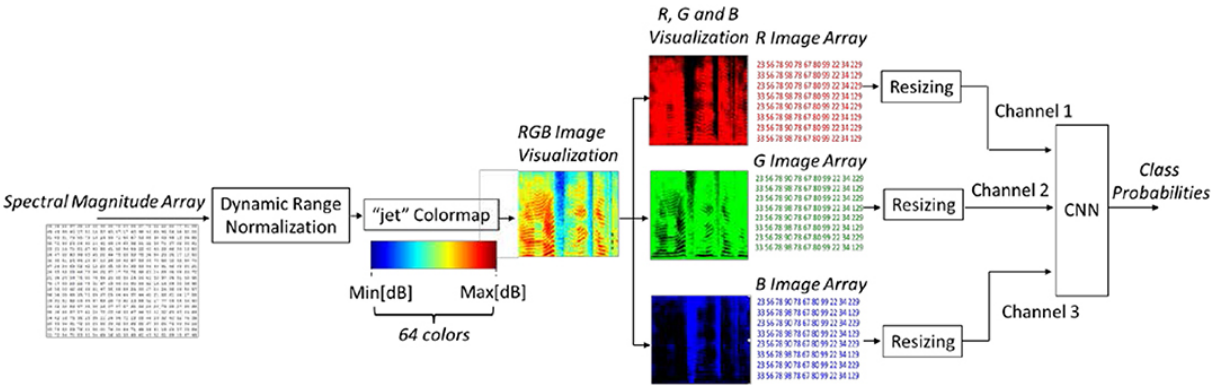


Figure 5: Framework illustrating the CNN RGB preprocessing.[6]

The 3 colour channels displayed features of the strength of each frequency in 3 different levels. As shown in Figure 6, the decomposition served as an effective method for additional feature generation, and it is robust even for noisy data. Additionally, we observed that RGB decomposition isolated the speech from the noisy data as the red image only contained higher decibel audio, which for our data completely removed the lower decibels noises.

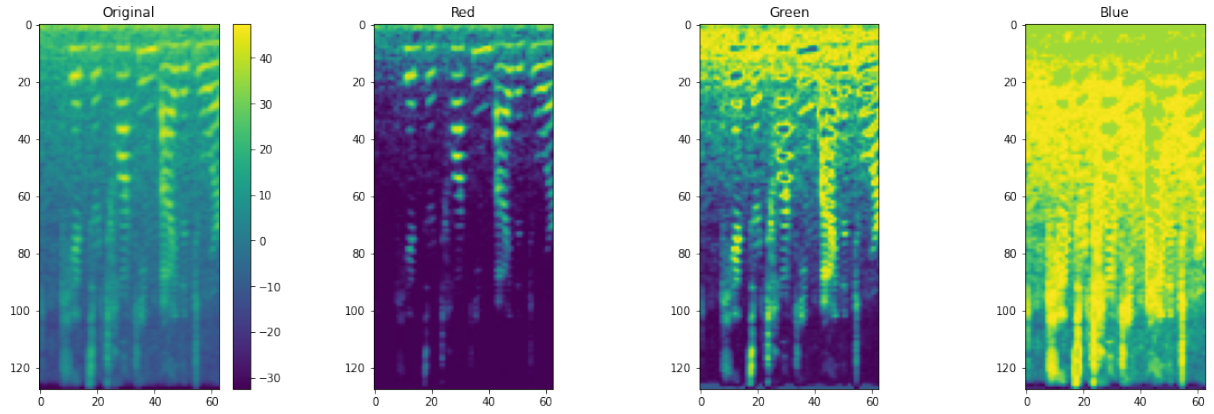


Figure 6: Mel-Sprectrogram from noisy data with RGB decomposition.

4.2.4 RGB + CNN + LSTM

We further extended the ideas of the previous two proposed modifications to the CNN to build a fully stacked model. That is, RGB decomposition was used for input transformation before the CNN architecture, then LSTM layers were stacked on top of the CNN output. The full stacked model can potentially inherit both the advantages of the previous two ideas: activation level feature extraction and time-dependency feature extraction.

The hyperparameters for this architecture is inherited from the CNN+LSTM model as it is essentially a multi-channel CNN+LSTM with 3 colour channels instead of a single mel-spectrograms. The model structure is shown in Figure 7.

Figure 7: Model strcuture for the fully stacked model (RGB + CNN + LSTM)

4.2.5 Autoencoder + CNN

Noisy speech data without doubt would directly affect context understanding and emotional identification in communication. Especially within the mel-spectrograms, where it displays all the information about frequencies and would also pick up most of the frequency activation induced by noises. Therefore, we consider a model that learns the difference between noisy information and speech information.

We consider a denoising autoencoder model for this task. Noisy mel-spectrograms are inputted into the autoencoder model, which then constructs their clean counterparts. Hence, we expect that if audio data is denoised to be as close to the clean oneas possible, then the model has successfully separated the noise from the useful information.

Furthermore, this also implies that the trained encoder has the potential to learn a more compressed feature set for the noisy data which can distinguish between noises and speech information. Leveraging this, an additional CNN model is stacked for classification based on the compressed features generated by the encoder [19].

Its structure is shown in figure 8.

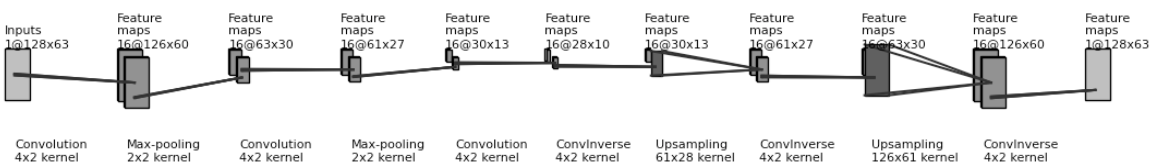


Figure 8: Autoencoder Model

4.2.5.1 Training

Training the denoising autoencoder is not as straightforward as the classification models. We have only 10 distinct speeches for each actor for all emotion classes, which in total would be 240 for all 24 actors. However, permuting these two datasets to generate all possible noisy speech samples results in each clean speech sample being contained in the permuted dataset 79 times. This means that splitting the data amongst actors as mentioned earlier is not sufficient, as the autoencoder might potentially learn speaker-dependent features from the repetitive clean speech contained in the noisy audio.

The approach to generate a balanced training/test set for a denoising purpose involves stratified sampling. For each of the 79 noise samples, uniform stratified sampling is used to select 3 unique actors, then from each selected actor, one sample from each emotion is stratified (in total 5). This idea is illustrated in Figure 9.

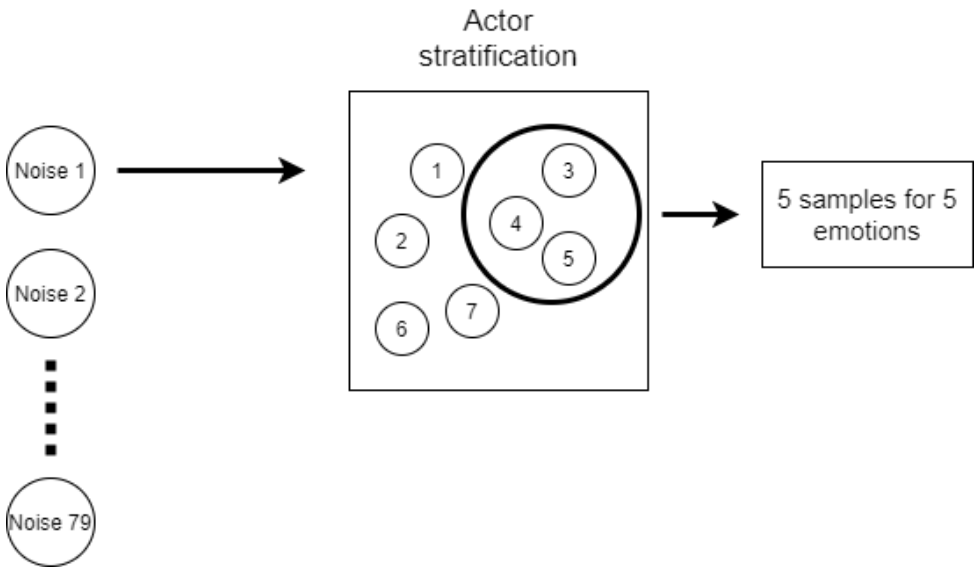


Figure 9: Noisy speech dataset permutation based on stratification sampling

After the stratified dataset is generated, train/test split with the ratio of 6:4 is used to generate a dataset for the autoencoder training. Since we are intended to use the encoder to do further classification, the second train/test split is done to the denoising test set, with a ratio of 8:2. This idea is illustrated in Figure 10

The major hyper-parameters we fine-tuned for the autoencoder is the kernel size of convolutional layers (for the same reason as stated in Baseline CNN model) and the number of filters for convolutional layers. The optimal kernel size we have found has the rectangular shape of (4, 2). The optimal number of filters for convolutional layers is 16. The visual results of the hyperparameter turning can be found in Figure 22 and 23 in the Appendix. The finalised optimal hyperparameters are listed in table below.

To check the performance of autoencoder, we only compared the pixel similarity of the denoised mel-spectrograms and the clean mel-spectrograms. In other words, the mean squared loss between the two matrices element-wise. Evaluation of the denoised audio quality was omitted because there was no efficient algorithm that can successfully reconstruct audio from

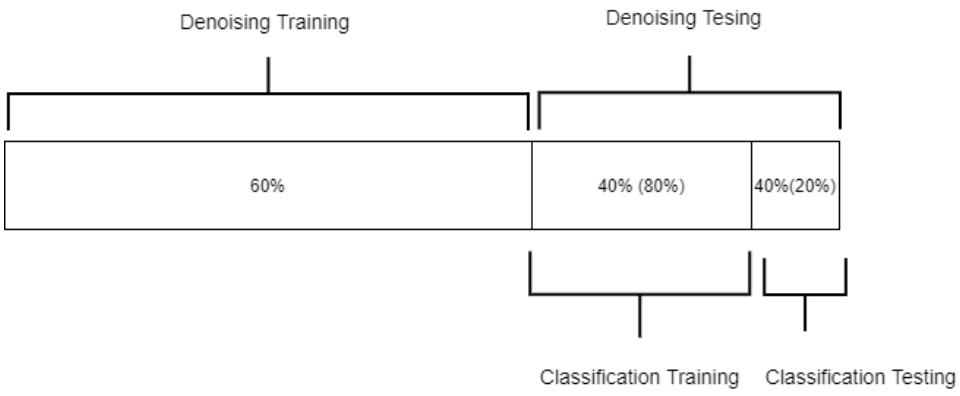


Figure 10: Diagram for Autoencoder + CNN train/test split

Hyper parameters	
Activation	ReLU
Filter number	16
Learning Rate	0.002
Batch Size	30
Optimiser	Adam
Loss Function	MSE Loss
epoch number	50

Table 3: : A summary of the hyperparameters for the Autoencoder model

mel-scaled frequencies.

Figure 11 shows the effectiveness of the denoising. The autoencoder was able to recognise the distinct features of the noise and mostly filtered out.

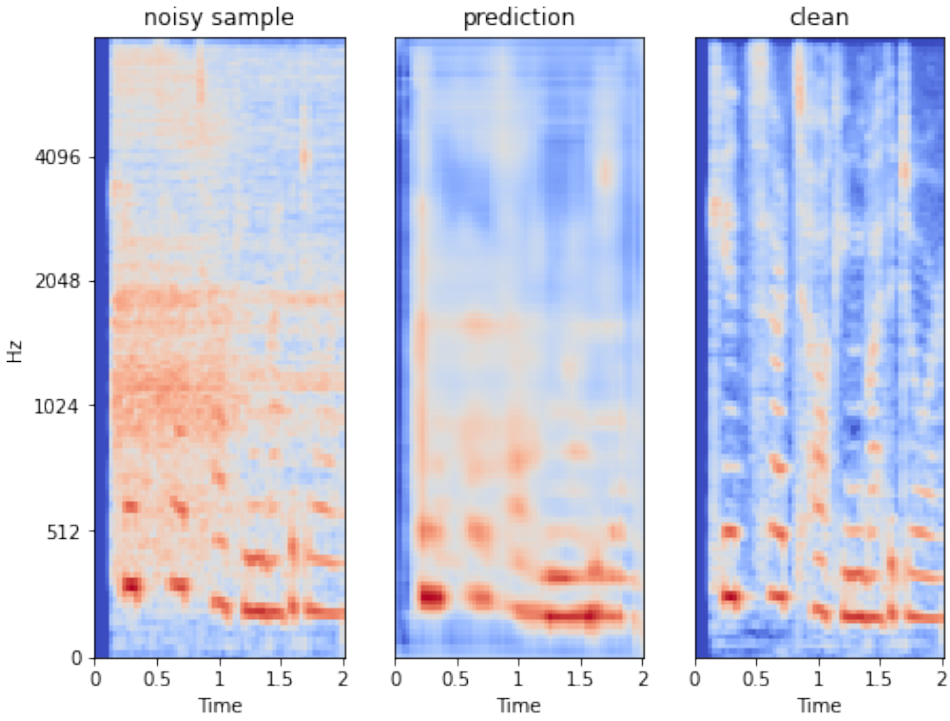


Figure 11: Noisy and denoised versions of a clean Mel-spectrogram

5 Main findings

5.1 Hidden Layer Activation

As shown in Figure 12, the baseline CNN model is able to learn various patterns presented by the input mel-spectrogram. In fact, some filters identified the low/high activation zones in the input, while the others detected horizontal/vertical features. This justified the effectiveness of CNN on spectrogram analysis with its ability for pattern detection.

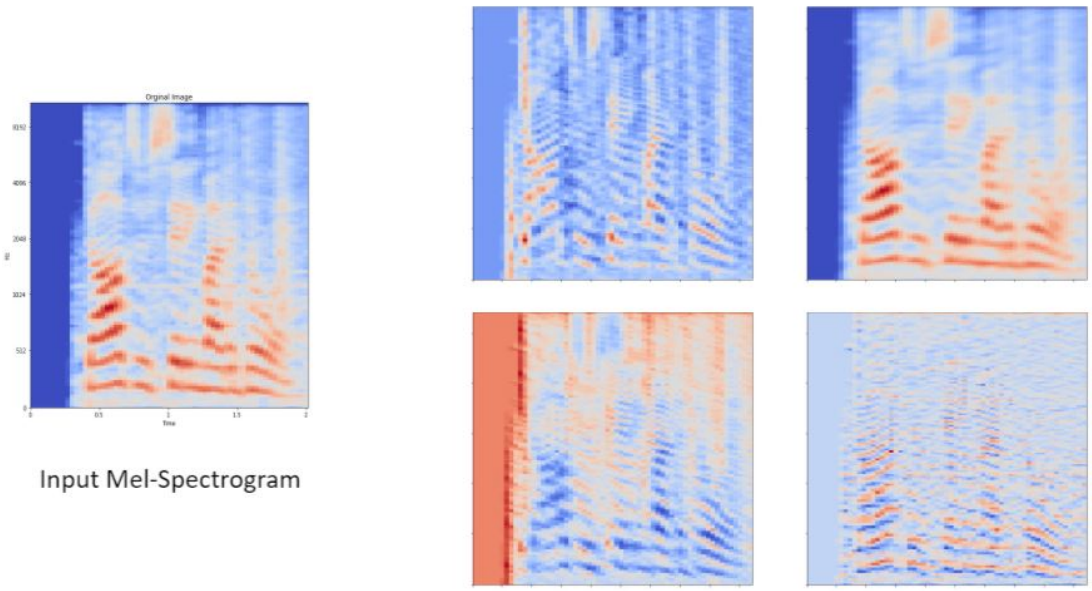


Figure 12: First hidden layer activation in baseline CNN. Vertical axis: frequency (mel). Horiozntal axis: time (sec). Reddish colour indicates high activation, vice versa for blueish colour.

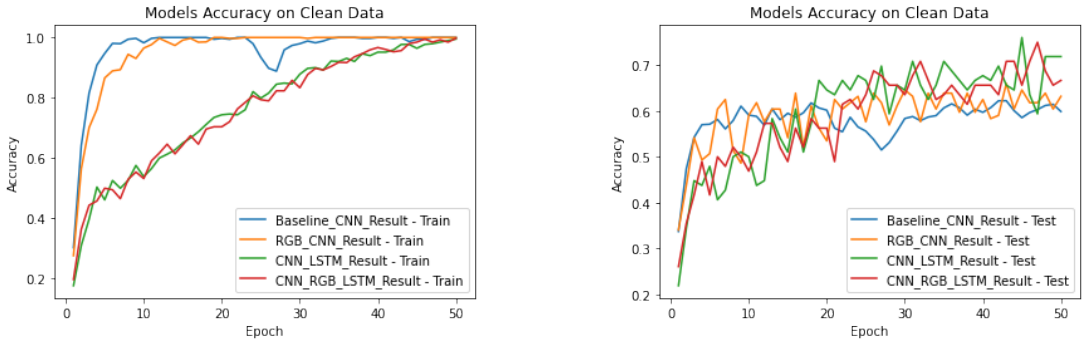
5.2 Measures of Performance

For measuring the performance of our models on the data we decided to use the metrics mentioned below.

1. Training Accuracy - The accuracy of the trained model on the same set of data it was trained on.
2. Test Accuracy - The accuracy of the trained model on a test set of that data disjoint from the training set.
3. Confusion Matrices - Matrices that displays the inter-class distribution of prediction labels comparing to the true labels.

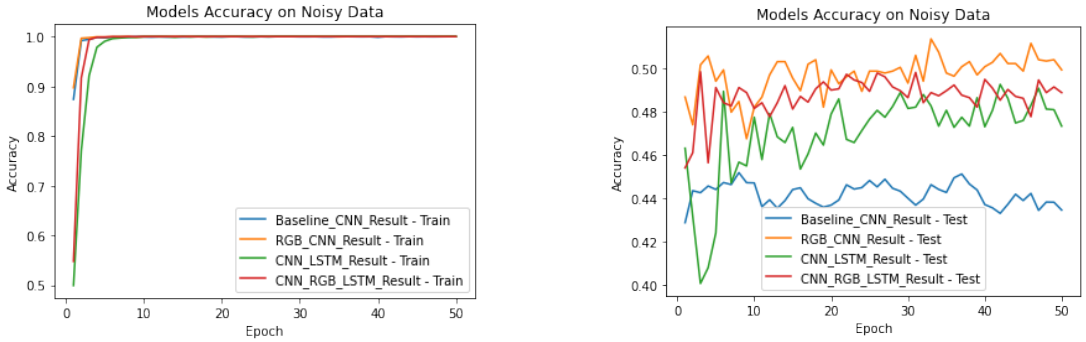
The results of training and testing all the models on the clean and the noisy data can be observed below.

5.3 Model Performance



(a) Training set accuracy of the models on clean data (b) Test set accuracy of the models on clean data

Figure 13: Plot showing the training and test accuracy of the models on clean and noisy data



(a) Training set accuracy of the models on noisy data (b) Test set accuracy of the models on noisy data

Figure 14: Plot showing the training and test accuracy of the models on clean and noisy data

Figure 13a shows the training accuracy on the clean data of the four models. The baseline CNN and the RGB model was able to reach training accuracy above 90% in 10 epoches, while both the RGB+CNN+LSTM and CNN+LSTM models took 50 epoches to reach the same level. This is expected as the CNN+LSTM architecture is by nature, more computational expensive to train. Coupling with the corresponding test accuracies shown in Figure 13b, it is clear that all 3 modifications achieved better performance than the Baseline CNN model.

In addition, the effect of the inclusion of noise on the four models was also investigated. The four models were trained and tested on noisy audio data, and their results are shown in Figure 14. While the train accuracies were seemingly increased significantly, the corresponding test accuracies were decreased by more than 15%. This phenomena persisted across all four models. One interesting thing to note is that the CNN+RGB model achieved the highest test accuracies across epoches, while its extension RGB+CNN+LSTM was second to it. This might be because CNN+LSTM was negatively affected by the presence of noises, hence impacting the overall performance. From these observation, we can conclude that CNN+RGB is the best modification of all 4 models for noisy data.

5.4 Confusion Matrices

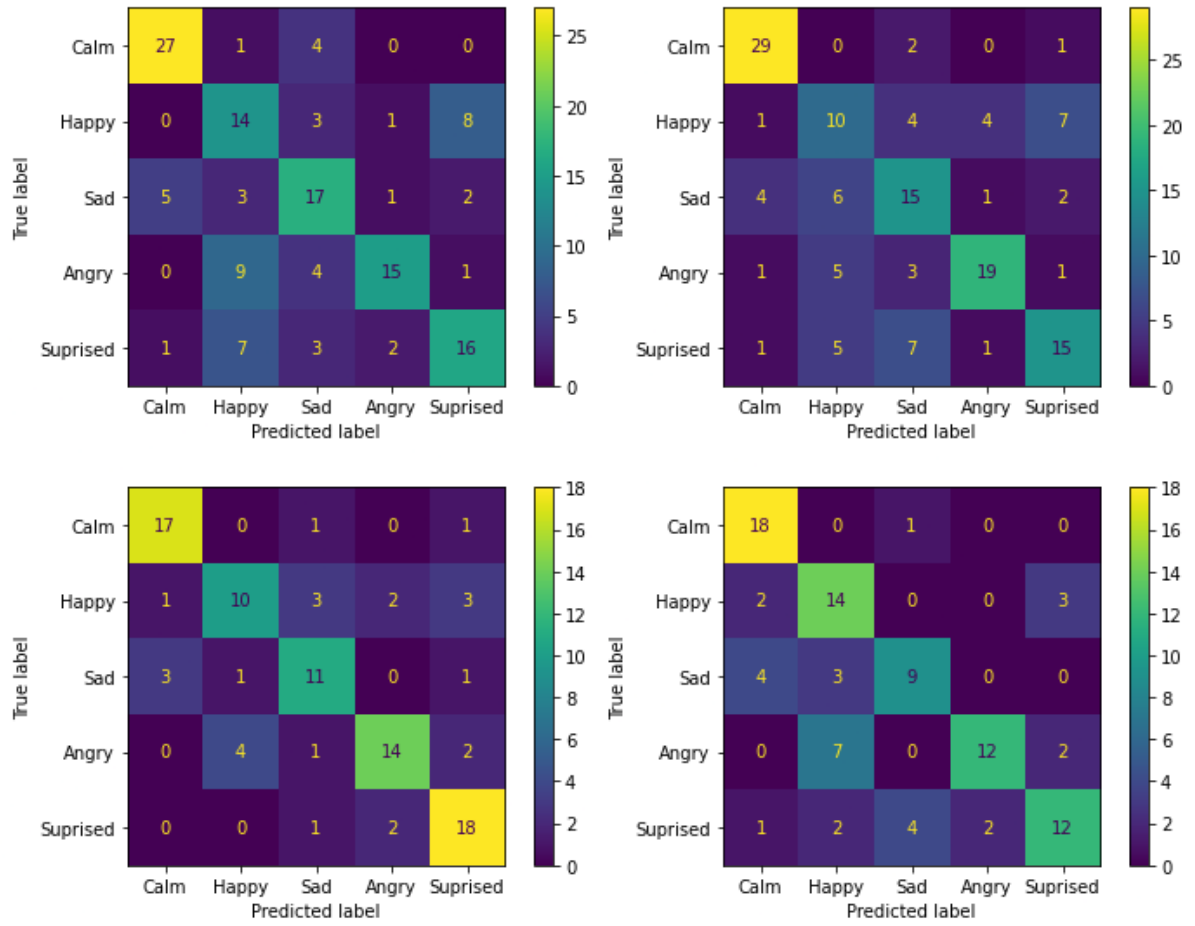


Figure 15: Clean confusion matrices
Top-Left : Baseline CNN Top-Right : CNN + RGB Bottom-Left : CNN + LSTM Bottom-Right : CNN + RGB + LSTM

Figure 15 shows the confusion matrices for the 4 models, when they are trained and tested on clean data. Observe that Baseline CNN (top left) and CNN + RGB (top right) exert some similarities to an extent, as they both have difficulties identifying emotions that are not calm. More specifically, happy is the most often misclassified emotion for both the Baseline CNN and the CNN + RGB. This suggests that these model have difficulty distinguishing between emotions with a particularly high activity, high frequency region.

Additionally, the models with LSTM architectures from Figure 15 (bottom two) suggest that introducing RNN to the CNN base model had improved the models classification accuracy on clean data. This shows that the time-dependent features can be extracted by LSTM layers, improving the performance. RGB + CNN + LSTM seems to be more prone to misclassifying the happy class, while CNN + LSTM does not show such nature, further impling the weakness of RGB decomposition outlined earlier.

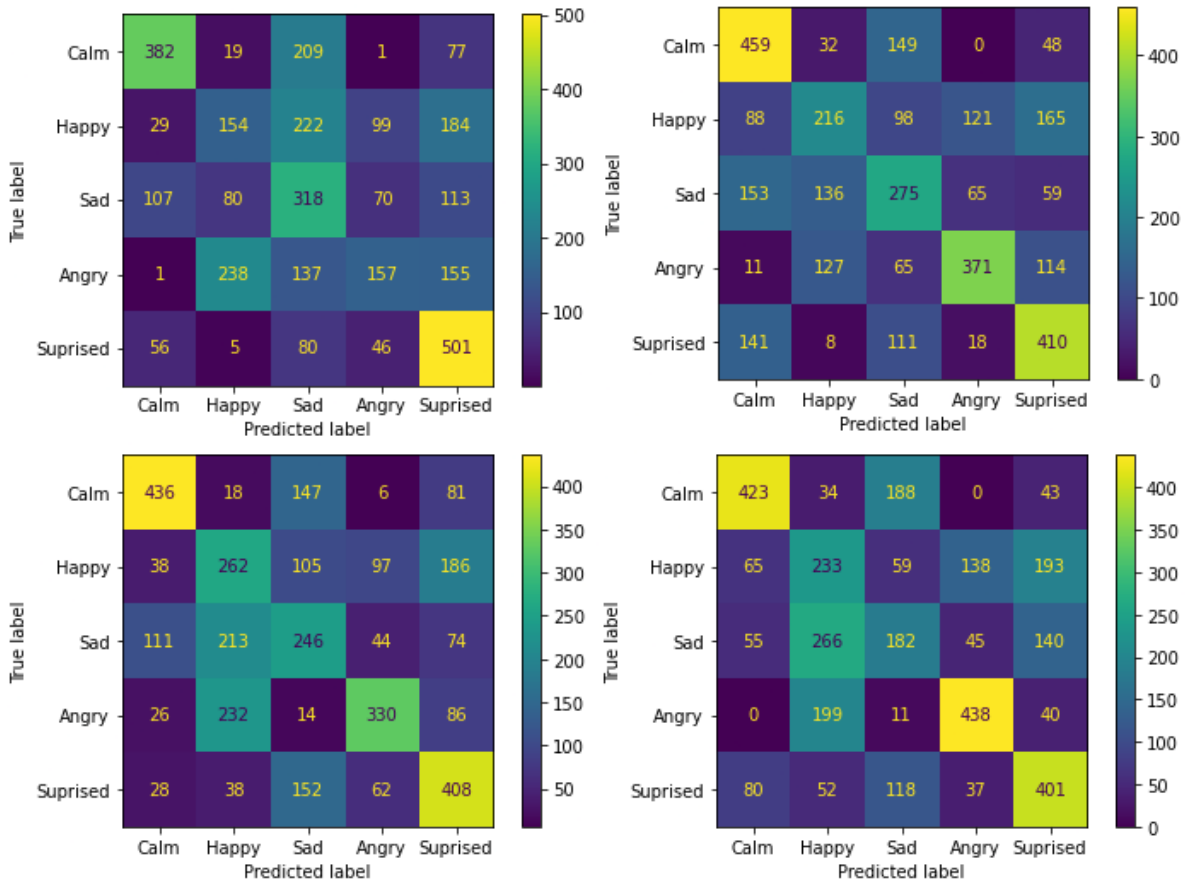
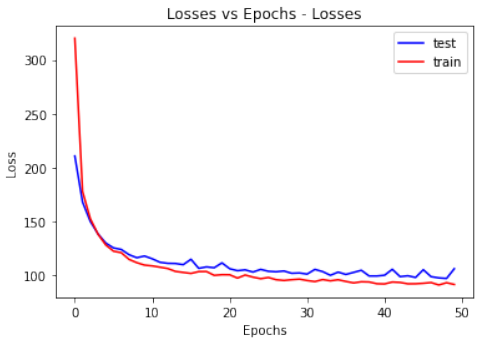


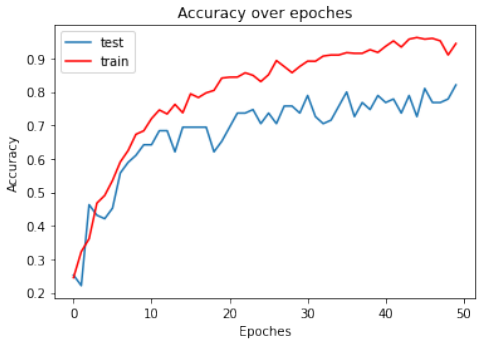
Figure 16: Noisy confusion matrices
Top-Left : Baseline CNN Top-Right : CNN + RGB Bottom-Left : CNN + LSTM Bottom-Right : CNN + RGB + LSTM

Figure 16 shows the confusion matrices for the 4 models, when they are trained and tested on noisy data. This figure confirms that our models have difficulty classifying happy, sad and angry, but are proficient in classifying calm and surprised samples. This may suggest that our models are only able classify speech if it is very high energy or very low energy emotions, and all emotions that exist in between are more difficult to classify. This can be attributed to those emotions having weaker features in the activation layers as they tend to exist over a wide range of frequencies and thus are difficult to classify with noise obscuring most of that region.

5.5 Best Model



(a) Losses of autoencoder during over epoches



(b) Encoder+CNN accuracy over epoches

Figure 17: Plot showing the training and test accuracy of the models on clean and noisy data

Noise interference was the major issue for the above models, despite the effect of the modifications. As a result, a denoising autoencoder is included to reconstruct clean mel-spectrograms from their clean counterpart. As seen in Figure 17a, the model is improving gradually with minimal signs of over-fitting.

Using the trained Encoder with a stacked CNN for classification yielded improvement in performance in noisy data (Figure 17b). While the previous 4 models were only able to reach a maximum accuracy of around 55%, this architecture was able to attain 70%. This showed that the compressed feature set, generated by the Encoder distinguishes some difference between noisy frequency information and useful frequency information.

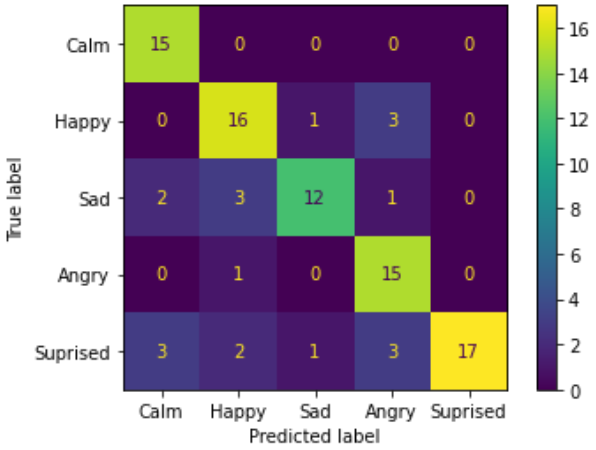


Figure 18: Confusion matrix for the Autoencoder + CNN model on noisy data

Finally, we have the confusion matrix for the Autoencoder + CNN model trained and tested on noisy data. Interestingly, the Autoencoder + CNN model, unlike the models shown above is able to correctly identify middle intensity emotions well. This further suggests that the Encoder is able to distinguish some difference between noisy frequency information and useful frequency information.

6 Limitations of the study and Future work

The main limitation present in our model is the small variation in the speech and noise data available. As we only have two speech phrases and 13 unique noise types, we can not be certain that our model generalises to different types of noises or general speech. To remedy this limitation, future studies can be focused on looking into using generative noise models that are able to general any type of noise and overlay that noise over general speech data[20].

Building on this limitation in the range of the data provided, we have yet to apply our model to speech from languages other then English. To account for this in future studies we will need to add similar speech from a variety of different languages, while also ensuring that there is no class imbalance in the multilingual data.

Additionally, while we believe that our autoencoder is correctly denoising the noisy data, we have only been able to observe the change in the mel-spectrogram. This is because short time Fourier transform with mel-scale, which is used in the construction of the mel-spectrogram, is not inverseable without significant lose. This means that we are unable to listen to the denoised audio data for comparison and thus unable to confirm if the audio reconstruction is working. In future we will need to further explore denoising techniques and potential find a method that allows for proper evaluation of the performance on the denoised data.

While the autoencoder has shown itself to be the best model in regards to the classification of noisy audio data, we have yet to investigate the impact of integrating RGB decomposition into the autoencoder model. In future studies, implementing this along with further tuning the models through investigating the effect of a different number of neurons in the linear layer, could result in a potentially even better performance from our models.

Finally, since we have been able to decompose our data into effectively an RGB image there is the possibility of using common computer vision techniques for classifying our data. In future studies, transfer learning with pre-trained image classification models, such as AlexNet, should be investigated.

7 Conclusion

In conclusion, we have investigated the performance of CNN models with modifications on emotion classification based on audio data. Our finding suggested that CNN models can generally identify mel-spectrogram activation features. We also found that concatenating LSTM layers after a CNN model was able to detect time-dependency features and achieved the highest accuracy on clean data, however, it was prone to noise inference. RGB decomposition was able to increase the robustness against noises but with the consequence of decreasing clean data performance. Overall, all 3 modifications suffered a reduction of accuracy by relatively the same amount against noisy data.

A denoising autoencoder was shown to be able to mask out a portion of the noisy frequencies. Using the trained encoder, a compressed feature set, based on the mel-spectrograms, was generated. Classifying emotions using a multi-channel Baseline CNN on such compressed

features was able to achieve an increased accuracy to around 70%, which is a significant improvement from the modifications above.

Finally, a more thoughtful implementation and integration of denoising techniques should be investigated in the future as it was seen to show promising results. Additionally, RGB decomposition coupled with image classification transfer learning is also a suggested extension of this project.

References

[1] North Shore Speech Therapy. *Language Comprehension*. 2012.

[2] Henrik Nordström. "Emotional Communication in the Human Voice". 2019. ISBN: 978-91-7797-735-0.

[3] Babak Joze Abbaschian, Daniel Sierra-Sosa, and Adel Elmaghraby. "Deep Learning Techniques for Speech Emotion Recognition, from Databases to Models". In: *Sensors* 21.4 (2021). ISSN: 1424-8220. DOI: 10.3390/s21041249. URL: https://www.mdpi.com/1424-8220/21/4/1249.

[4] AWS. *Contact Lens for Amazon Connect*. 2020.

[5] Fei Ma et al. "Learning Better Representations for Audio-Visual Emotion Recognition with Common Information". In: *Applied Sciences* 10.20 (2020). ISSN: 2076-3417. DOI: 10.3390/app10207239. URL: https://www.mdpi.com/2076-3417/10/20/7239.

[6] Margaret lech et al. "Real-Time Speech Emotion Recognition Using a Pre-trained Image Classification Network: Effects of Bandwidth Reduction and Companding". In: *Front. Comput. Sci* (2020). DOI: 10.3389/fcomp.2020.00014. URL: https://doi.org/10.3389/fcomp.2020.00014.

[7] Alex Krizhevsky, Ilya Sutskever, and E Geoffrey Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* (2012).

[8] Steven R. Livingstone and Frank A. Russo. "The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English". In: *PLOS ONE* 13.5 (May 2018), pp. 1–35. DOI: 10.1371/journal.pone.0196391. URL: https://doi.org/10.1371/journal.pone.0196391.

[9] Chandan KA Reddy et al. "A Scalable Noisy Speech Dataset and Online Subjective Test Framework". In: *Proc. Interspeech 2019* (2019), pp. 1816–1820.

[10] Mandeep Singh and Yuan Fang. *Emotion Recognition in Audio and Video Using Deep Neural Networks*. 2020. arXiv: 2006.08129 [eess.AS].

[11] Mahidhar Dwarampudi and N. V. Subba Reddy. "Effects of padding on LSTMs and CNNs". In: *CoRR* abs/1903.07288 (2019). arXiv: 1903.07288. URL: http://arxiv.org/abs/1903.07288.

[12] D. O'Shaughnessy. *Speech Communication: Human and Machine*. Addison-Wesley series in electrical engineering. Addison-Wesley Publishing Company, 1987. ISBN: 9780201165203. URL: https://books.google.com.au/books?id=mHFQAAAAMAAJ.

[13] S. S. Stevens, J. Volkmann, and E. B. Newman. "A Scale for the Measurement of the Psychological Magnitude Pitch". In: *The Journal of the Acoustical Society of America* 8.3 (1937), pp. 185–190. DOI: 10.1121/1.1915893. eprint: https://doi.org/10.1121/1.1915893. URL: https://doi.org/10.1121/1.1915893.

[14] Zhengwei Huang et al. "Speech Emotion Recognition Using CNN". In: *Proceedings of the 22nd ACM International Conference on Multimedia*. MM '14. Orlando, Florida, USA: Association for Computing Machinery, 2014, pp. 801–804. ISBN: 9781450330633. DOI: 10.1145/2647868.2654984. URL: https://doi.org/10.1145/2647868.2654984.

[15] Abdul Malik Badshah et al. "Deep features-based speech emotion recognition for smart affective services". In: *Multimedia Tools and Applications* 78.5 (Mar. 2019), pp. 5571–5589. ISSN: 1573-7721. DOI: 10.1007/s11042-017-5292-7. URL: https://doi.org/10.1007/s11042-017-5292-7.

[16] Jeff Donahue et al. "Long-term recurrent convolutional networks for visual recognition and description". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 2625–2634. DOI: 10.1109/CVPR.2015.7298878.

[17] Wootaek Lim, Daeyoung Jang, and Taejin Lee. "Speech emotion recognition using convolutional and Recurrent Neural Networks". In: *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. 2016, pp. 1–4. DOI: 10.1109/APSIPA.2016.7820699.

[18] MathWorks. *Documentation Jet, Jet Colormap Array*. 2018. URL: https://au.mathworks.com/help/matlab/ref/jet.html?requestedDomain=www.mathworks.com.

[19] Pascal Vincent et al. "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". In: *J. Mach. Learn. Res.* 11 (Dec. 2010), pp. 3371–3408. ISSN: 1532-4435.

[20] Upasana Tiwari et al. "Multi-Conditioning and Data Augmentation Using Generative Noise Model for Speech Emotion Recognition in Noisy Conditions". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 7194–7198. DOI: 10.1109/ICASSP40776.2020.9053581.

Appendices

Baseline CNN Hyper Parameters Tuning

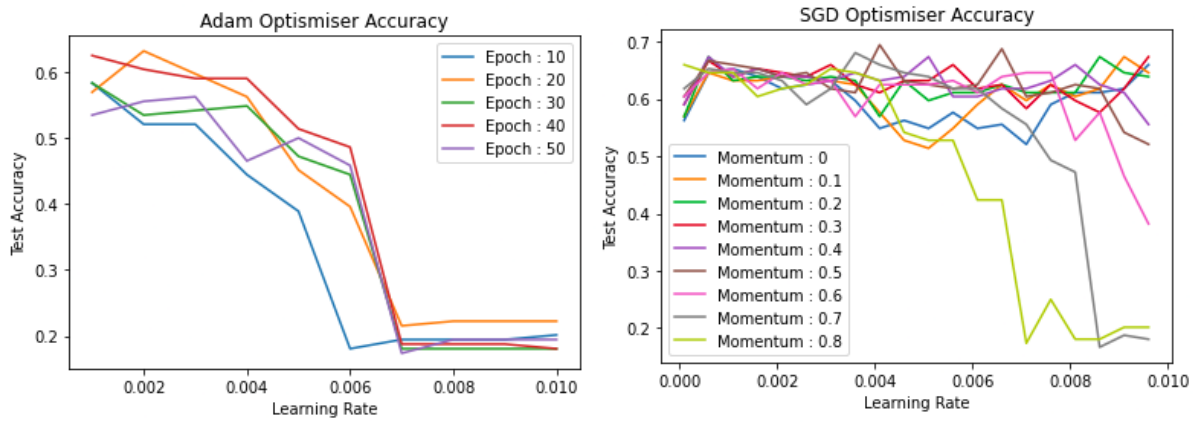


Figure 19: Hyperparameter tuning for Baseline CNN

CNN + LSTM

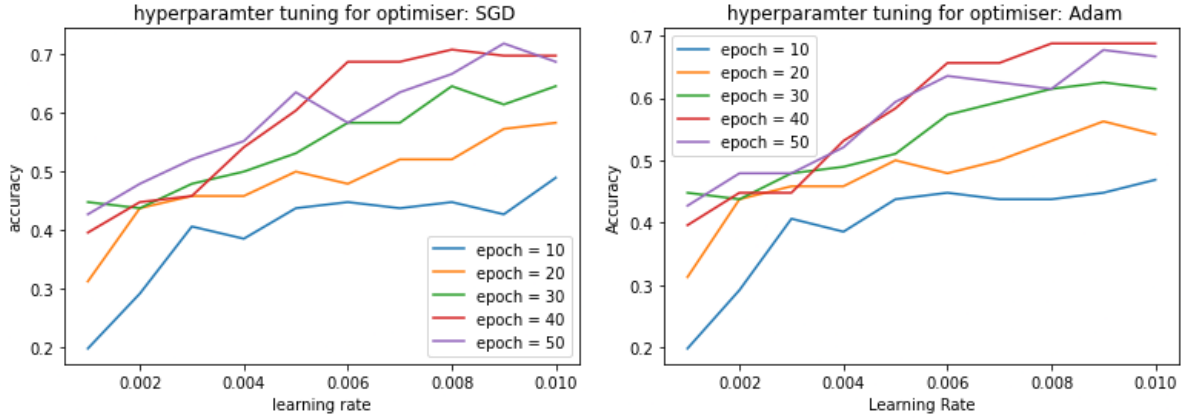


Figure 20: CNN + LSTM learning rate and epoch hyperparameter tuning with
Left : SGD Right : Adam

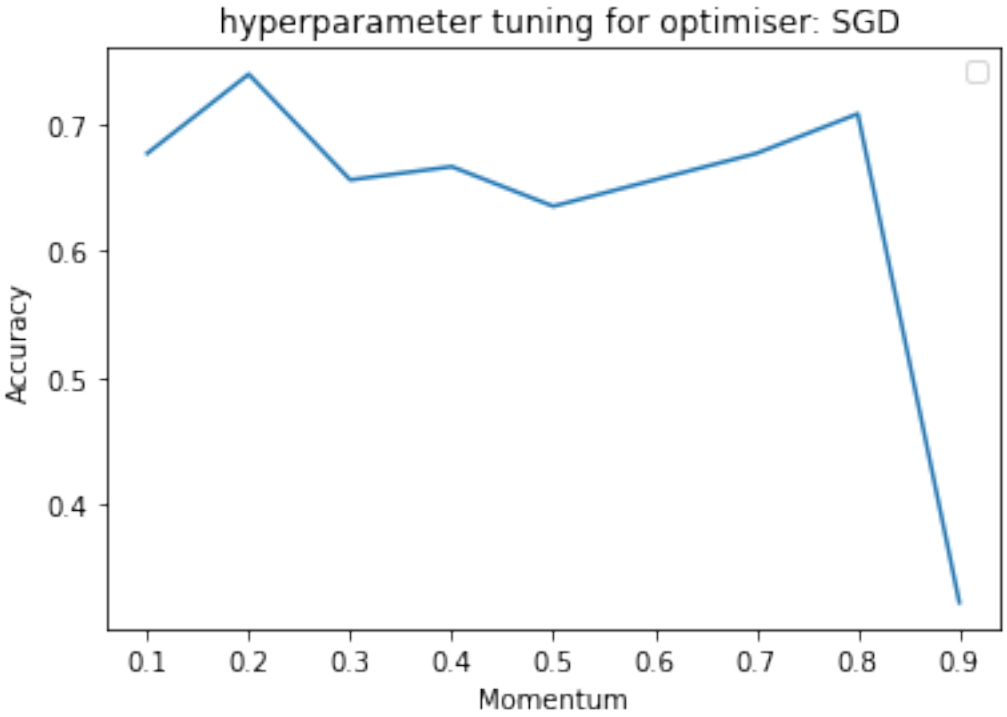
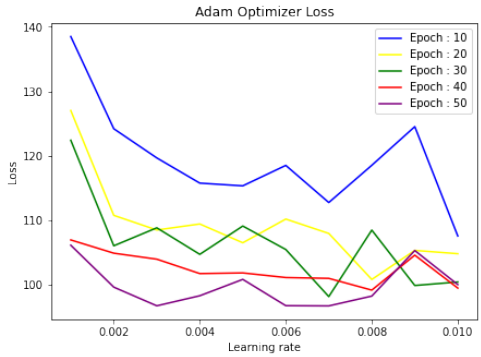
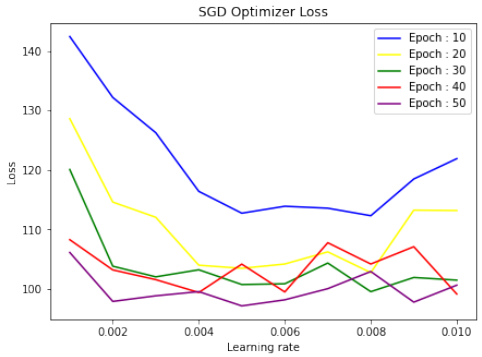


Figure 21: CNN + LSTM Momentum tuning

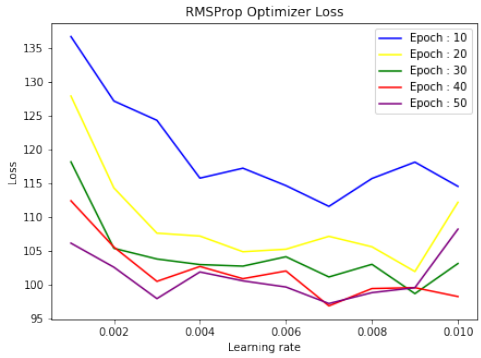
Autoencoder + CNN



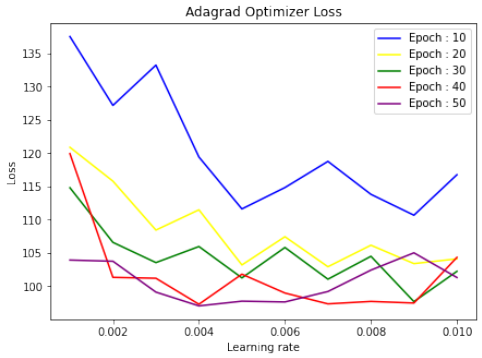
(a) Adam optimiser turning



(b) SGD optimiser turning



(c) RMSProp optimiser turning



(d) Adagrad optimizer turning

Figure 22: Autoencoder hyperparameters turning for optimszer, learning rate and epoch

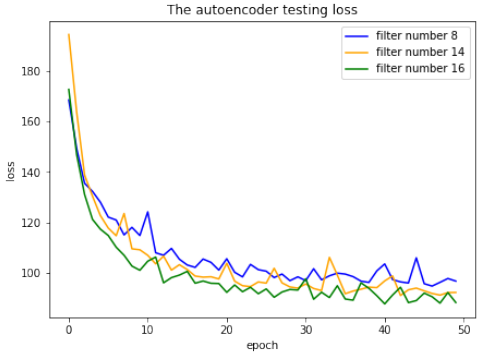


Figure 23: Autoencoder hyperparameters turning for the number of filters in the convolution layers

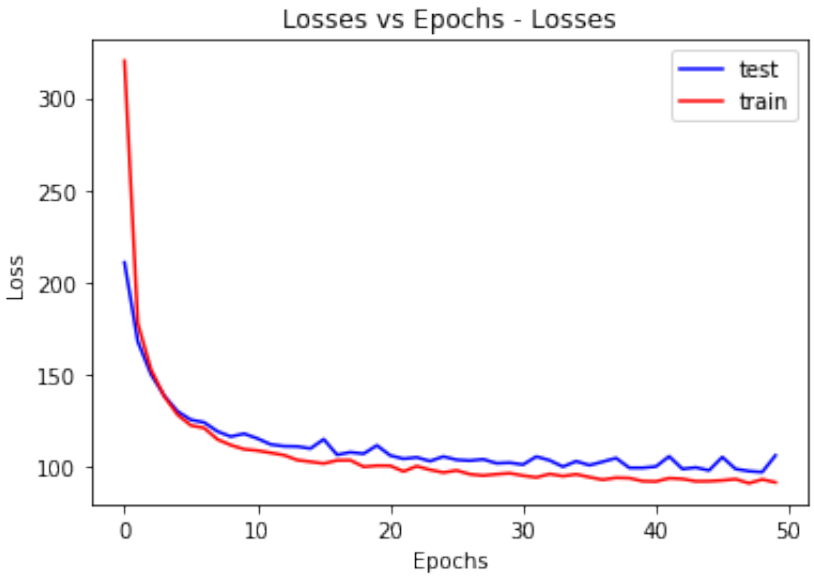


Figure 24: Loss graph of autoencoder with optimal hyper-parameters