Benha University
Faculty of Engineering at Shoubra
Electrical Engineering Department
Computer Systems Engineering
Academic Year 2021/2022

Audio-Visual Emotion Recognition System

A graduation project report submitted in partial fulfillment of the requirements for the degree of BSC in Computer Systems Engineering

Submitted By:
Mahmoud Mohamed Abdelwahab
Mohamed Ramadan Abo Ela
Amr Kamal Fouad
Marwa Adel
Mariam Maher

Supervised By:
Dr: Eman Abd-El Ghafar

Table of Contents

List of figures

List of Table

Acknowledgement

First of all we would like to express our gratitude to our advisor **Dr. Eman Abd-El Ghafar** who has been of great help throughout these last months, supporting us along the way with any question or problem we have had. Without her help this process would have been much harder and much more heavy-handed. Her support has kept us motivated through difficult times. For all this We are grateful not only for her technical help but also for her calm and excellent attention at all times. Thank you.

ABSTRACT

Emotions are the gateway to communicate one's viewpoint to others. Thus recognizing the emotion becomes one of the important aspects. Humans can communicate their emotions by their audio and facial expressions .This project presents a multimodal emotion recognition system, which is based on the analysis of audio and visual cues. It mainly aims to classify at least 4 or 6 emotions namely sad, happy, anger, neutral, calm and fear.From the audio channel, Mel-Frequency Cepstral Coefficients features are extracted. For the visual part, two strategies are considered. First, facial landmarks' geometric relations, i.e. distances and angles, are computed. Second, we summarize each emotional video into a reduced set of key-frames.In order to do so, a convolutional neural network is applied to key-frames summarizing videos. . Once both models have been trained separately, they will be combined together into a new model, since fusing the results of audio emotion recognition and facial expression recognition to varying degrees can be helpful for audio-visual emotion recognition .The experiments conducted on the RAVDESS and SAVEE datasets show significant performance improvements.

Chapter 1

1.1 Introduction

Emotions are difficult to recognize because they depend on context, facial expressions, speech and gestures. The latest technologies based on data-centric techniques have come a long way, but modern day machine learning algorithms is still a very difficult task despite recent efforts to augment artificial intelligence techniques to consider context. For this reason, people who are working with emotion recognition tend to work with individual aspects such as identifying a person's emotions based purely on looking at their faces, through speech, or through the vocabulary used. "If there is so much room for error when it comes to reading a person's facial expressions, we must question how a machine can ever be programmed to get it right."

Today, Emotion Recognition Systems can help firms vet job applicants, analyze the emotional impact advertisements have on people as well as helping police detect criminal suspects

Many systems are based on the work of the psychologist Paul Ekman (Ekman, 2011) who proposed seven basic emotions which he then changed to six that are recognized around the world: happiness, sadness, fear, anger, surprise and disgust. His research took him to the conclusion that the recognition of emotions was similar across cultures, and thus universal.

The purpose of this project is to try to recognize emotions through the combination of two techniques based on Paul Elkman's study of the six basic emotions. Since the correlation between facial expression and emotion has been questioned due to the fact that different cultures have varying expectations for when it is appropriate to use a certain expression and in what manner, it was decided to add emotion detection through audio in order to have two different angles.

Taking all the above into account, the project will plan out in the following way: First, an emotion detection model through audio features will be trained by trying different combinations of datasets and neural networks to try and achieve the highest possible accuracy. Then the same will be done with video features (facial expressions). Once both models have been trained separately, they will be combined together into a new model in order to see whether a greater accuracy percentage is achieved together rather than separately.

1.2 Motivation

The main motivation to carry out this project is to learn new programming techniques as well as expanding new horizons in a field we have become increasingly fascinated with machine learning. What we wish to obtain when developing this project is a good accuracy in emotion detection through video and audio separately and a better result when combining the two models together. In theory a model that recognizes emotions through video and audio at the same time should learn better and therefore be more accurate than two individual models.

1.3 Aim

The main aim of our project is to Help computers differentiate between six different emotions using audio-visual data.

1.4 Objectives

- Focusing on related work in the area of video-audio emotion recognition.
- Determining suitable feature extraction methods for audio files.
- Apply Facial landmark recognition for video.
- Investigating the most suitable classification model for audio and video emotion recognition.
- Perform fusion between two classification models (audio & video).

1.5 Project Scope

Machine learning /deep learning system for emotion recognition from video and sound on video-audio dataset to help in detecting human emotion by video and sound by high efficiency.

1.6 Methodology

- **Planning and Literature review:** Reading papers related to emotion recognition from video, sound and fusion multimodal to build background about this field
- **Explore datasets:** Searching for video-sound datasets or implementing our own dataset.
- **Structuring Audio Models, techniques and testing:** know more about most used models, algorithms and techniques for audio then Comparing between available techniques and methods to choose the most appropriate among them then implementing audio model and testing
- **Structuring Video Models, techniques and testing:** know more about most used models, algorithms and techniques for video then Comparing between available techniques and methods to choose the most appropriate among them then implementing video model and testing
- **Structuring Fusion Models, techniques and testing:** know more about most used models, algorithms and techniques for fusion then Comparing between available techniques and methods to choose the most appropriate among them then implementing fusion model and testing
- **Testing system performance:** Testing different methods and making comparisons between the results.

1.7 Project Plan

It was critical to understand the phases we would go through in our project. We obviously wanted to know about related work and gather a literature review first, so we read academic papers relevant to our research. Then we searched for datasets that were available. Then, based on our tests with available datasets linked to our project, we select the dataset. Then we moved on to the preprocessing phase, after that we began experimenting with the proposed models and approaches, collecting data and making comparisons.

This estimated project plan defines project goals and objectives, specifies tasks and also timelines for completion.

Table 1-1 : Project plan phases

Phases	Duration	Start Date	End data
Planning and Literature review	30 days	9/10/2021	7/11/2021
Specify the problem description and solution.	7 days	9/10/2021	15/10/2021
Read the most recent research related to our project.	10 days	16/10/2021	25/10/2021
Generate a summary of the applied models in related work.	3 days	26/10/2021	28/10/2021
Read the most recent researches related to our first model (Audio)	10 days	29/10/2021	7/11/2021
Dataset	7 days	8/11/2021	14/11/2021
Search and download available datasets for our project	7 days	8/11/2021	14/11/2021
Structuring Audio Models, techniques and testing	48 days	15/11/2021	1/1/2022
Run Different audio model based on MFCC feature extraction method	10 days (parallel)	15/11/2021	24/11/2021
Run Different audio model based on spectrogram feature extraction method	10 days (parallel)	15/11/2021	24/11/2021
Midterm (break)	9 days	25/11/2021	3/12/2021
Run Different audio model based on MFCC feature extraction method	26 days (parallel)	4/12/2021	29/12/2021
Run Different audio model based on spectrogram feature extraction method	26 days (parallel)	4/12/2021	29/12/2021
Write an accuracy comparison table between two feature extraction methods	3 days	30/12/2021	1/1/2022
Reading Related work of Video	6 days	2/1/2022	7/1/2022
Read the most recent researches related to our second model (video)	6 days	2/1/2022	7/1/2022

FINAL First term (break)	28 days	8/1/2022	6/2/2022
Documenting introduction and Related work	11 days	7/2/2022	17/2/2022
Implementing Audio Model and Testing	15 days	18/2/2022	4/3/2022
Training Audio Models	12 days	18/2/2022	1/3/2022
Collecting and comparing results	3 days	2/3/2022	4/3/2022
Seminar	2 days	5/3/2022	6/3/2022
Structuring Video Models, techniques and testing	37 days	7/3/2022	12/4/2022
Training and Testing different Models with different techniques	15 days	7/3/2022	21/3/2022
Implementing Video Model and Testing	9 days	22/3/2022	30/3/2022
Midterm (break)	10 days	31/3/2022	9/4/2022
Collecting and comparing results	3 days	10/4/2022	12/4/2022
Structuring Fusion Models, techniques and testing	55 days	13/4/2022	6/6/2022
Training and Testing different Models with different techniques	12 days	13/4/2022	24/4/2022
Implementing Fusion Model and Testing	6 days	25/4/2022	30/4/2022
FINAL Second term (break)	27 days	1/5/2022	27/5/2022
Implementing Fusion Model and Testing	7 days	28/5/2022	3/6/2022
Collecting and comparing results	3 days	4/6/2022	6/6/2022
Final Documenting	10 days	7/6/2022	16/6/2022
Final presentation preparation	3 days	17/6/2022	19/6/2022

Chapter 2 **(Related Work)**

2.1 Introduction

We go over a background and some principles in Performance Metrics and classification and some related work studies. Emotion recognition can be used in different applications like companies that want to improve customer experience, helping blind people to read facial expressions and helping robots interact more intelligently with people and analyze their emotions.

2.2 Background

Emotion recognition is an important research field for Human-Computer Interaction. Emotion recognition has many applications. For example, instead of filling out a lengthy survey about how you feel at each point watching an educational video or advertisement, you can consent to have a camera watch your face and listen to what you say. Other uses include helping children with autism, helping people who are blind to read facial expressions, helping robots interact more intelligently with people, and monitoring signs of attention while driving in an effort to enhance driver safety. Audio-Video Emotion Recognition is now attacked with Deep Neural Network modeling tools. Some papers introduce cases of the superiority in multi-modality over audio-only or video-only modality and other studies introduce models of fusion both audio and video.

2.3 Feature Extraction and Data Preprocessing

We will explain how the audio files have been processed as well as the video files and the procedure that has been followed to extract the desired features.

2.3.1 Data Preprocessing

Data preprocessing is a crucial step that helps enhance the quality of data to promote the extraction of meaningful insights from the data. Data preprocessing refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models [1] [2].

Audio signal has to be preprocessed before the features extraction step which will be mentioned in this section. Preprocessing includes making segmentations of audio files, resampling their sample rate or mixing it with other datasets. Making segmentations of an audio file means dividing it into chunks to have more data to train on or to know the effect of signal time on results. [1]

Video preprocessing techniques are required to reduce the complexity of the succeeding steps consisting of video text detection, localization, segmentation, recognition, and script identification. [2]

2.3.2 Feature Extraction

Audio feature extraction is a necessary step in audio signal processing. It removes unwanted noise and balances the time-frequency ranges by converting digital and analog signals. to describe audio, extract more than one feature. There are several ways of extracting features from an audio file such as Mel-frequency cepstral coefficients (MFCC) and Mel spectrograms (Mel Spectrogram) are commonly used in speech recognition and speaker recognition systems. [1]

Video Feature extraction means to find out the "point of interest" or differentiating frames of video. These features are consistent over several video frames of the same scene and after the video scene is changed the value of these features are changed. all the frames from each video are processed independently. These undergo two main steps, the first one consists of detecting the facial landmarks of the face and the second one consists of aligning the face. Therefore we can use features to classify videos[3].

MFCC

The set of Mel-Frequency Cepstral Coefficients is a cepstral representation of the audio signal obtained based on the Mel-scaled spectrum [4]. The most used MFCCs consists of between 13 to 45 coefficients.

Mel Spectrogram

A spectrogram is an image representation of the waveform signal, it shows its frequency intensity range over time, it can be very useful when evaluating the signal's frequency distribution over time. Mel-Spectrogram allows plotting amplitude on frequency vs time graph on a Mel scale [5]. Spectrogram are commonly used to display frequencies of sound waves produced by humans, animals, etc as recorded by microphones. Spectrograms are used extensively in the fields of sonar, radar, speech processing and can be used to identify spoken words phonetically.
A spectrogram can be generated by an optical spectrometer, a bank of band-pass filters, by Fourier transform or by a wavelet transform in which case it is also known as a scalogram. A spectrogram is usually depicted as a heat map, i.e., as an image with the intensity shown by varying the color or brightness [6].

Facial Landmarks

Used to localize and represent specific parts of the face by using shape prediction methods. Detecting facial landmarks can be divided into two steps: Localizing the face in the image and detecting the key facial landmarks on the region of interest. OpenCV is used to detect facial landmarks in an image [3]

Aligning The Face

Face alignment consists in identifying the geometric structure of faces in digital images and obtaining a canonical alignment of the face based on translation, scale, and rotation. [3]. [7]

2.4 Performance Metrics

Once the model has been trained, the accuracy and loss graphs are plotted. The loss learning curve measures our model error, or "how bad our model is doing", therefore, the lower the loss becomes, the better the model's performance will be. The accuracy learning curve captures the model's performance, so the higher it is, the better the model becomes.

2.4.1 Accuracy

It is the simplest metric to use and implement and is defined as the number of correct predictions divided by the total number of predictions, multiplied by 100. Can be implemented by comparing ground truth and predicted values in a loop or simply by utilizing the scikit-learn module to do the heavy lifting.

For example, Start by just importing the *accuracy_score* function from the *metrics* class. Then, just by passing the ground truth and predicted values we can determine the accuracy of our model [7].

2.4.2 Confusion Matrix

It is a tabular visualization of the ground-truth labels versus model predictions. Each row of the confusion matrix represents the instances in a predicted class and each column represents the instances in an actual class[7].

2.5 Audio Visual Emotion Recognition Dataset

An early step of the project was to carry out a thorough research of audio-visual datasets and gather enough information on them for afterwards being able to select those most suited for the problems to solve. Two of the most used datasets were chosen.

Table 2-1 : Show Datasets details

RESEARCHED DATASETS					
Dataset	**Nature**	**Emotions**	**Records**	**Actors**	**Emotion Levels**
SAVEE	Audio-Visual	7	480	4 actors	1
RAVDESS	Audio-Visual	8	2880	24 actors (12 male, 12 female)	2

2.5.1 RAVDESS

The Ryerson Audio-Visual Database of Emotional Speech and Songs a validated multimodal database of emotional speech and song. The database is gender balanced consisting of 24 professional actors (12 male,12 female), vocalize two statements in a North American accent. Speech includes calm, happy, sad, angry, fearful, surprise, and disgust expressions, and song contains calm, happy, sad, angry, and fearful emotions Each one of these expressions is produced at two levels of emotional intensity (normal and strong), with an additional neutral expression and can be obtained in three different formats: Audio-only, Audio-Video and Video-only. This dataset was downloaded in both Audio only and Audio-Video formats; one for emotion recognition through audio and the other for emotion recognition through video [8].

The speech dataset used contains a total of 1440 files (60 trials x 24 actors) for the eight emotions mentioned above. While the song dataset contains a total of 1056 files (44 trials x 24 actors) for the five emotions. Each of the files has a unique filename consisting of a 7-part numerical identifier which define the different characteristics:

Figure 2-1 : RAVDESS actor examples (source: (LIVINGSTONE & RUSSO ,2018))

- Modality (01 = full-AV, 02 = video-only, 03 = audio-only).
- Vocal channel (01 = speech, 02 = song).
- Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
- Emotional intensity (01 = normal, 02 = strong). NOTE: There is no strong intensity for the 'neutral' emotion.
- Statement (01 = "Kids are talking by the door", 02 = "Dogs are sitting by the door").
- Repetition (01 = 1st repetition, 02 = 2nd repetition).
- Actor (01 to 24. Odd numbered actors are male, even numbered actors are female).

2.5.2 SAVEE

The Surrey Audio-Visual Expressed Emotion (SAVEE) Database, was recorded for the purpose of developing automatic emotion recognition systems. It consists of recordings from only male actors (four actors) aged from 27 to 31 for seven emotions: Anger, disgust, fear, happiness, sadness, surprise and neutral. For each emotion, a total of 15 sentences were recorded; 3 common, 2 emotion-specific and 10 generic. For the neutral emotion, the three common and 12 emotion-specific sentences were also recorded adding up to 30 neutral sentences. In total, there are 480 utterances per speaker [9]

Figure 2-2 : SAVEE actor frames examples (source (Jackson & Haq ,2015))

It is important to mention that the recordings took place in a 3D vision laboratory during different periods of the year. The recordings made sure to avoid bias due to fatigue by dividing the text prompts into groups such that each group had sentences for each emotion. At the same time, a 3dMD dynamic face capture system was used to capture the 2D frontal color video and Beyerdynamic microphone signals. The main advantages for using this dataset since the beginning is the audio-visual nature together with the fact that it is the smallest dataset which makes it ideal to start with. It speeds up the process when trying new solutions and it is a good starting point for initial results.

2.6 Related Work

We will go over some research about speech and audio emotion recognition.

2.6.1 Audio

These papers used different dataset as RAVDESS [10] [11],SAVEE[1],IEMOCAP, Berlin Database of Emotional Speech [12] [13] , RML [14] ,BAVED [15] and uses feature extraction as MFCC [10] [16] and Mel spectrogram [17] [18] [12] [13] [11] from audio and implementation models like as random forest [10], MLPNN [10], SVM [10],CNN [10] ,KNN[1],LSTM [19] and Get the best accuracy of the techniques used in these papers is 70.83% on RAVDESS dataset

Table 2-2 : Audio Related work

Paper	Year	Dataset	Feature extraction	Model	Accuracy	Average accuracy	Classes
Emotion Recognition from Speech Signals Using Machine Learning and Deep Learning Techniques. [12]	2021	RAVDESS SAVEE	MFCC	random forest	RAVDESS 61.81% SAVEE 68.75%		8
				MLPNN	RAVDESS 68.75% SAVEE 75%	70.65%	
				SVM	RAVDESS 70.13% SAVEE 70.83%	70.13%	
				CNN	RAVDESS 70.83% SAVEE 61.46%	70.83%	
				KNN	RAVDESS 61.11% SAVEE 63.54%		

Improved Speech Emotion Recognition using Transfer Learning and Spectrogram Augmentation [17]	2021	IEMOCAP	Mel spectrogram	ResNet34	3 Experiment on IEMOCAP 1. WA=66.02% 2. WA=65.62% 3. WA=63.61%		
Time Window Analysis for Automatic Speech Emotion Recognition [12]	2018	Berlin Database of Emotional Speech	Spectrogram	CNN	76.87%		
Speech Emotion Recognition from Spectrograms with Deep Convolutional Neural Network [13]	2017	Berlin emotions dataset	Spectrogram	CNN	84.3%		
Multimodal Emotion Recognition on RAVDESS Dataset Using Transfer Learning [11]	2021	RAVDESS	Mel Spectrogram	Alexnet	61.67%		
				CNN-14	76.58%		
Deep features-based speech emotion recognition for smart affective services [13]	2017	EMO-DB	Spectrogram	Alexnet	79.14%		
		RML			85.46%		6

Human emotion recognition by optimally fusing facial expression and speech feature [14]	2020	AFEW 6.0	Mel spectrogram	LSTM +CNN	37.90%		
		eNTERFACE'05			49.15%		
Arabic Speech Emotion Recognition Employing Wav2vec2.0 and HuBERT Based on BAVED [15]	2021	BAVED	(Raw audio) Wav2vec 2.0 Hubert Base Hubert Large	MLP and Bi-LSTM Classifiers	89% 87% 84%		4
LIGHT-SERNET: A LIGHTWEIGHT FULLY CONVOLUTIONAL NEURAL NETWORK FOR SPEECH EMOTION RECOGNITION [16]	2021	IEMOCAP	MFCC	CNN	(improvised) 68% (scripted + improvised) 66%		5
		EMO-DB			92%		
Multimodal Emotion Recognition using Deep Learning. [19]	2021	IEMOCAP		LSTM	73.98%		

2.6.2 Video

Many models are used in the field of emotion recognition from video, And the most used one is CNN combined with another deep learning network, such as CNN+RNN [14] and CNN+LSTM [20]. AlexNet, GoogleNet and ResNet are also used [21] [22] .Some papers use other models such as VGG-19 ,VGG-16 and bi-LSTM. One of the most used datasets is the RAVADESS [11] [23]dataset, Also SAVEE [21] And RML [21] [14] are used in many papers.

Table 2-3 : Video Related work

paper	year	dataset	feature extraction	model	accuracy	No of classes
Video-based person-dependent and person-independent facial emotion recognition [20]	2020	SAVEE RML	facial landmark using openface	VGG-16	94.45	6 Emotions
				VGG-19	95.60	
			23 frames	ResNet-50	98.23	
				AlexNet	94.86	
				GoogleNet	97.18	
Human emotion recognition by optimally fusing facial expression and speech feature [14]	2020	RML		CNN + RNN	88.96%	6 Emotions
		AFEW 6.0			48.30%	
		eNTERFACE'05			78.38%	
Multimodal Emotion Recognition on RAVDESS Dataset Using Transfer Learning [11]	2021	RAVADESS	Saliency Maps	STN + MAX Voting	57.08%	8

A CNN-LSTM BASED DEEP NEURAL NETWORKS FOR FACIAL EMOTION DETECTION IN VIDEOS [21]	2021	RAVADESS	Masked face images in RGB format	CNN + LSTM (6 Layers)	65.35%	6
		CREMA-D dataset	Masked face images in RGB format	CNN + LSTM	78.52%	6
Human Emotion Recognition in Video using Subtraction Pre-Processing [22]	2019	RAVADESS	face detection or landmark detection	AlexNet structure	79.74	6
				GoogleNet	62.89	
				ResNet-4	75.89	
A Proposal for Multimodal Emotion Recognition Using Aural Transformers and Action Units on RAVDESS Dataset [23]	2022	RAVADESS	OpenFace toolkit	bi-LSTM	62.13% ± 2.51%	8

2.6.3 Fusion

Related work papers in the table below used different datasets like RML,Enterface05,BAUM-1s, RAVDESS and Crema-D and achieved different accuracies with different networks like DBN , MRPN and STN . Multi-modal Residual Perceptron Network for Audio-Video Emotion Recognition paper [24].which achieved the highest accuracy in the related work with accuracy 91.4% on multimodal fusion using video frames augmentation and audio augmentation (MRPN (end-to-end), Resnet18+Transformer(avg), VA+AO). on RAVADESS dataset.

Table 2-4 : Fusion Related work

Paper	Year	Dataset	Accuracy	Model	No of classes
Audio-visual emotion fusion (AVEF): A deep efficient weighted approach [24]	2019	RML	82.38%	DBN	6 Emotions
			(86.6,90.1)		Binary
		Enterface05	85.69%		6 Emotions
			(92.3,91.8)		Binary
		BAUM-1s	59.17%		6 Emotions
			(80.5,82.3)		Binary
Multimodal Emotion Recognition on RAVDESS Dataset Using Transfer Learning [25]	2021	RAVDESS	80.08%	STN with a temporal model or a pooling strategy before concatenating them with the aural embeddings	8
A Proposal for Multimodal Emotion Recognition Using Aural Transformers and Action Units on RAVDESS Dataset [23]	2022	RAVDESS	86.70%	late fusion models -multinomial logistic regression algorithm	8
Multi-modal Residual Perceptron Network for Audio-Video Emotion Recognition [26]	2021	RAVDESS	91.4%	MRPN (end-to-end), Resnet18 + Transformer(avg), VA+AO	7 EMOTIONS
		Crema-D	83.15%		
Learning Affective Features with a Hybrid Deep Model for Audio-Visual Emotion Recognition [27]	2016	RML	80.36%	DBN	6 EMOTIONS
		eNTERFACE05	85.97%		
		BAUM-1s	54.57%		

Chapter 3 **(AUDIO WORK)**

3.1 Introduction

In this chapter, we will review the work we have done in the audio model, we will review our work in the data extraction process and how to choose the appropriate method for that project, and then we will review our work in building the model and how to modify it to reach a satisfactory result and high accuracy.

3.2 Preprocessing and feature extraction

The first step in any automatic sound recognition system is to extract features i.e., identify the components of the audio signal that are good for identifying the linguistic content and discarding all the other stuff which carries information like background noise. This phase is done after preprocessing which is applied to make us test our system on in different conditions, we obtain by manipulating our dataset.

In order to be able to choose the appropriate method for extracting Feature, we had to try the most famous methods available, so we chose two methods to prefer between them, MFCC and SPECTROGRAM, and this is due to what we found in the previous work (as we mentioned in the chapter two).

3.2.1 MFCC

Mel Frequency Cepstral Coefficients (MFCCs) are a feature widely used in automatic sound recognition. They were introduced by Davis and Mermelstein in the 1980's and have been state-of-the-art ever since. We transform the sound waveform into MFCCs representation as shown in (figure 3.1). The reason we chose this form is that it is the most usable and efficient one to the patterns of human recognition and preservation. The most used MFCCs consist of between 13 to 45 coefficients. The first 13 represent the basic feature of the wave, and the rest of the feature are redundant (even if you feed it into your model, you will get slight performance) as the lower order coefficients contain most of the information about overall spectral shape of the source. The zero order coefficient indicates the average power of input signal and the first-order coefficient represent the increasing levels of spectral details [1][4].

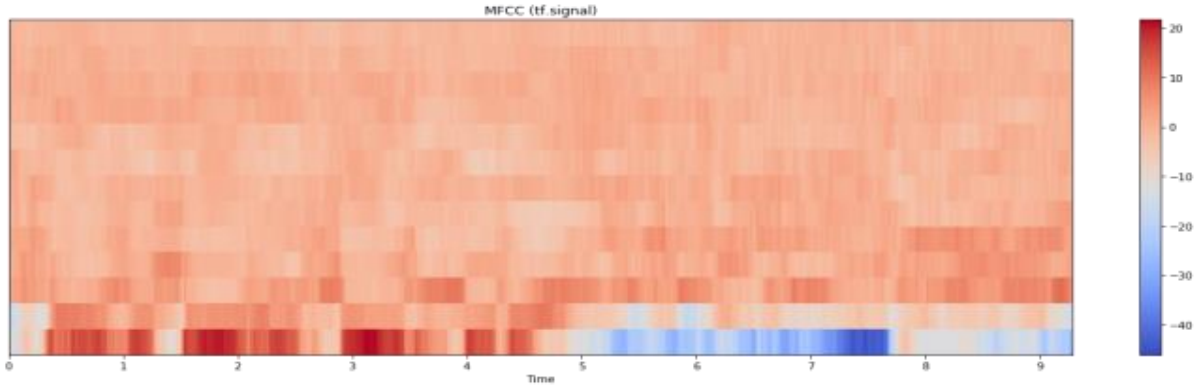


Figure 3-1 :MFCC Coefficient

We tried an existing model that works with MFCC to see how accurate it is in analyzing the components of sound and how accurate it is in recognizing emotions

3.2.1.1 MFCC models

- **Model No 1 [28]**

Two Parallel CNN Model trained on RAVDESS speech and song data set on 8 emotions, split data set into 80 % for train, 10 % for test and 10 % for validation, with augmentation using Additive White Gaussian Noise (AWGN) on audio signals. Model achieved an average accuracy 73.23 ± 2 % test accuracy. Fig 3.2 Fig 3.3 show loss plot and confusion matrix respectively.

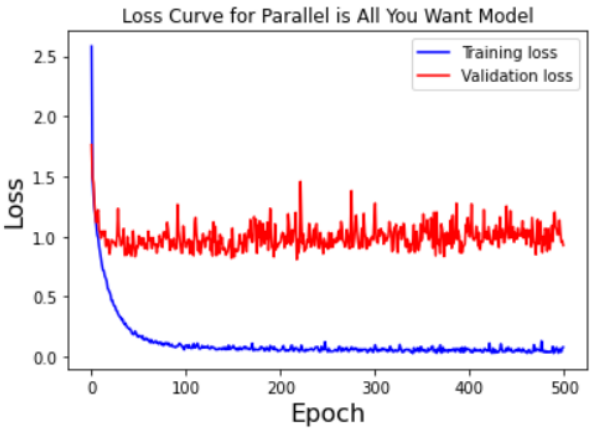


Figure 3-3 : Loss plot

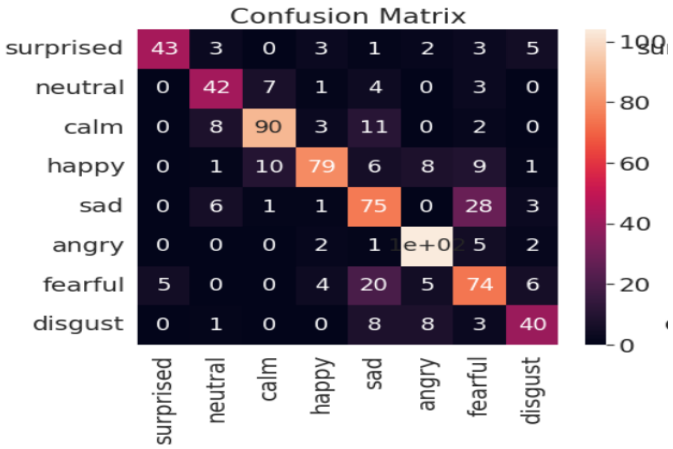


Figure 3-2 : Heatmap of confusion matrix

- **Model No 2 [10]**

CNN Model trained on RAVDESS speech and song data set on 6 emotions, split data set into 80 % for train, 10 % for test and 10 % for validation , Model achieved an average accuracy 74 ± 1 % test accuracy, figure 3.4 show results with best accuracy .

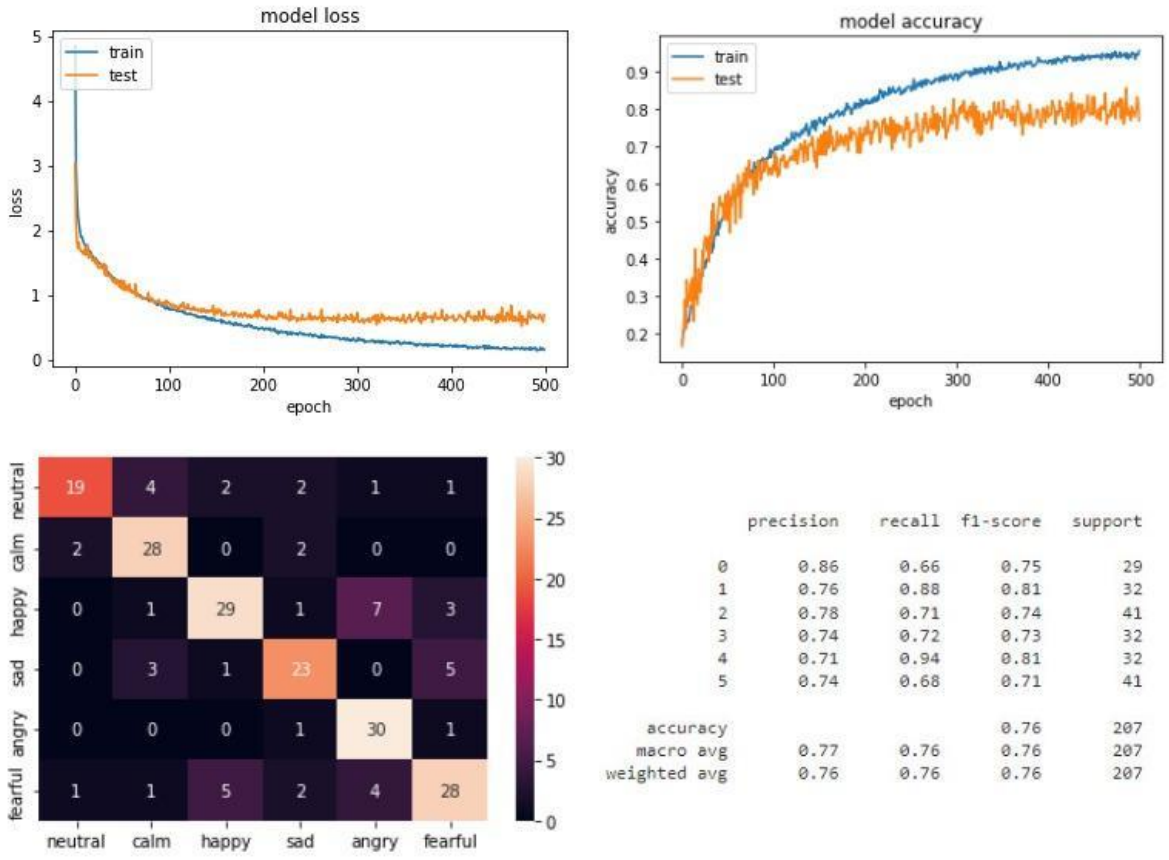


Figure 3-4 : Show model loss, accuracy plot, heatmap of confusion matrix and classification report respectively

3.2.2 SPECTROGRAM

A spectrogram is an image representation of the waveform signal, it shows its frequency intensity range over time, it can be very useful when evaluating the signal's frequency distribution over time [5].

The first thing that is done is to try and extract useful information from the digital representation of an audio signal. The Fourier transform is a mathematical formula that decomposes a signal into its individual frequencies and the frequency's amplitude. In other words, it converts the signal from the time domain into the frequency domain, also called a spectrum. This is possible since any signal can be decomposed into a set of sine and cosine waves that add up to the original signal. To compute a power spectrogram, the function librosa.stft(y) is used (librosa development team, 2021), where 'y' is the time series obtained from the audio file. The Short-time Fourier Transform (STFT) represents a signal in the time-frequency domain by computing Discrete Fourier Transforms (DFT) over short overlapping windows. Therefore the function returns an array of Fourier transform coefficients (complex numbers). These coefficients give the phase and amplitude of the audio signal [6]. Since humans don't perceive the phase very well, the signal is reduced to the amplitude and therefore the absolute number of the complex value is obtained as shown in (figure 3.5).

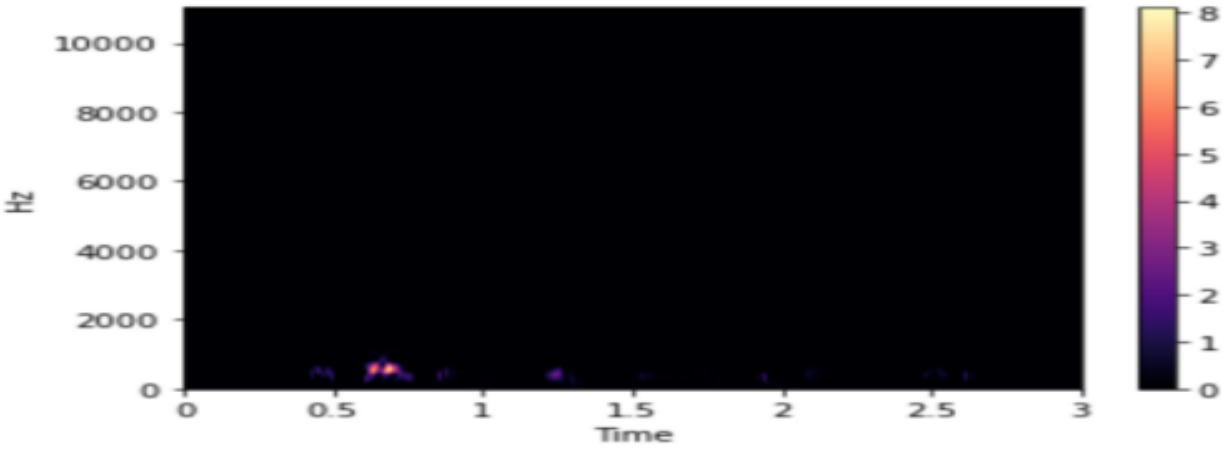


Figure 3-5 : Show signal is reduced to the amplitude and therefore the absolute number of the complex value is obtained as shown here

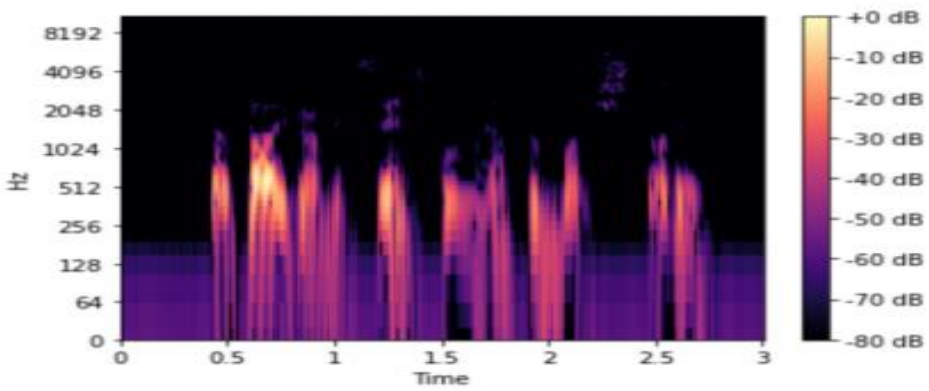


Figure 3-6

Not much can be seen and this is because most sounds humans hear are concentrated in very small frequency and amplitude ranges. Therefore, in order to observe something a small adjustment is made, the y-axis (frequency) is transformed to log-scale and the color-axis (amplitude) is transformed to decibels (librosa development teams, 2021). as shown in (figure 3.6).

Given the spectrogram obtained earlier, it is mapped on to the mel scale acquiring the mel-spectrogram (librosa's development team, 2021). Both the y-axis (frequency), and the power (amplitude squared) axis are once again transformed to log scale and decibels respectively as shown in (figure 3.7).

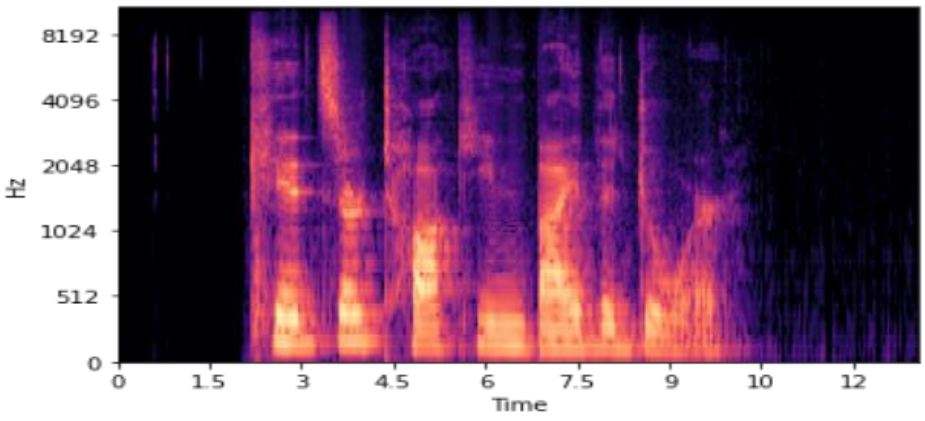


Figure 3-7

The next section discusses our experiments trying different models with spectrogram feature extraction, the focus is not on the model itself, but on the spectrogram and the results it achieves in emotion recognition.

3.2.2.1 Spectrogram models

In this section our Goal in the spectrogram models that will be showed in **Table 3.1** is to know only the average of the accuracy the spectrogram achieve and not to go deeply in the models.We tried 6 different models from different resources to evaluate the range of accuracy achieved by spectrogram feature extraction. These models are listed in the next table **Table 3.1 [29][30][31]**.

Table 3-1 : Spectrogram models

Model NO	Model	Dataset	No of classes (Emotions)	Test accuracy
1	Stacked_CNN_Attention_LSTM	RAVDESS Speech	8	61%
2	Parallel_CNN_Attention_LSTM	(16 Actor) RAVDESS Speech	8	64.5%
3	2D CNN (6 Convolution layers)	RAVDESS Speech & Song	6	69%
4	VGG16 Fine Tuning With Image Augmentation	RAVDESS Speech	8	58%

The Models results and work in the above table will be discussed briefly below**.**

- **Model No 1**

Stacked_CNN_Attention_Lstm Model is referenced in [29]. Dataset is splitted into train, validation and test sets, with the following percentage (80,10,10) %. Preprocessing in this reference or in this work that the signals are loaded with a sample rate of 48KHz and cut off to be in range of [0.5,3] seconds. Mel spectrogram is calculated and used as input for the model. This model is trained on speech only of the RAVDESS dataset as mentioned above in the table and the dataset is augmented by adding Additive White Gaussian Noise (with SNR in range [15,30]) This enormously improved accuracy and removed overfitting. The model as mentioned in referenced work in [29] consists of three convolution blocks and a LSTM block then a linear softmax layer. The model is trained using optimizer SGD and learning rate 0.01 as mentioned in the reference and achieved 61.33% test accuracy. Figures that show our results on this referenced work are listed below. Figure 3.8 shows confusion matrix and Figure 3.9 shows train loss vs validation loss. Figure 3.10 shows a heatmap of the confusion matrix.

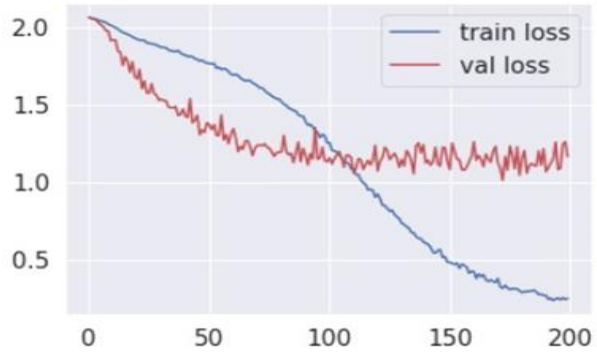


Figure 3-8: Loss plot

Figure 3-10 : Confusion matrix

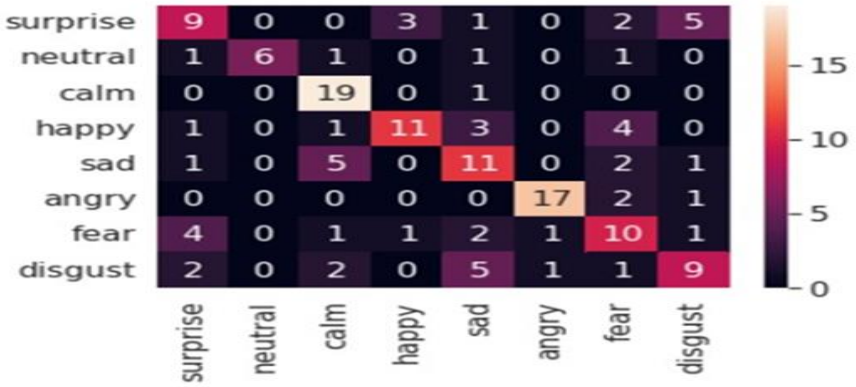


Figure 3-9 : Heatmap of confusion matrix

- **Model No 2**

Parallel_CNN_Attention_LSTM model is referenced in [29].This model is close to Model No 1. Dataset is splitted into train, validation and test sets, with the following percentage (80,10,10) %. Preprocessing in this reference or in this work that the signals are loaded with a sample rate of 48KHz and cut off to be in range of [0.5,3] seconds. Mel spectrogram is calculated and used as input for the model. This model is trained on speech only of the RAVDESS dataset as mentioned above in the table and the dataset is augmented by adding Additive White Gaussian Noise (with SNR in range [15,30]) This enormously improved accuracy and removed overfitting. The model as mentioned in the following reference [29] consists of 4 blocks of convolutional blocks and LSTM block (Parallel_CNN_Attention_LSTM). The model is trained on 16 actors only from the RADVASS dataset as mentioned in table 3.1**.** The model is trained using optimizer SGD and learning rate 0.01 as mentioned in the reference and achieved 64.5% as a test accuracy. Figures that show the results of this model are listed below. Figure 3.11 shows confusion matrix and Figure 3.12 shows train loss vs validation loss. Figure3.13 shows heatmap of confusion matrix

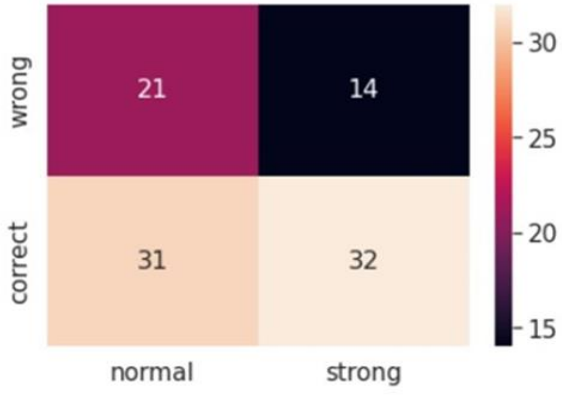


Figure 3-11 : Confusion matrix

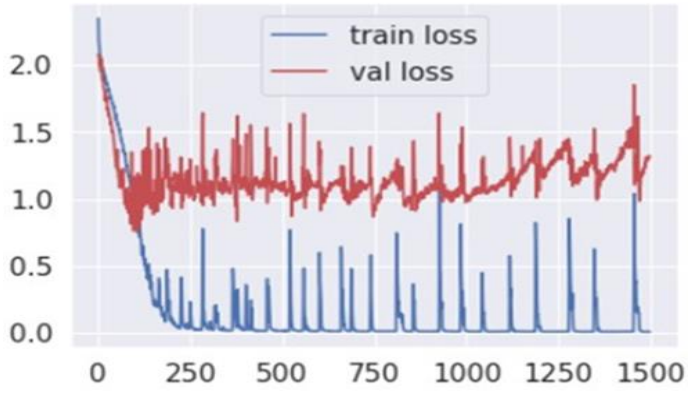


Figure 3-12 : Loss plot

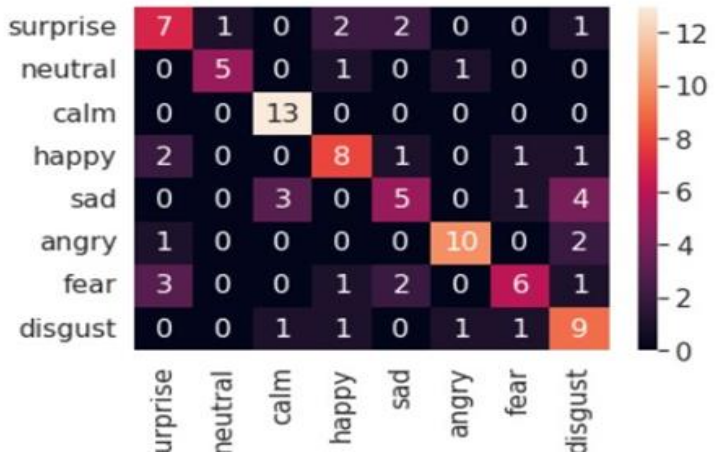


Figure 3-13 : Heatmap of confusion matrix

- **Model No 3**

The 2D CNN model is referenced in [31]. As mentioned in the reference the dataset splitted into 80% train, 10 % test and 10% validation set. The model consists of 6 convolutional layers each layer is followed by a batch normalization layer to normalize output from the previous convolutional layer then followed by activation function followed by max pooling layer then dropout. After the six layers there is a flatten layer then a Dense layer with the number of units equal to the number of the classes being classified. The model is trained using optimizer Adam. The model achieved a test accuracy of 69%. figure 3.14, figure3.15, figure 3.16 shows accuracy and loss plot and heatmap of confusion matrix respectively.

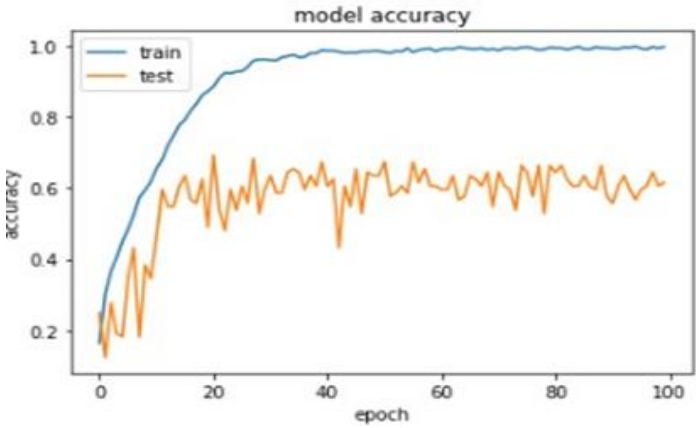


Figure 3-14 : Model accuracy plot

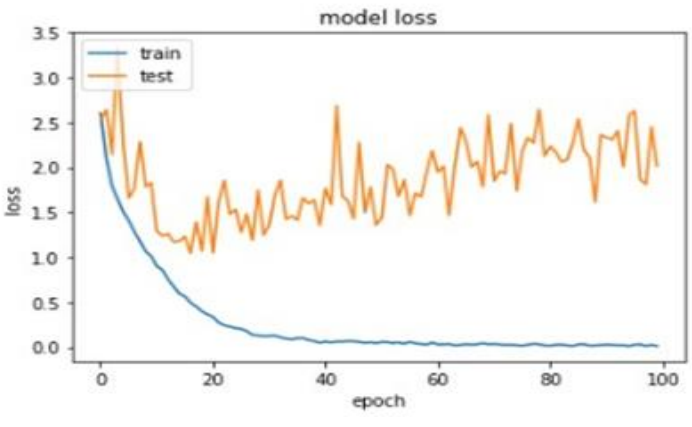


Figure 3-15 : Loss plot

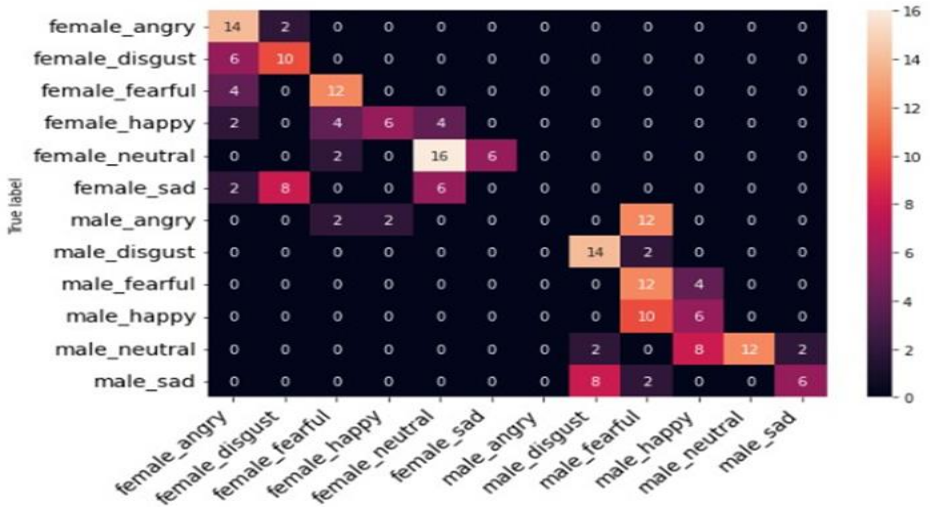


Figure 3-16 : Heatmap of confusion matrix

- **Model No 4**

VGG16 Fine Tuning with Image Augmentation Model is referenced in [30]. We did some work on the dataset like dividing the dataset into classes labeled by the classes (emotions). Then we calculated spectrogram images for audio and saved it in files labeled by the emotions. Model is trained using Adam optimizer and categorical_crossentropy as a loss function as mentioned in the reference with initial learning rate (5e-5) and uses a function that reduces learning rate when validation accuracy has stopped improving with minimum learning rate (0.000001). This model achieved an accuracy of 58% test accuracy. figure3.17, figure3.18 shows accuracy and loss plots respectively. figure3.19 shows the classification report.

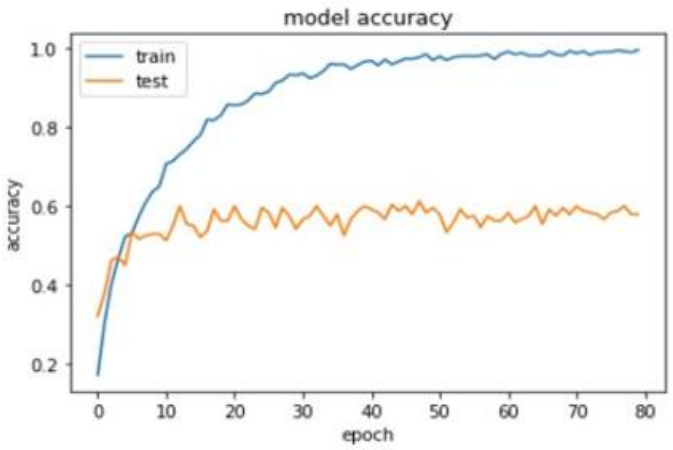


Figure 3-18 : Accuracy plot

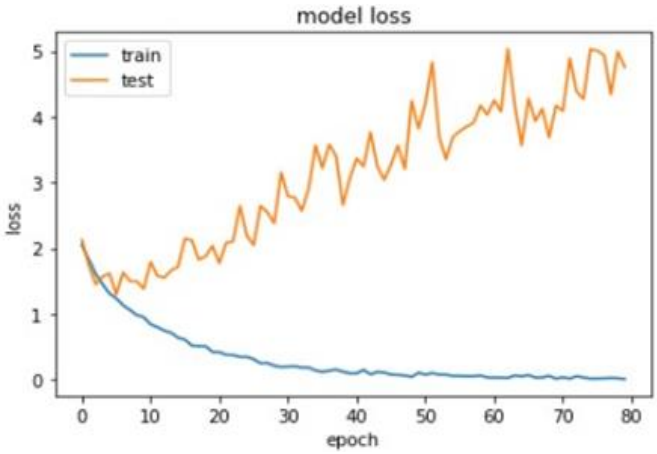


Figure 3-17 : Loss plot

	precision	recall	f1-score	support
angry	0.60	0.81	0.69	32
calm	0.69	0.62	0.66	32
disgust	0.55	0.66	0.60	32
fearful	0.59	0.53	0.56	32
happy	0.57	0.25	0.35	32
neutral	0.50	0.44	0.47	16
sad	0.34	0.41	0.37	32
surprised	0.77	0.84	0.81	32
accuracy			0.58	240
macro avg	0.58	0.57	0.56	240
weighted avg	0.58	0.58	0.57	240

Figure 3-19 : Classification report

3.2.3 Results of feature extraction

As shown in figure 3.20 we can see that the best accuracy achieved by the spectrogram was 69%, and the best accuracy achieved by the MFCC was 75%, so we decided to use the MFCC as a features extraction method in our project.

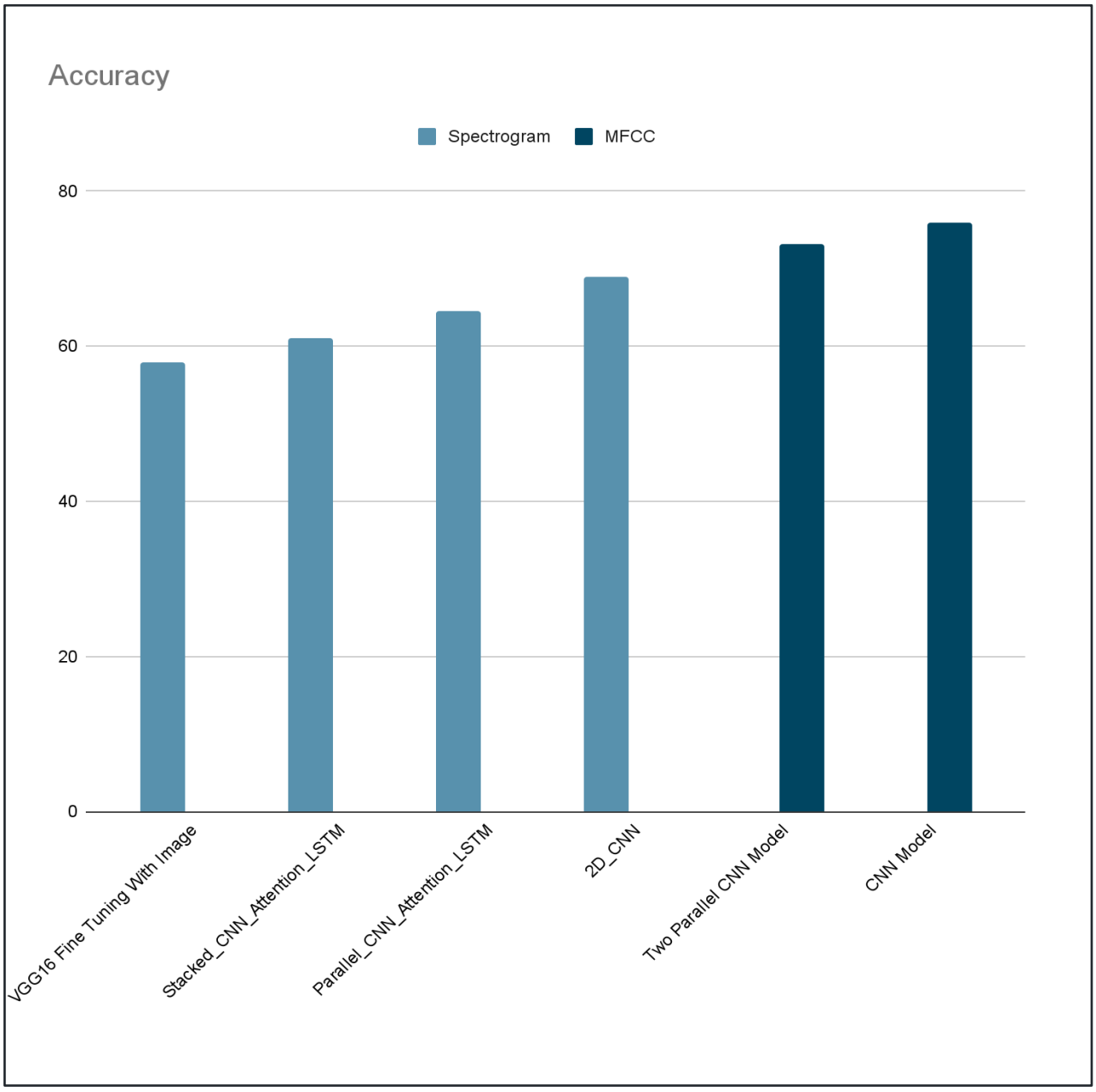


Figure 3-20 : Chart of accuracy results of feature extraction

3.3 Model Work

In order to build a model with good accuracy, we had to choose the appropriate model through previous work, so we chose two models that have the highest accuracy using the MFCC, Then we will make adjustments and improvements in order to achieve a satisfactory result and high accuracy

3.3.1 Two Parallel CNN Model

We trained this model before in section 3.2.1.1 to choose the feature extraction method, now we will go in depth in this model in sections 3.3.1.1 and 3.3.1.2, and try to make changes to achieve higher accuracy in section 3.3.1.3.

3.3.1.1 Model Architecture

The work in this reference [28] proposes two parallel convolutional neural networks (CNN) in parallel with a transformer encoder network to classify audio data. Model is shown in the figure below Figure 3.21.

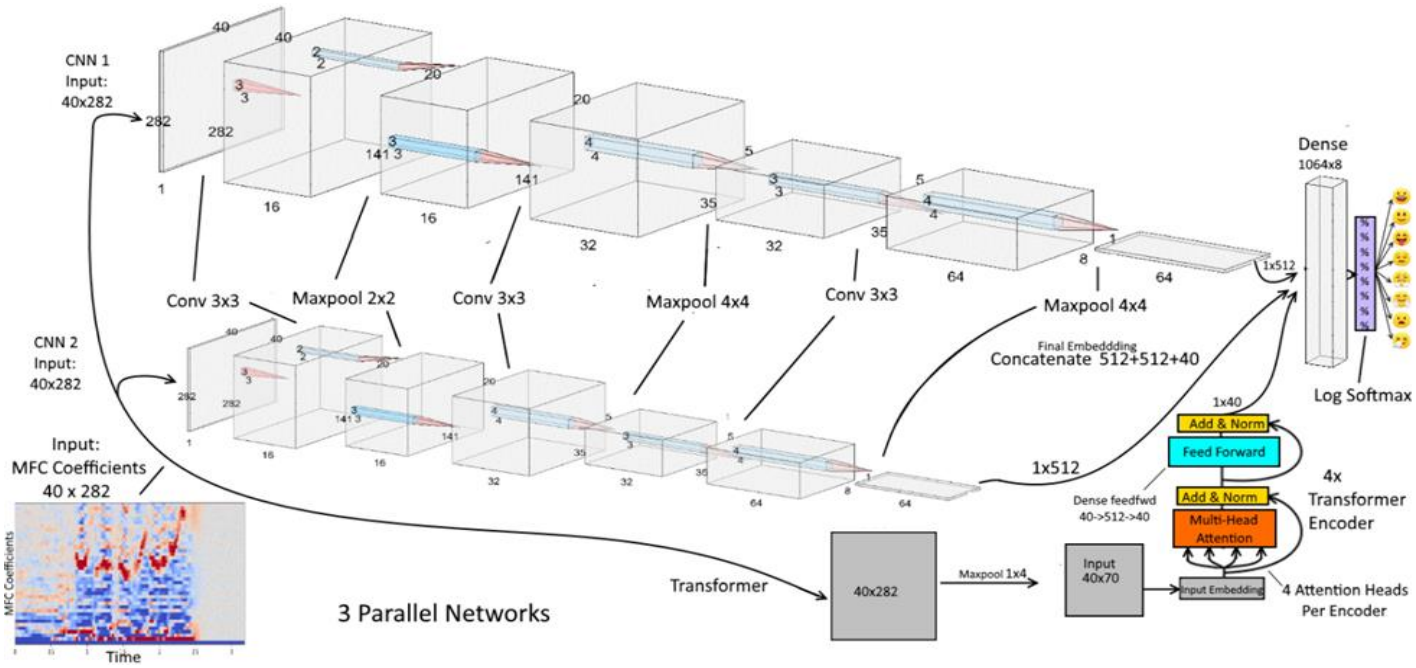


Figure 3-21 : Two parallel CNN model architecture

We use 3x3 kernels in all 3 layers in both CNN blocks. The first layer only has a single input channel creating a 1x3x3 filter, with 16 output channels necessitating 16 such unique 1x3x3 filters with 1x3x3=9 weights per filter. The next layer has 16 input channels and 32 output channels, producing 32 unique 16x3x3 filters, each filter having 16x3x3 = 144 weights. That is, the second layer is applying 32 differently weighted 16x3x3 filters to an input volume of 16x20x141 (the 2x2 max pooled output of the first layer), producing an output feature map of 32x5x35 after 4x4 stride 4 max pooling. The last layer has 32 input channels, so a 32x3x3 filter, and 64 output channels, so 64 unique such filters with 32x3x3=288 weights each. The last layer produces an output feature map of 64x1x8 after 4x4 stride 4 max pooling.

3.3.1.2 Model implementation

After we know model Architecture in the previous section, in this section will know how the model works and its results.
We combine the CNN for spatial feature representation and the Transformer for temporal feature representation. Because of the sequential nature of the data, we will also use the Transformer to try and model as accurately as possible the temporal relationships between pitch transitions in emotions. CNNs with 2D convolutional layers are the gold standard for image processing, other than the recent advances in the Transformer for images. 2D convolution layers accept input feature maps in a (N,C,H,W) (batch size, channel, height, width) format. We have 7356 MFCC plots - 2452 native (RAVDESS speech & song) and 4904 noise augmented by (AWGN) augmentation - each MFCC plot is of shape 40x282 with 40 MFC coefficients representing different Mel pitch ranges, with 282 timesteps for each MFC coefficient. We can imagine MFCC plots to be a black and white image with 1 signal intensity channel. Our MFCC input feature tensor will thus be of shape (7356, 1, 40, 282) before splitting for training.trained on Ravadess speech and song data set , split data set into 80 % for train , 10 % for test and 10 % for validation, using **`CrossEntropyLoss`** as loss function , **`SGD`** as optimizer and learning rate **`LR = 0.01`** . Model achieved an average accuracy 73.23 ± 2 % test accuracy on 8 emotions using RAVDESS speech & song.Figure 3.2 Figure 3.3 show loss plot and confusion matrix respectively.

3.3.1.3 Model Schemas

Table 3-2 : Two parallel CNN model schemas

Schema No	Schema	Dataset	No of emotions	Accuracy
1	Add Layer	RAVDESS speech & song	6	75.12 %
2	One block CNN instead of 2 parallel CNN	RAVDESS speech & song	6	77.03 %
3	3 parallel CNN blocks	RAVDESS speech & song	6	77.51 %
4	Two TRANSFORMER LAYERS INSTEAD OF 4	RAVDESS speech & song	6	77.99 %
5	AUGMENTATION On Train Only	RAVDESS speech & Song	6	79.4 %
6	**LR DECREASE WITH TIME** **&** **AUGMENTATION** **On Train Only**	**RAVDESS** **Speech** **&** **Song**	**6**	**80.86 %**
7	Decrease No of Emotions classified	RAVDESS Speech & Song	6	76.71%
8	Decrease No of Emotions classified & AUGMENTATION On Train Only	RAVDESS Speech & Song	5	80.12%

9	AUGMENTATION On Train Only & Change Drop out ratio of first two Convolutional layer in each CNN block	RAVDESS Speech & Song	6	79.9%
10	Decrease No of Emotions classified & AUGMENTATION On Train Only	RAVDESS Speech & Song	4	84.96%
11	AUGMENTATION On Train Only & CHANGING Drop ratio	RAVDESS Speech & Song	6	77%
12	Decrease No of Emotions classified & AUGMENTATION On Train Only	RAVDESS Speech & Song	5	78.95%
13	AUGMENTATION On Train Only & Change Drop out ratio of second Convolutional layer in each CNN block	RAVDESS Speech & Song	6	69.86%

- **Schema 1:**

Add 2D convolution layer in each CNN block with 64 input channels, 128 output channels, kernel size = 3, stride=1, padding=1, followed by batch normalization, Relu activation function, max pooling and Dropout with ratio 0.3. This schema achieves accuracy 75.12 % on average.

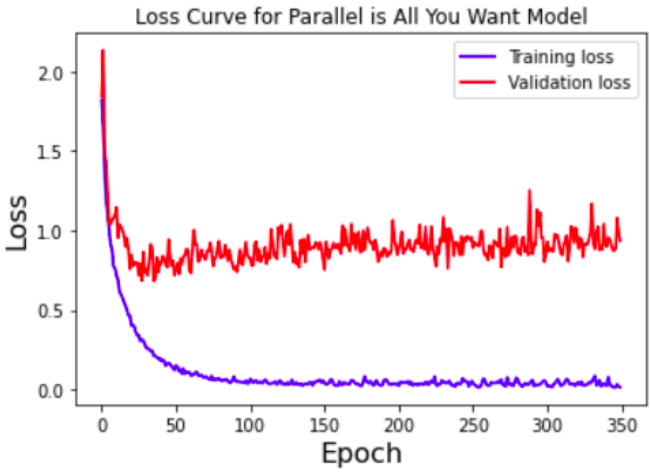


Figure 3-22 : Schema 1 loss curve

- **Schema 2:**

Removing one CNN block**,** Instead of two parallel CNN using one block CNN. This schema achieves accuracy 77.03 % on average

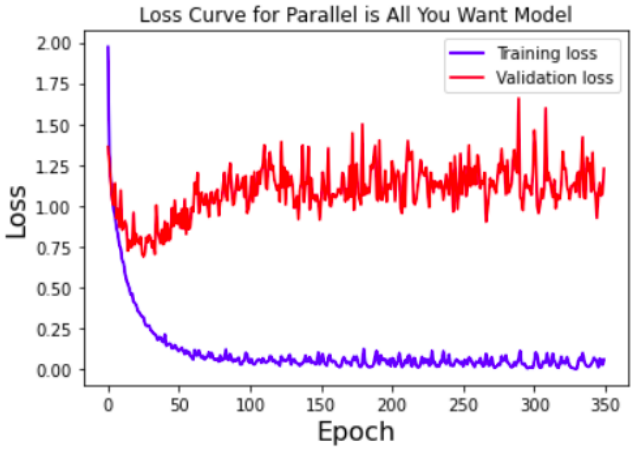


Figure 3-23 : Shema

- **Schema 3:**

Adding one CNN block**,** Instead of two parallel CNN using three parallel CNN, the one block that was added is identical to other blocks, This schema achieves accuracy 77.51 % on average

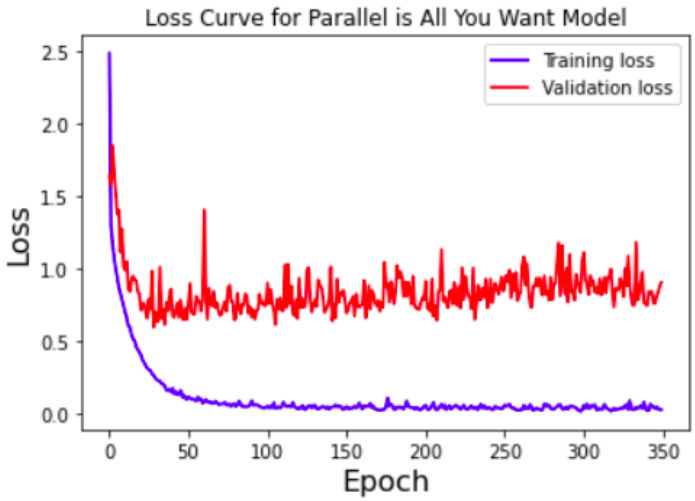


Figure 3-24 : Schema 3 loss curve

- **Schema 4:**

2 self-attention layers in each multi-head self-attention layer in each encoder block instead of 4 self attention layer. Complete transformer block contains 2 full transformer encoder layers instead of 4 full transformer encoder layers.So in this schema we used 2 transformer layer instead of 4 layers. This schema achieved accuracy 77.99%.

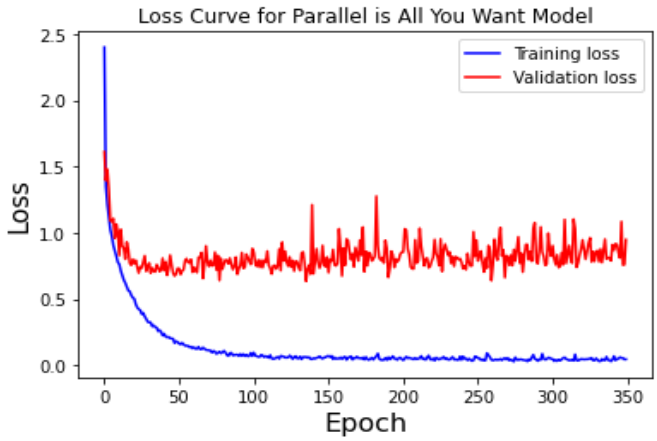


Figure 3-25: Schema 4 loss curve

- **Schema 5:**

Instead of making augmentation before splitting the dataset , will make augmentation after splitting on train dataset only .This schema achieves accuracy 79.4 % on average,

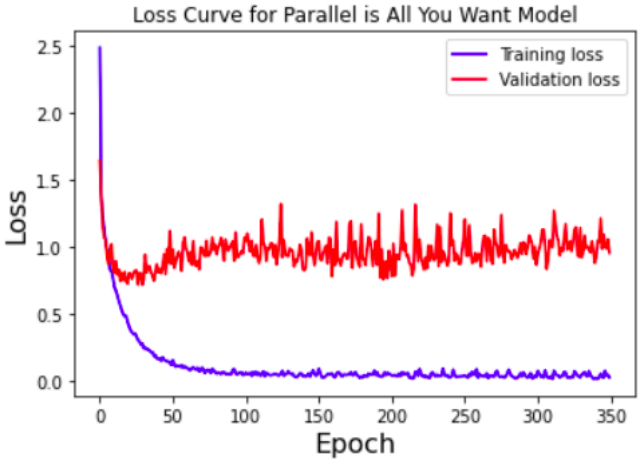


Figure 3-26 : Schema 5 loss curve

- **Schema 6:**

Change Learning Rate Dynamically at the middle of training from lr = 0.01 to lr = 0.001. This schema achieves accuracy 80.86 % on average.

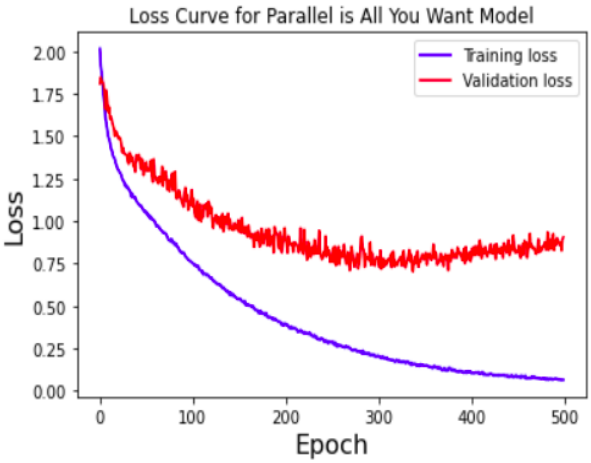


Figure 3-28 : Schema 6 loss curve

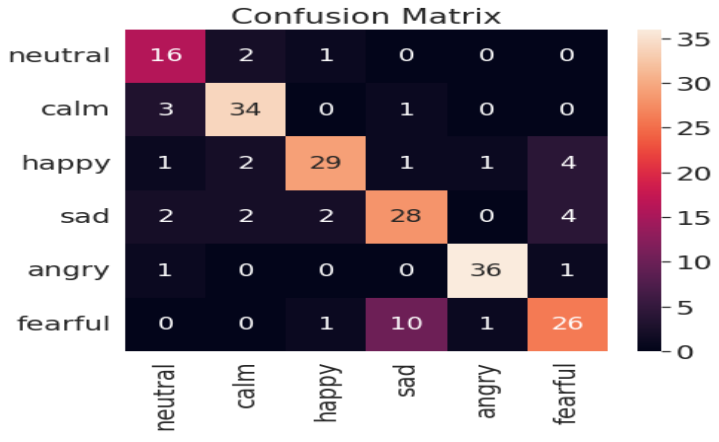


Figure 3-27 : Schema 6 Heatmap of confusion matrix

- **Schema 7:**

Same as Two parallel CNN implementation mentioned in section 3.3.1.2, but on 6 emotions instead of 8 emotions. This schema achieved test accuracy of 76.71%.

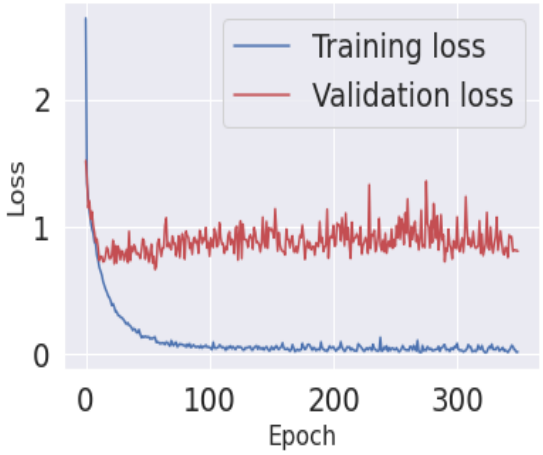


Figure 3-29 : Schema 7 loss curve

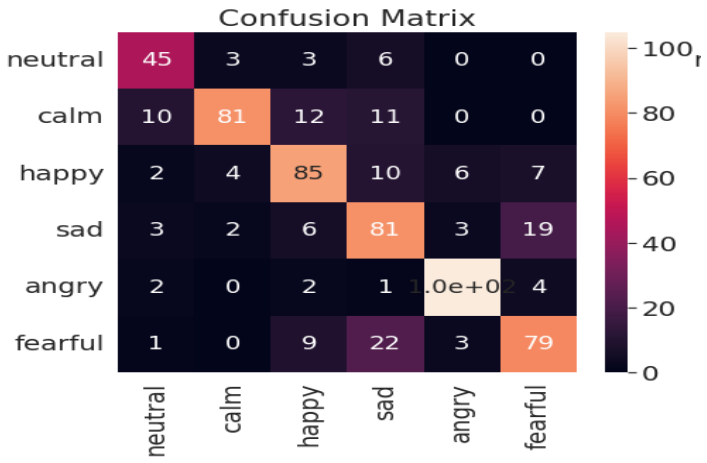


Figure 3-30 : Schema 7 heatmap of confusion matrix

- **Schema 8:**

Same as Two parallel CNN implementation mentioned in section 3.3.1.2, But on 5 emotions (neutral,happy,sad,angry,calm) and augmentation on train set only. This schema achieves an accuracy 80.12%.

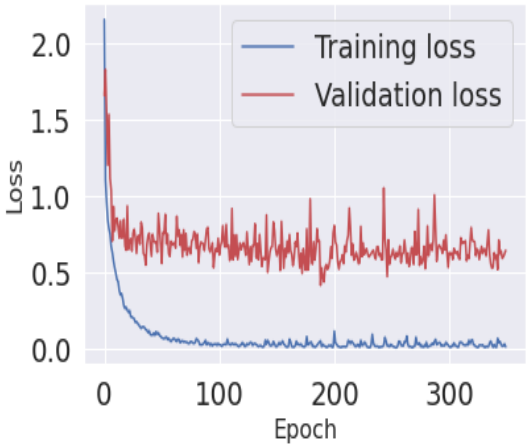


Figure 3-32 : Schema 8 Loss curve

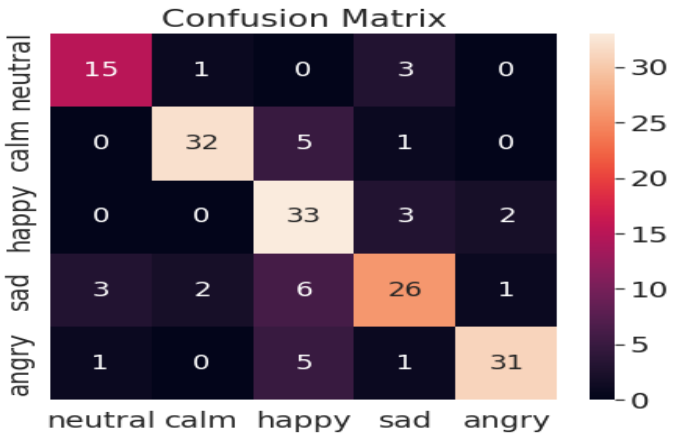


Figure 3-31 : Schema 8 heatmap of confusion matrix

- **Schema 9:**

Augmentation on train only set and changes the dropout ratio of the first two convolutional layers in each CNN block by 0.4 for the first layer and 0.6 for the second layer**.** This schema achieves accuracy of 79.9%.

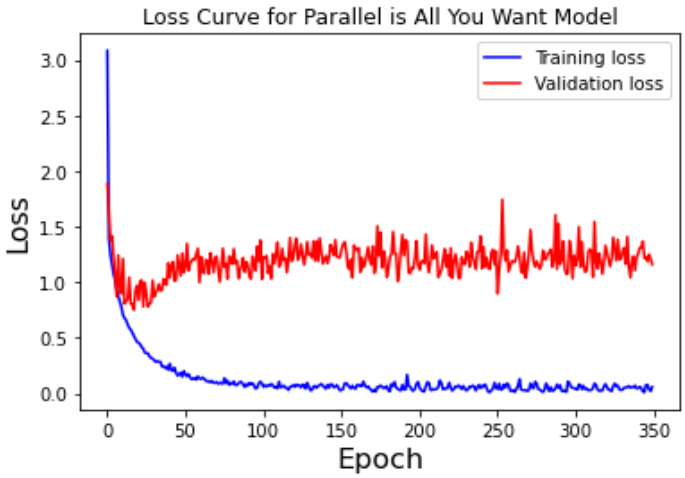


Figure 3-33: Schema 9 loss curve

- **Schema 10:**

Same as Two parallel CNN implementation mentioned in section 3.3.1.2, But on 4 emotions (neutral,happy,sad,angry)and augmentation on train set only. This schema achieves an accuracy 84.96%.

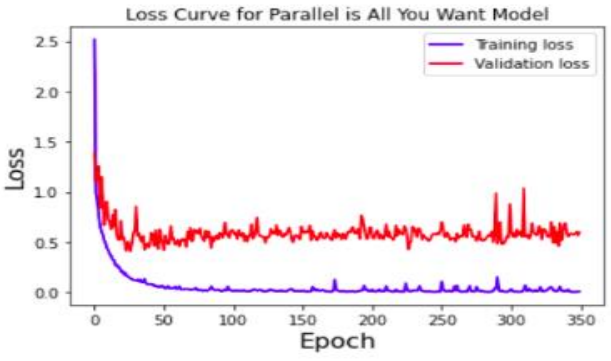


Figure 3-35: Schema 10 loss curve

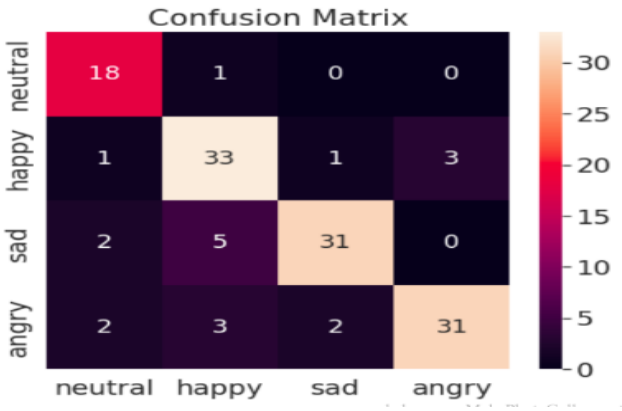


Figure 3-34 : Schema 10 heatmap of confusion matrix

- **Schema 11:**

Augmentation on train only and Change Drop out ratio for transformer layers from 0.4 to 0.3, change Drop out ratio for first convolution layer in each block from 0.3 to 0.2 and change Drop out ratio for second and third convolution layer in each block from 0.3 to 0.4. This schema achieves an accuracy of 77%.

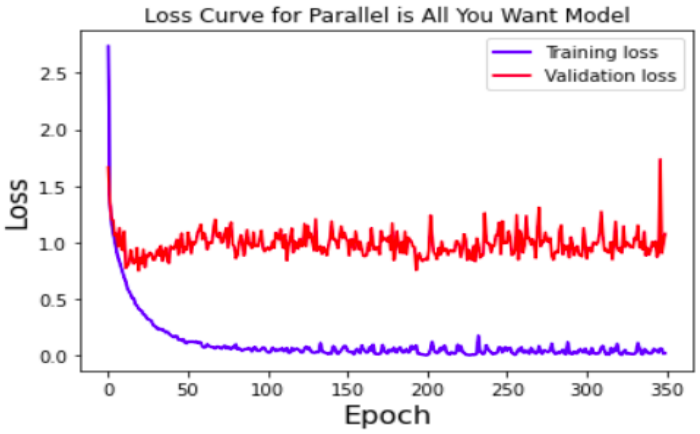


Figure 3-36 : Schema 11 loss curve

- **Schema 12:**

Same as Two parallel CNN implementation mentioned in section 3.3.1.2, But on 5 emotions (calm,happy,sad,angry,fearful) and augmentation on train set only. This schema achieves an accuracy 78.95%.

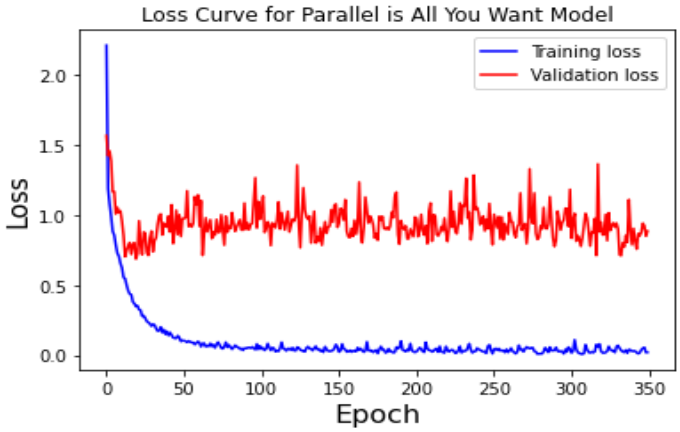


Figure 3-37 : Schema 12 loss curve

- **Schema 13:**

Augmentation on train only and change in dropout ratio of second convolutional layer from 0.3 to 0.8 in each CNN block of model. This schema achieved an accuracy of 69.86%.

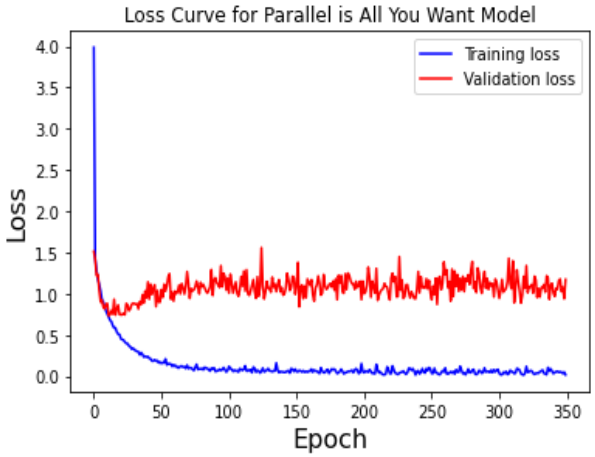


Figure 3-39: Schema 13 loss curve

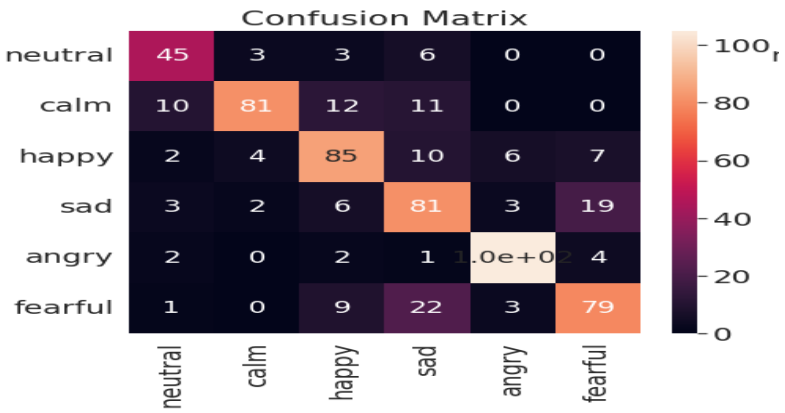


Figure 3-38 : Schema 13 heatmap of confusion matrix

3.3.1.4 Conclusion

Best accuracy achieved on the two parallel CNN model on 6 emotions is 80.86% by Schema 6 as mentioned in table 3.2.

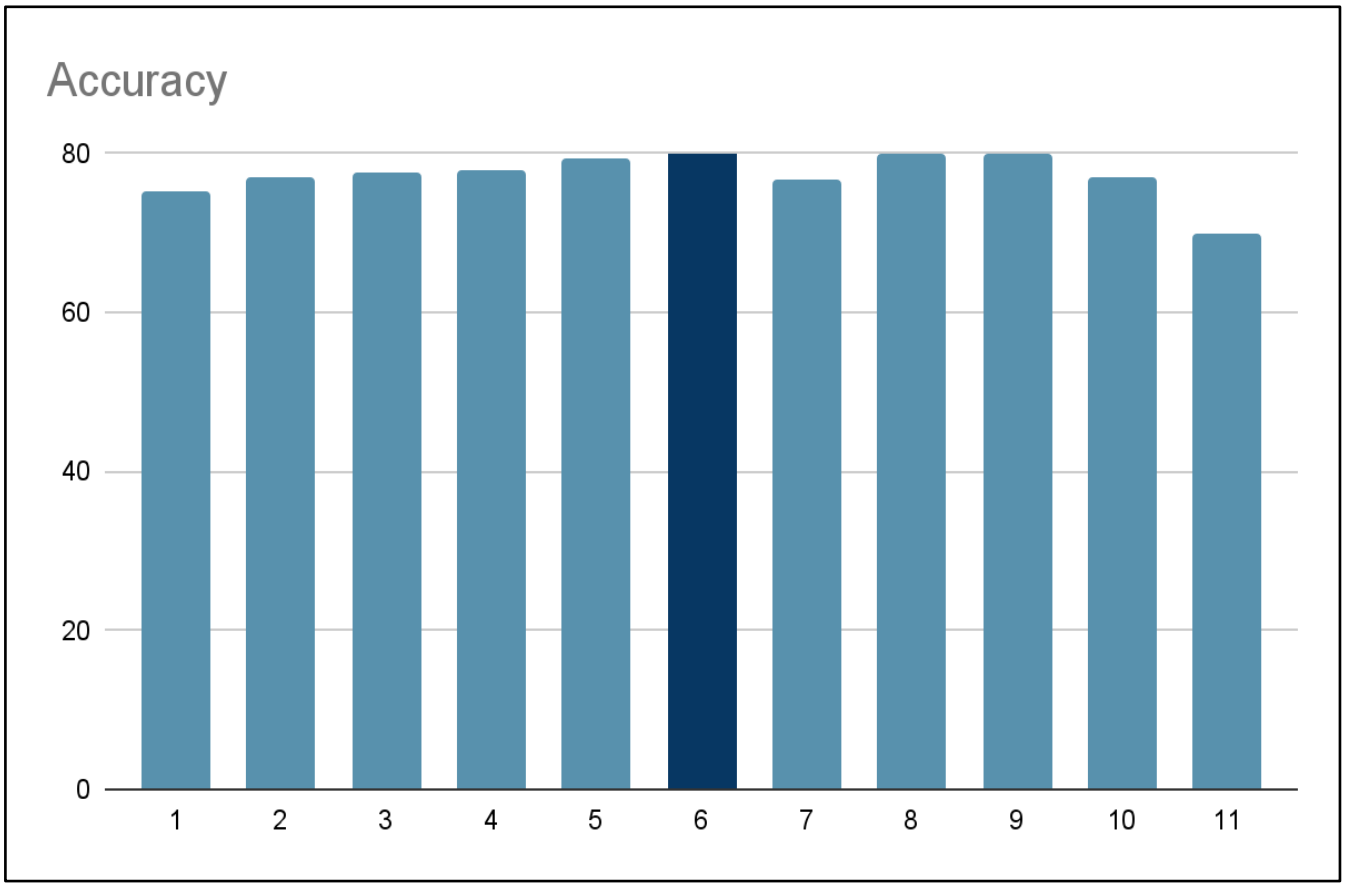


Figure 3-40 : Chart shows accuracy achieved by Two parallel CNN model schemas

3.3.2 CNN Model

We trained this model before in section 3.2.1.1 to choose the feature extraction method, Now we will go in depth in this model in sections 3.3.2.1 and 3.3.2.2, and try to make changes to achieve higher accuracy in section 3.3.2.3.

3.3.2.1 Model Architecture

The convolutional neural network model presented in this reference [10] consists of two convolution layers with kernel size of 8 and employing ReLU activation function. The first convolutional layer contains 128 filters, whereas the second contains 256 filters. Each of the convolutional layers is followed by a max-pooling layer with a pool size of 5. Followed by these layers are the flattening layer and three dense layers with the first two dense layers having 256 output perceptrons and employing ReLU activation function, whereas the third layer has eight output perceptrons employing softmax function. The following figure Fig 3.41 shows the architecture of the CNN model.

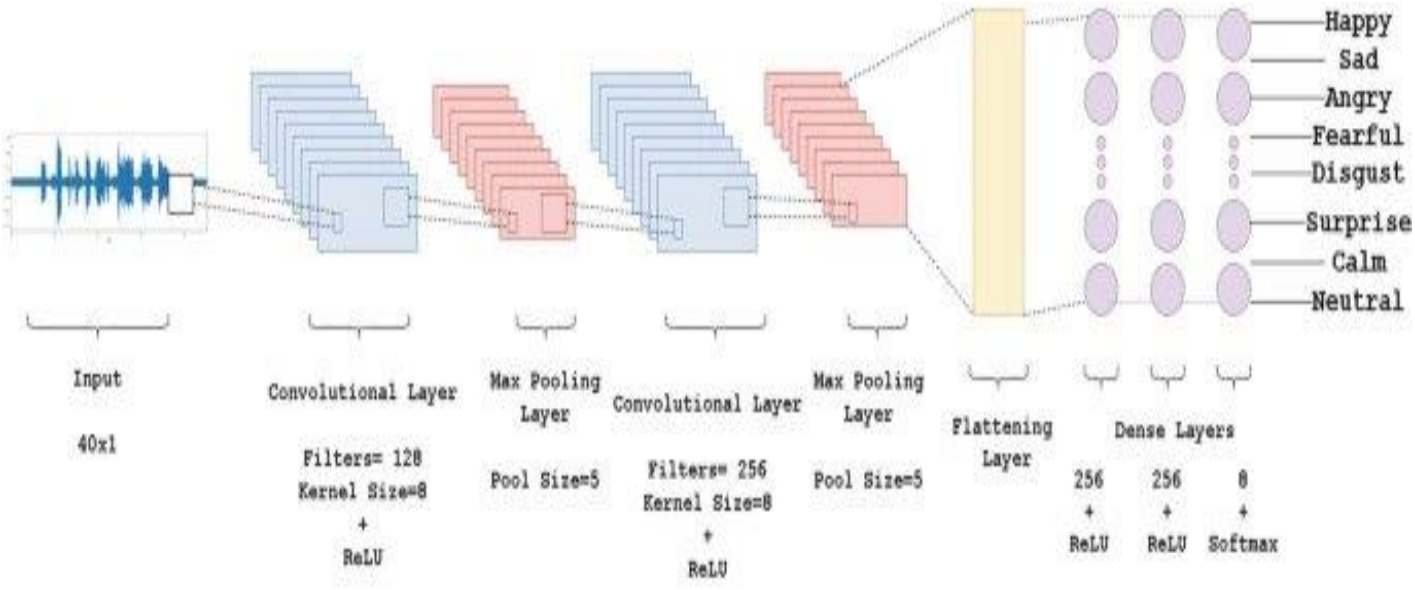


Figure 3-41: CNN model architecture

3.3.2.2 Model implementation

In order to implement this model, We used Model architecture fig 3.41 and its hyper parameter table 3.3 mentioned in the reference Results achieved in the reference is 70.83% on 8 emotions. But The reference did not mention the optimizer used in this model so we are going to try optimizers from related work [chapter two] to choose suitable optimizer, The optimizers that will try are ADAM and RMS. but on 6 emotions on RAVDESS Speech & Song and no augmentation happened on the dataset

Table 3-3 : Shows Hyper-parameters in the reference

Hyper-parameter	Value
Loss Function	Sparse Categorical cross entropy
Learning rate	0.00002
Dropout factor	0.3
Decay	0.9

In order to implement this model. we used an early stopping function to override overfitting. Any experiment or modification is tried at least 4 times to calculate the average test accuracy. So in order to change the test and train set each time we run the experiment we shuffled the data. So The shuffle parameter is needed to prevent non-random assignment to train and test set. With shuffle equal True we split the data randomly. and also we shuffled the data during training to help training converge fast and to prevent model from learning the order of training.

- **RMS:**

  Model achieved an average accuracy of 72.34% using hyperparameters in table 3.3 on RMS optimizer. Results of best accuracy achieved shown in fig 3.42.

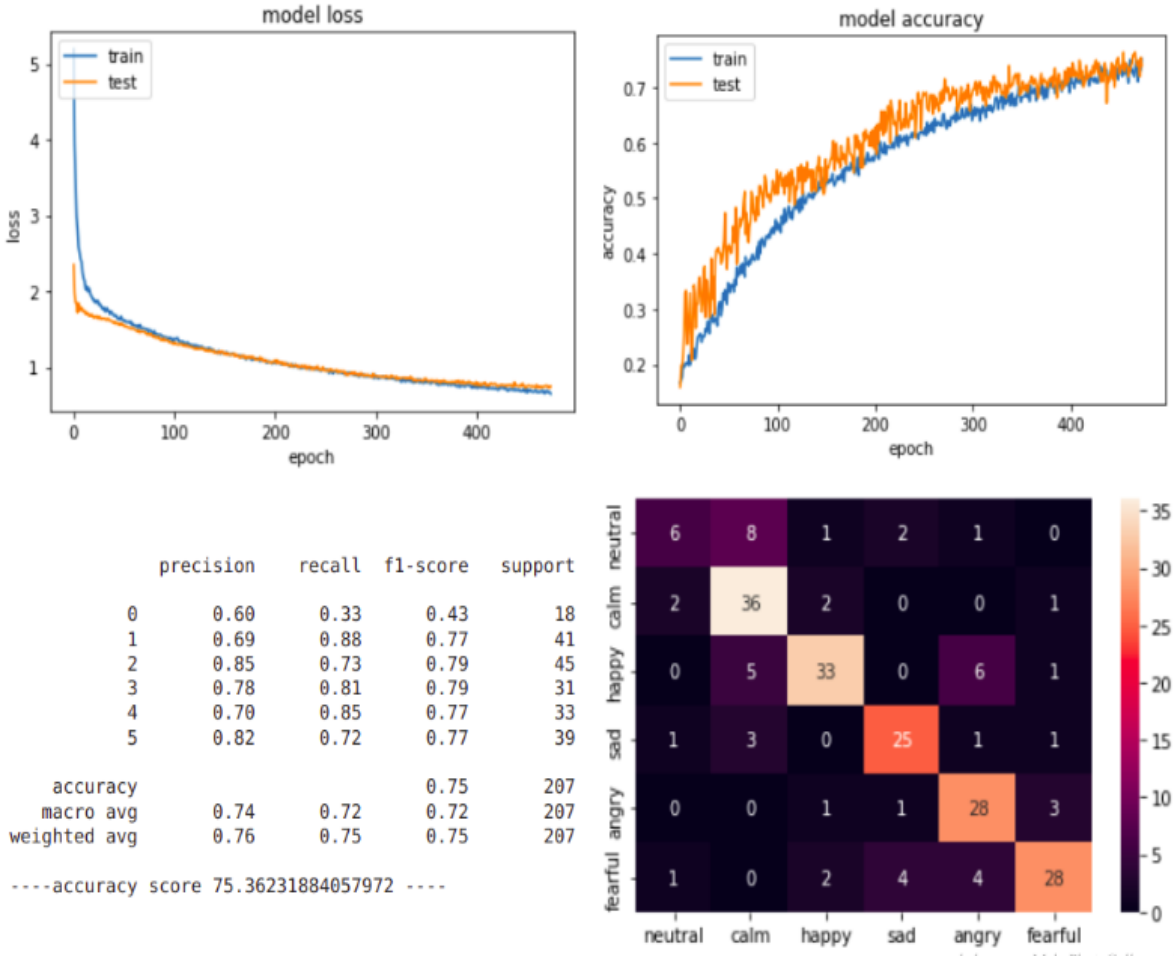


Figure 3-42 : Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **ADAM:**

  Using *ADAM* optimizer on this model with hyperparameters in table 3.3 achieved average accuracy 73.14 %, Fig 3.43 shows results of best accuracy.

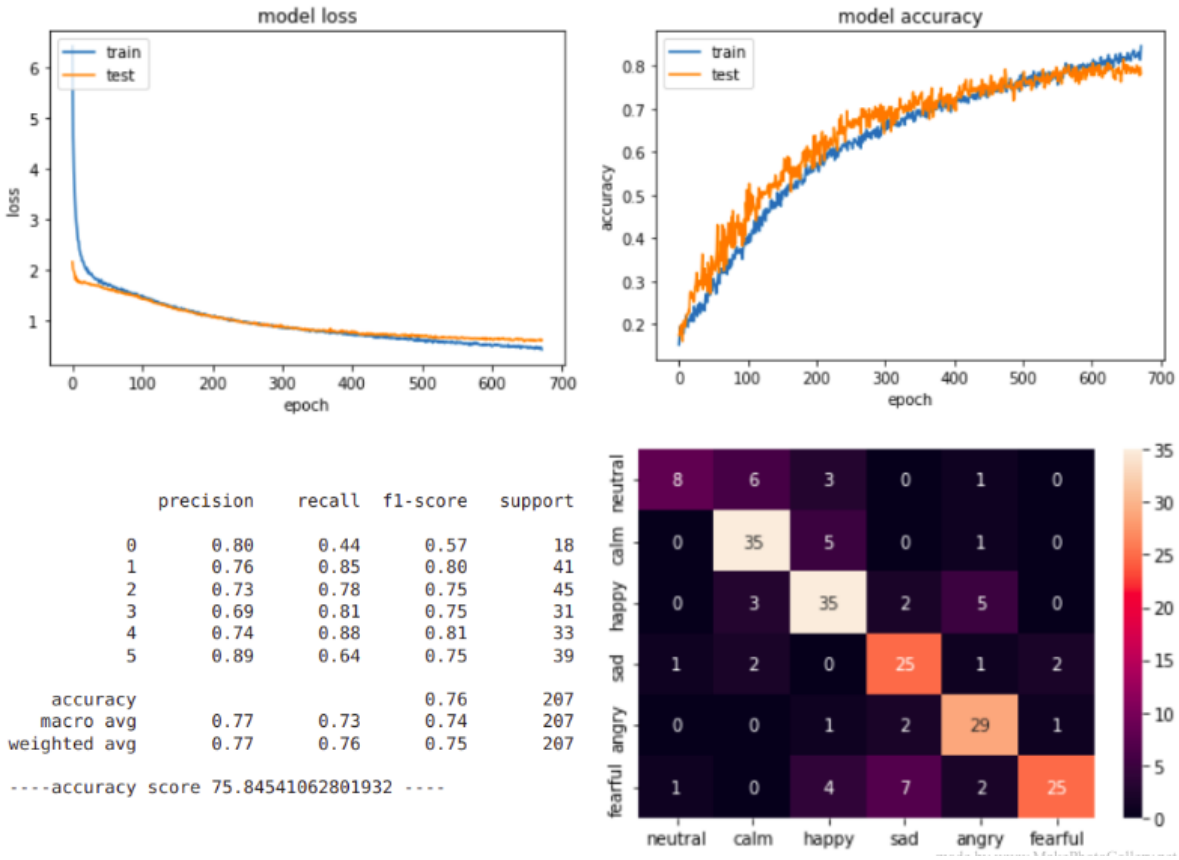


Figure 3-43: Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **Conclusion**:

After trying both Adam and RMS, we noticed that the difference in accuracy was in favor of Adam, but the difference was not big, the difference was (0.8%), so we will do other experiments to make sure that there is a clear and consistent difference in accuracy.

3.3.2.3 Model Modifications

The modifications we did on the model happened sequentially which means that we conclude in each modification that we will continue with this modification or without it according to the average accuracy achieved by it and also the complexity of the model after modification until we achieve a reasonable test accuracy.

- **Modification 1:**

  Changing learning rate from 0.00002 to 0.0002, increasing it, With optimizer RMS and ADAM.

- **ADAM:**

  Using learning rate 0.0002 instead of 0.00002 with Adam optimizer achieved an average accuracy 77.39 %. Fig 3.44 shows results of best accuracy.

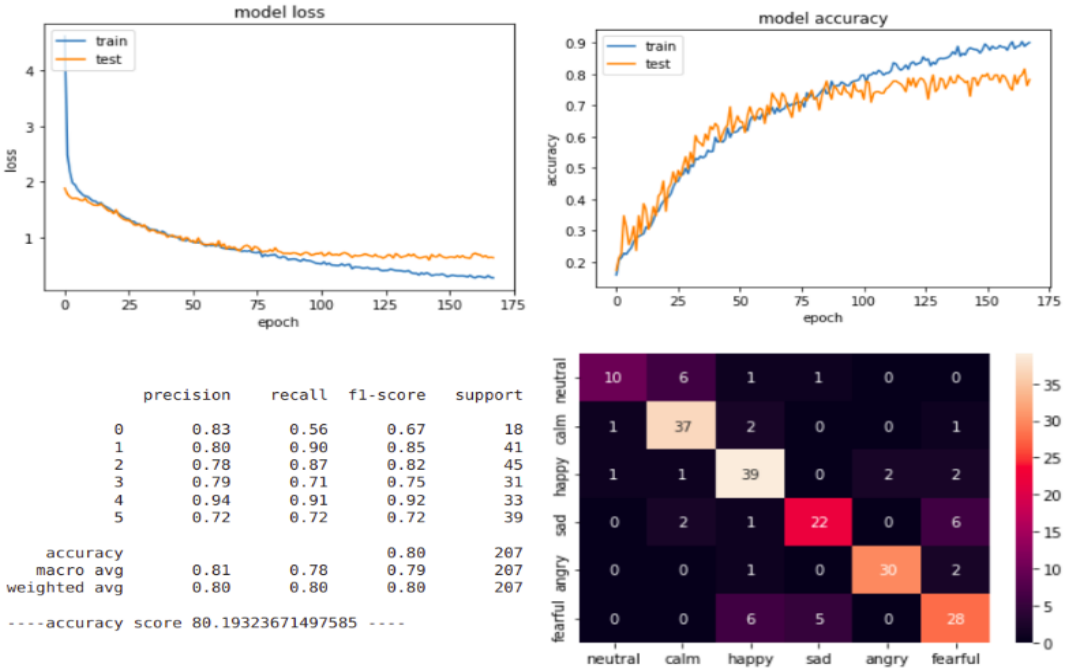


Figure 3-44: Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **RMS:**

  Using learning rate 0.0002 instead of 0.00002 with RMS optimizer achieved an average accuracy 75.55 %. Fig 3.45 shows results of best accuracy.

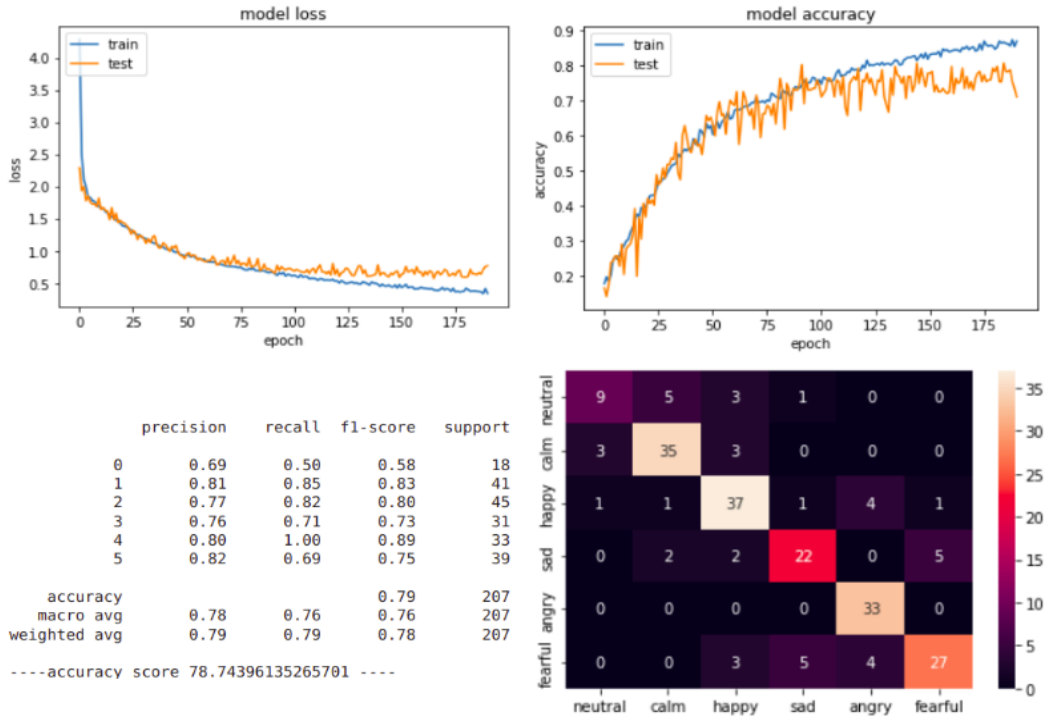


Figure 3-45: Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **Conclusion**:

  After trying both Adam and RMS with learning rate 0.0002, we noticed that accuracy increased due to increasing learning rate, but the difference still was not big between Adam and RMS, the difference was (1.8%), so we will continue with learning rate 0.0002 and both optimizers.

- **Modification 2:**

  Adding a new layer, Using 3 convolution layers instead of 2 convolution layers, The new layer added with filter 265, Kernel size 8 and drop out 0.3.

- **ADAM:**

  Adding a layer with Adam optimizer achieved an average accuracy 78.48 %. Fig 3.46 shows results of best accuracy.

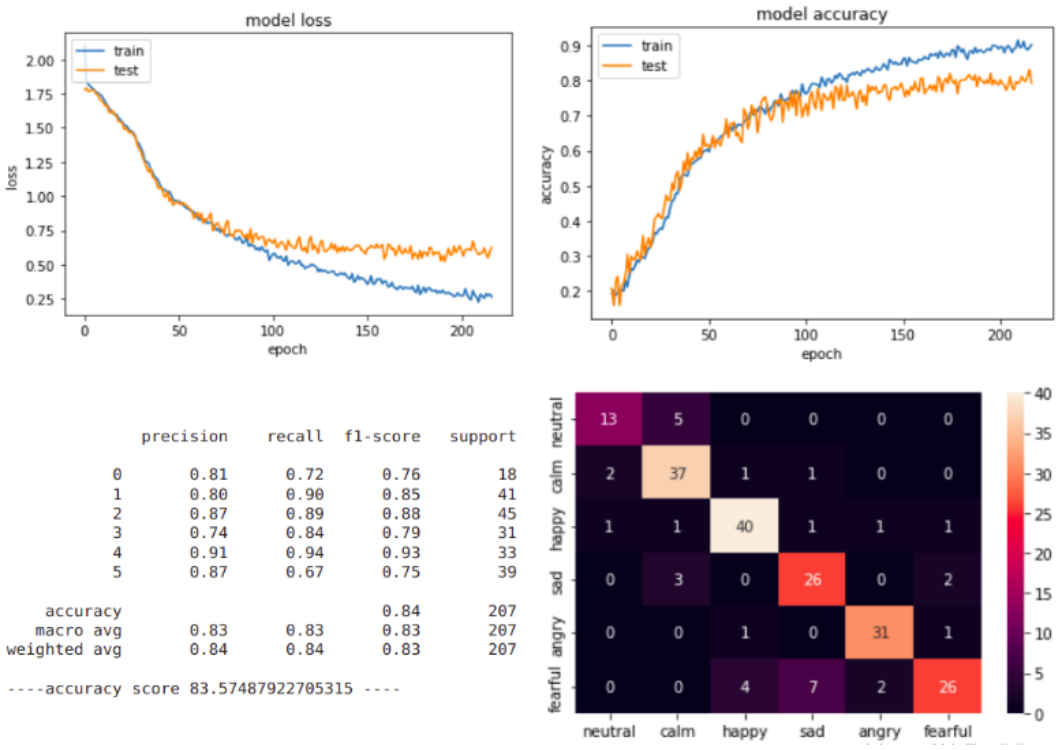


Figure 3-46 : Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **RMS:**

  Adding a layer with RMS optimizer achieved an average accuracy 74.59 %. Fig 3.47 shows results of best accuracy.

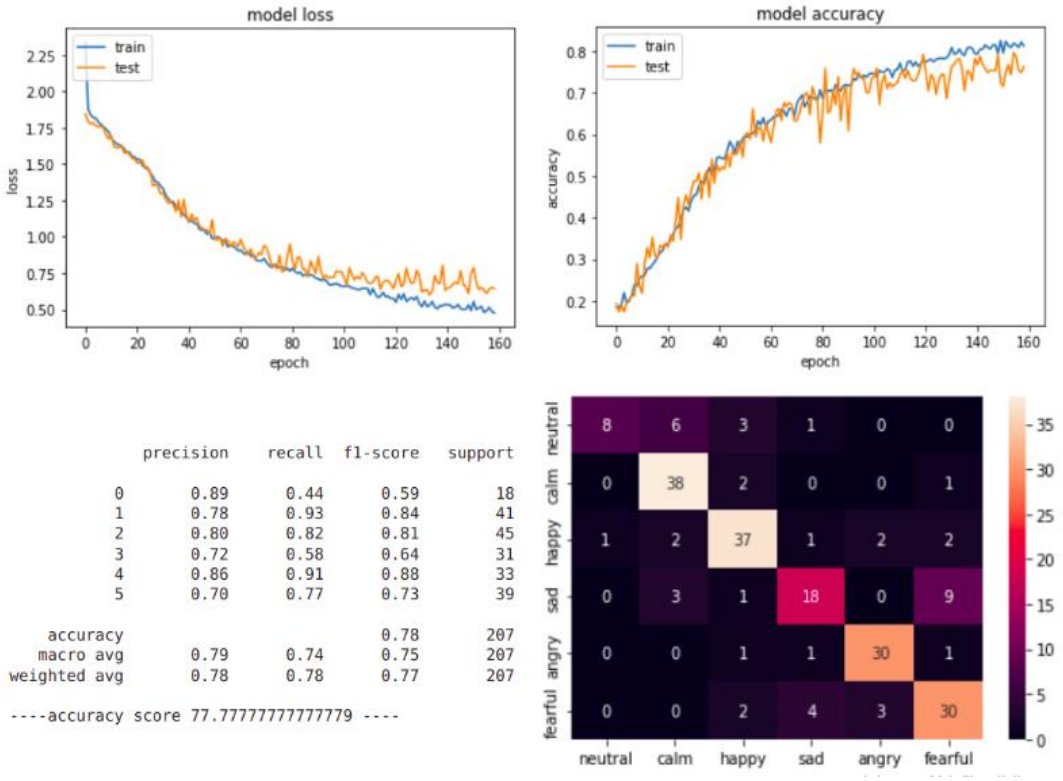


Figure 3-47: Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **Conclusion:**

  After trying both Adam and RMS with Adding a new layer, we noticed that accuracy increased by a small percentage (1%) with Adam optimizer and decreased by (1%) with RMS optimizer, but Due to complexity we will use 2 layers only and both optimizers.

- **Modification 3:**

  Removing two dense layers, use one dense layer instead of three

  - **ADAM:**

    Removing 2 dense layers with Adam optimizer achieved an average accuracy 78.73 %. Fig 3.48 shows results of best accuracy.

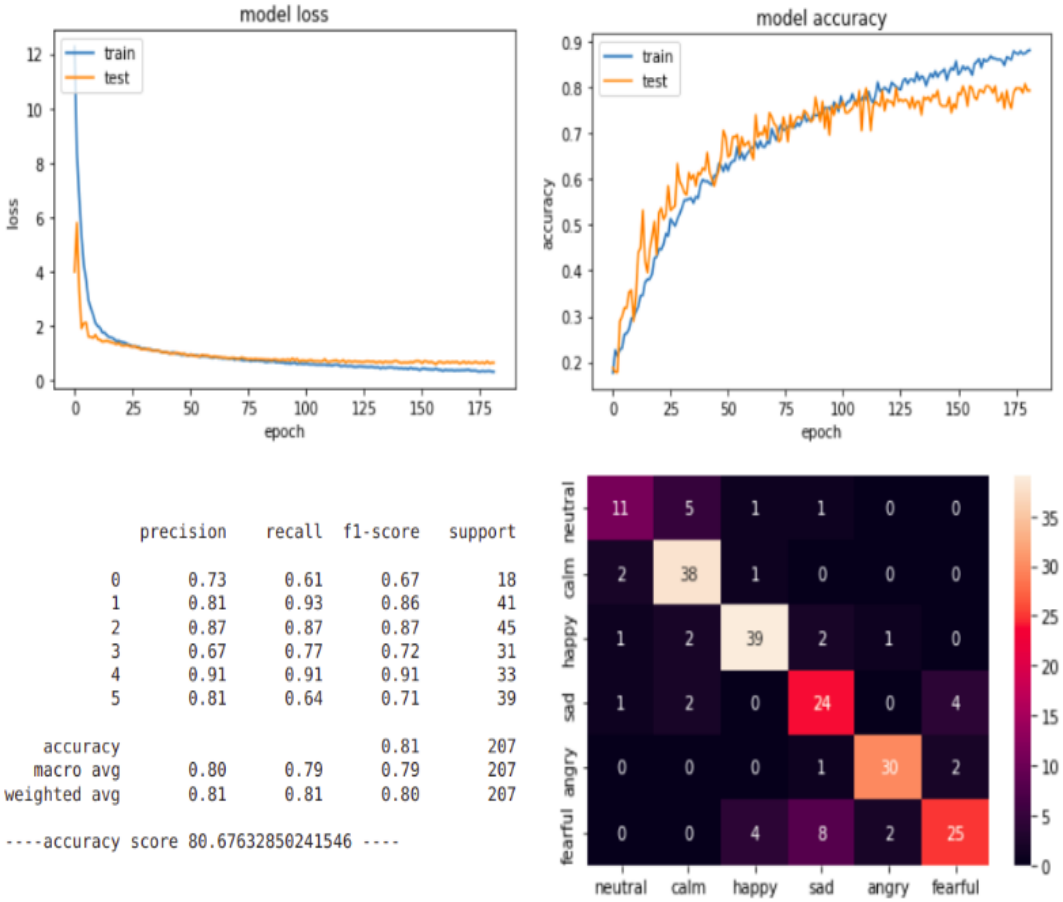


Figure 3-48 : Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **RMS:**

  Removing 2 dense layers with RMS optimizer achieved an average accuracy 80.57 %. Fig 3.49 shows results of best accuracy.

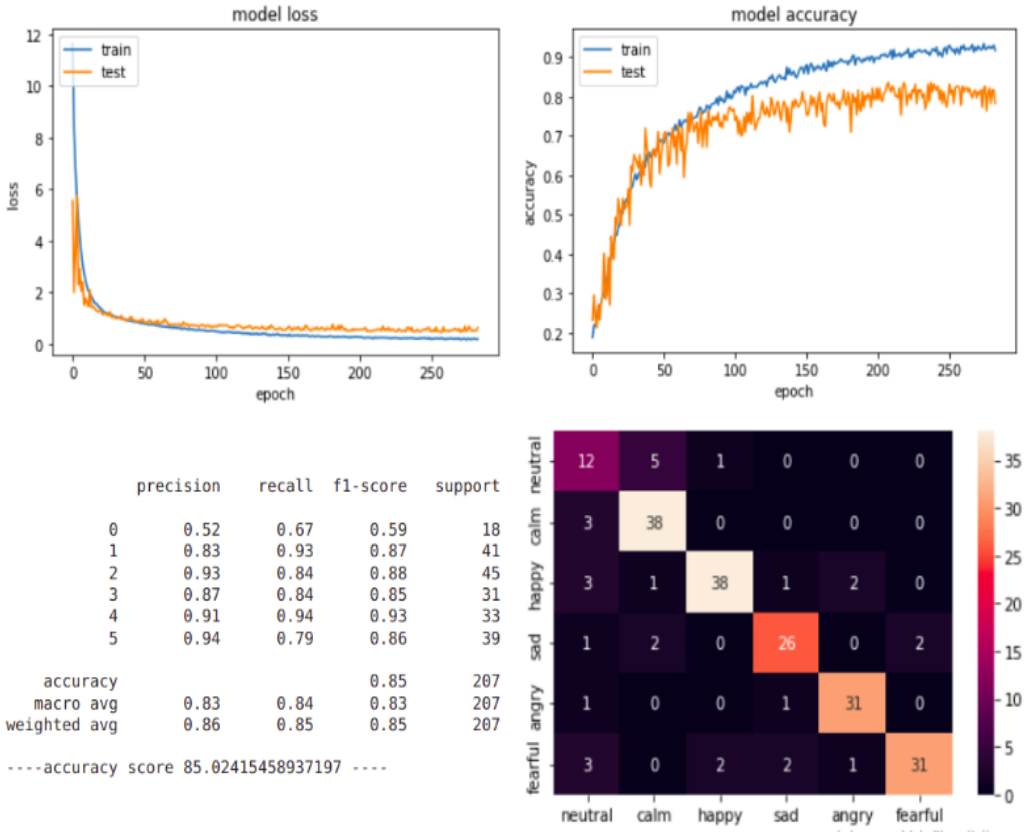


Figure 3-49 : Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **Conclusion:**

  After trying both Adam and RMS with Removing 2 dense layers, we noticed that accuracy increased by a small percentage (1.5%) with Adam optimizer and increased by (5%) with RMS optimizer, so we will continue with one dense layer and RMS optimizer.

- **Modification 4:**

  Changing Dropout ratio of the first convolutional layer using RMS optimizer.

  - **Dropout 0.1 instead of 0.3:**

    Achieved an average accuracy 77.38 %. Fig 3.50 shows results of best accuracy.

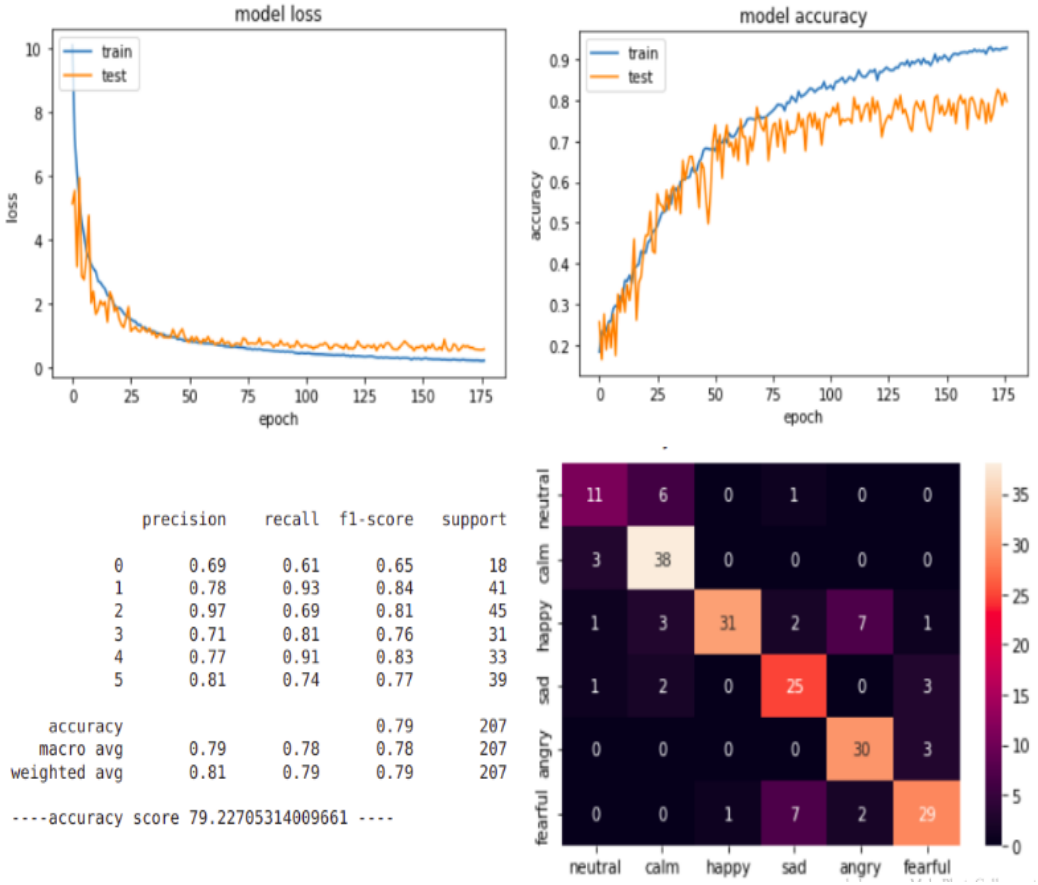


Figure 3-50: Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **Dropout 0 instead of 0.3:**

  Achieved an average accuracy 77.2 % fig 3.51 shows results of best accuracy.

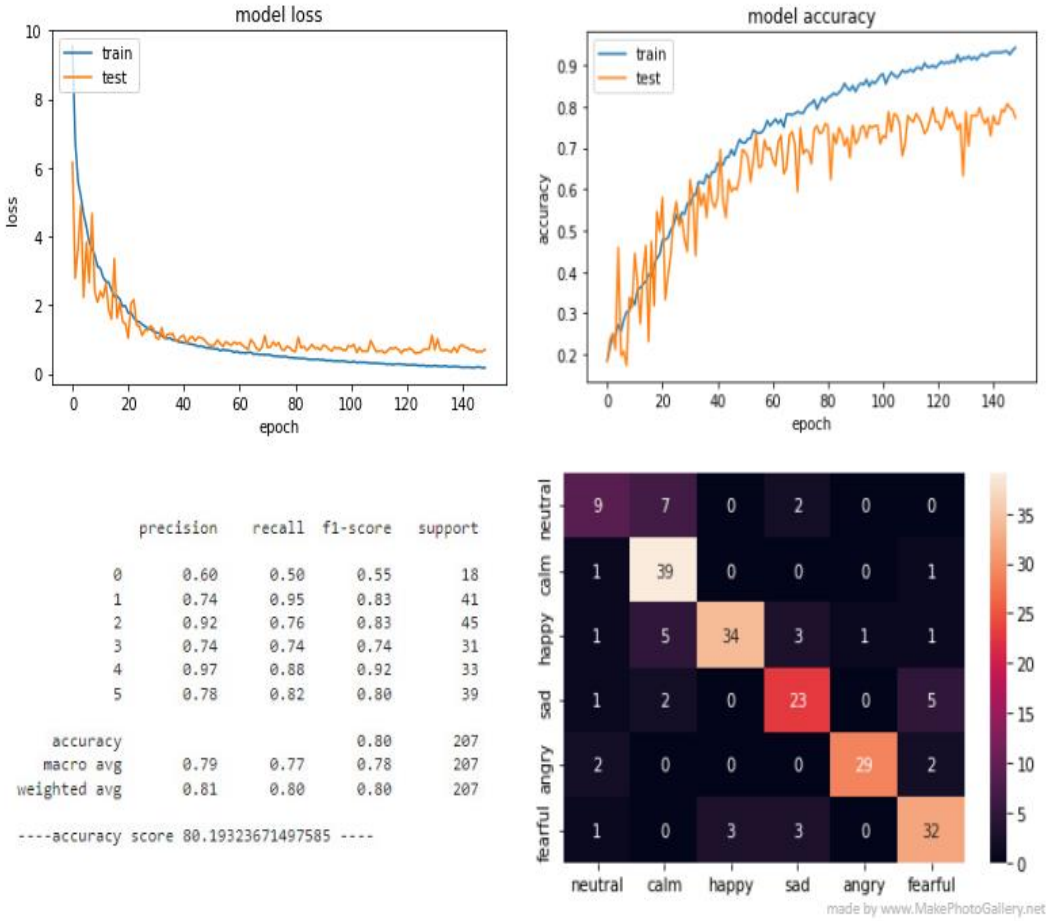


Figure 3-51 : Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **Conclusion:**

  By changing the dropout ratio of the first convolutional layer from default (0.3) to 0.1 and 0. we noticed that the accuracy decreased by (3%). So we conclude that we will continue with the default dropout ratio without changing it.

- **Modification 5:**

  Changing kernel size in all convolutional layers with RMS optimizer.

  - **Kernel size 12 instead of 8:**

    Achieved an average accuracy 82.002 %. Fig 3.52 shows results of best accuracy.

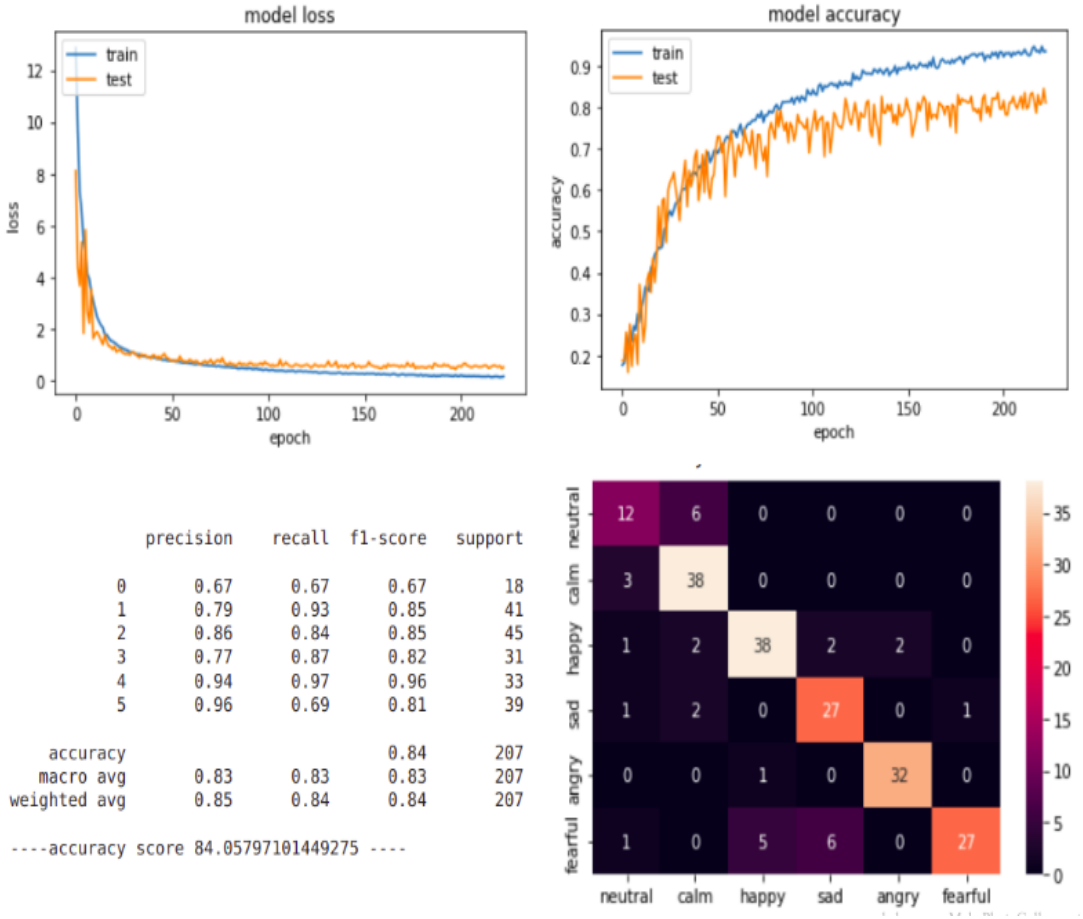


Figure 3-52 : Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **Kernel size 4 instead of 8:**

  Achieved an average accuracy 67.62 %. Fig 3.53 shows results of best accuracy.

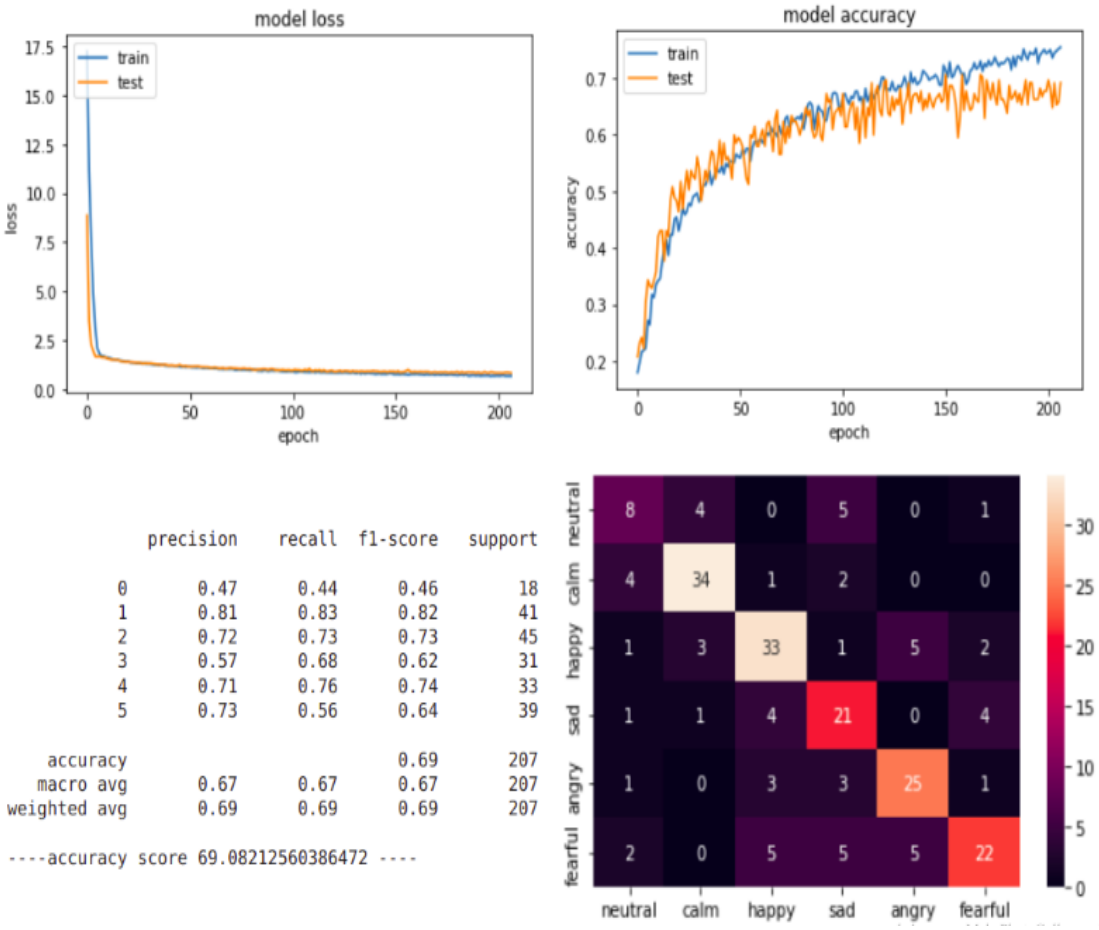


Figure 3-53 : Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **Kernel size 16 instead of 8:**

  Achieved an average accuracy 79.7 %. Fig 3.54 shows results of best accuracy.

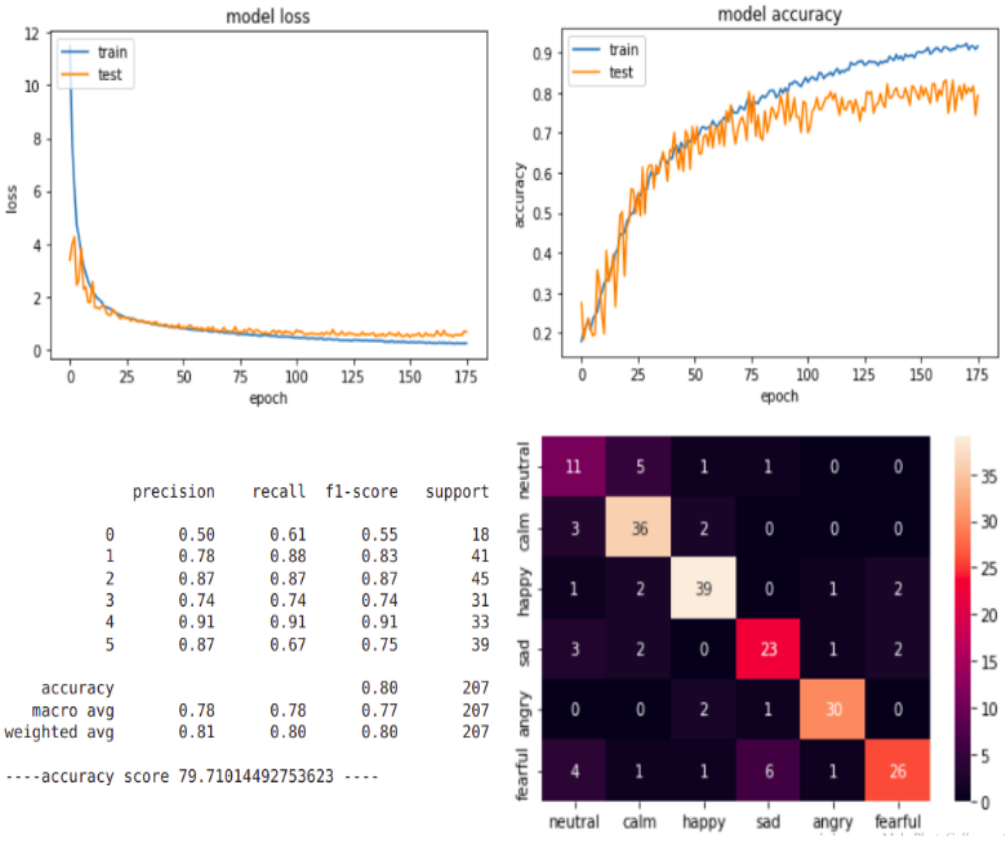


Figure 3-54: Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **Conclusion:**

  Changing kernel size to 12 instead of 8 increased accuracy by (2%), Changing kernel size to 4 instead of 8 decreased accuracy by (13%), Changing kernel size to 4 instead of 8 decreased accuracy by (1%), So we will continue with kernel size 12.

- **Modification 6:**

Changing filter size of the two convolutional layers from 128 filters which is the default in the first layer to 64 filters and from 256 filters in the second layer to 128 filters. it achieved an average accuracy of 78.8%. Fig 3.55 shows results of best accuracy.

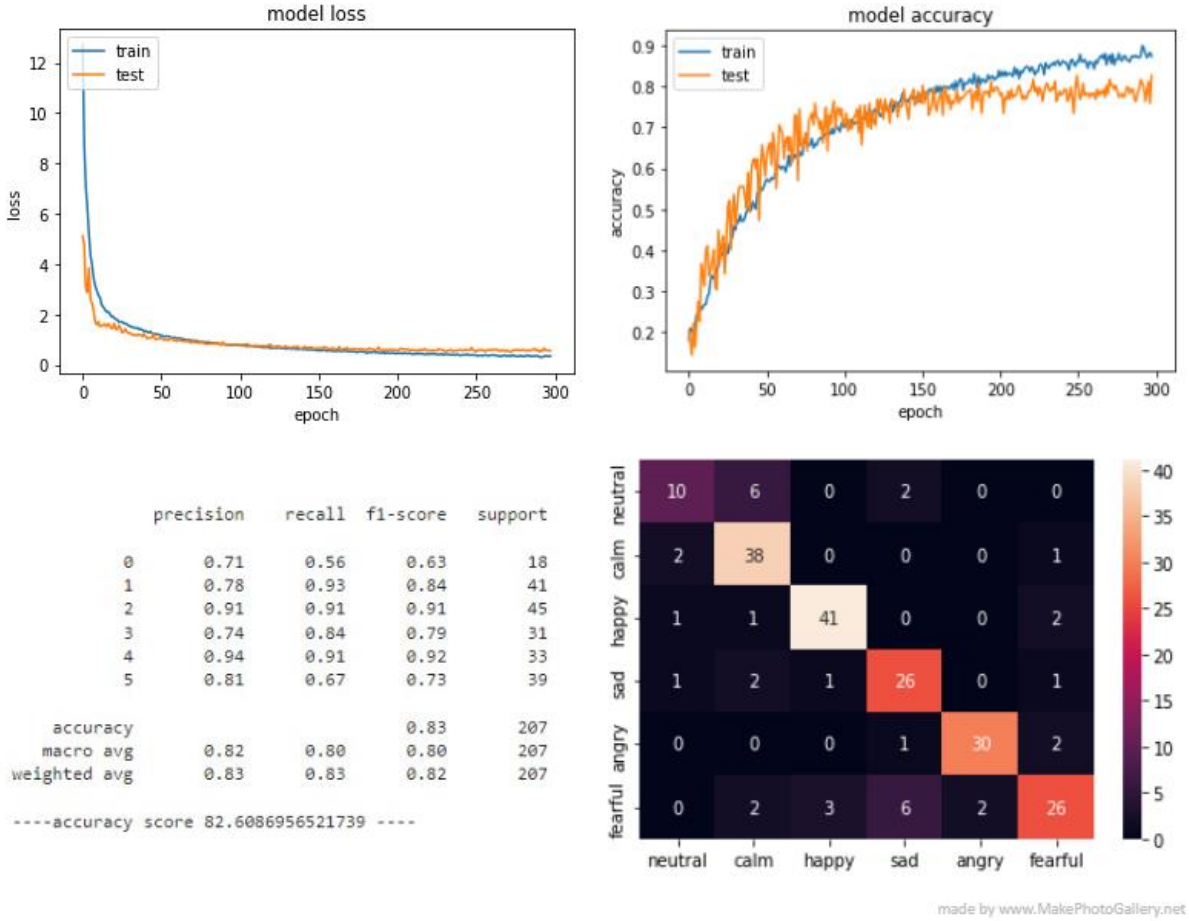


Figure 3-55 : Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **Conclusion:**

By changing filter size of the two convolutional layers from default to 64 and 128.we noticed that accuracy decreased by approximately 3%. So we conclude that we will continue with the default filter size without changing the filter size.

- **Modification 7:**

Augmentation on dataset (train only). The type of augmentation that happens on the dataset is time stretch by 0.5 then time stretch by 1.5 and finally pitch shift by 2. This modification achieved an average accuracy 82.2%. Fig 3.56 shows results of best accuracy.

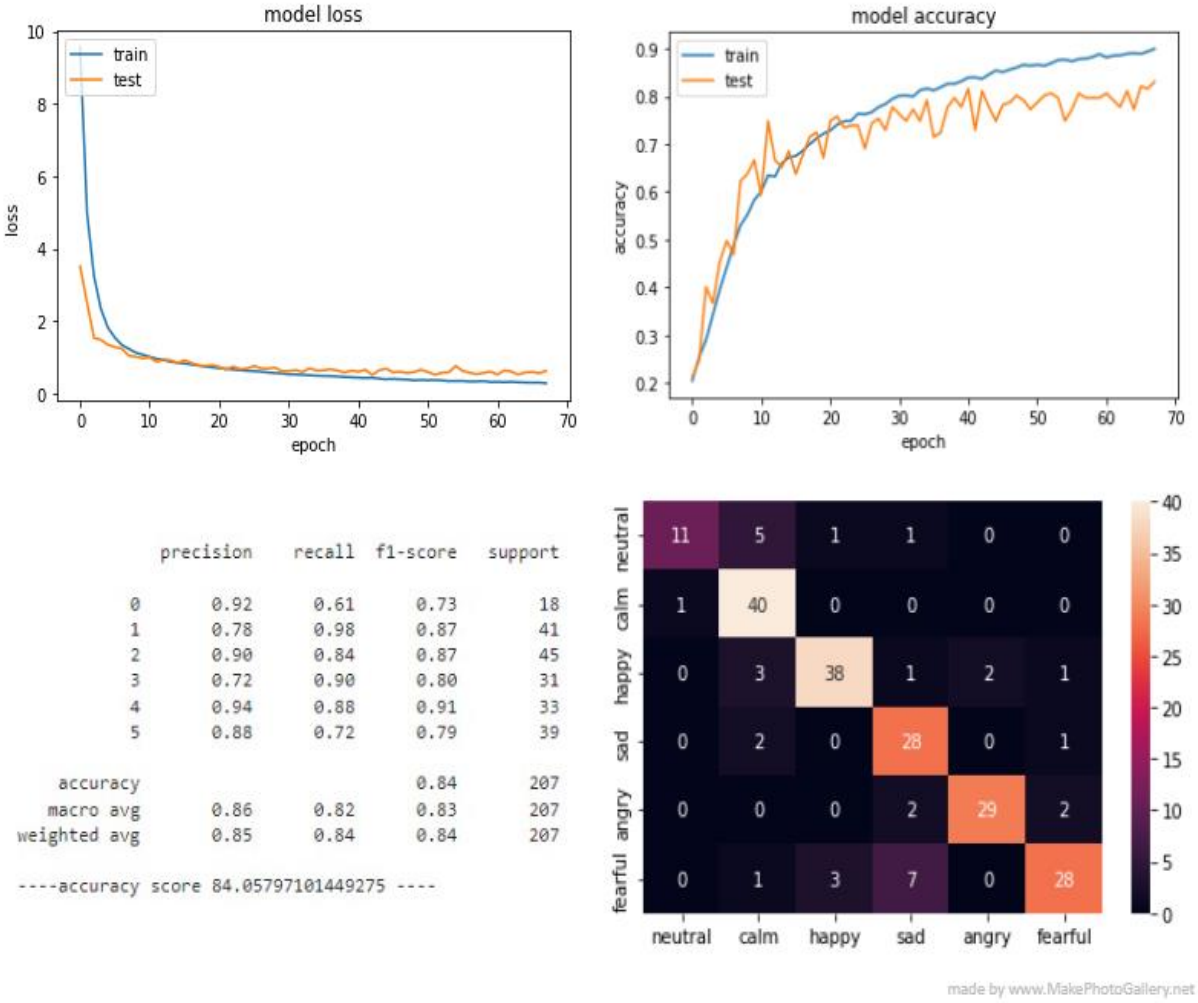


Figure 3-56 : Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **Conclusion:**

By adding augmentation we noticed that it didn't significantly affect accuracy. So we will continue without augmentation.

- **Modification 8:**

Adding Batch Normalization layers after convolution layers but this modification needs changing in dropout to work correctly so we will change dropout in first convolution layer to 0.1 instead of 0.3, This modification Achieved an average accuracy 78.08 %. Fig 3.57 shows results of best accuracy.

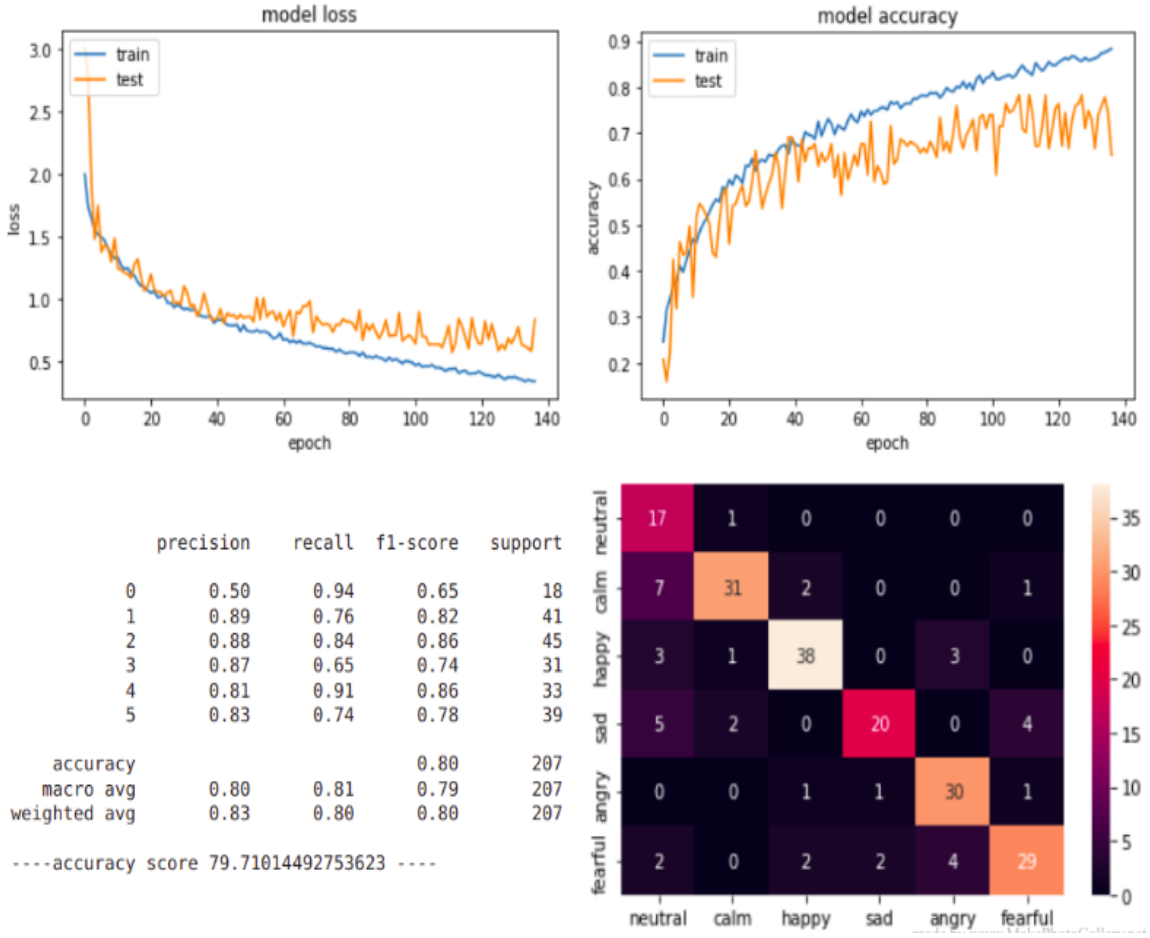


Figure 3-57: Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **Conclusion:**

Adding Batch normalization layers after convolution layers decreased accuracy by (4%). So we will continue without batch normalization.

- **Modification 9:**

Adding Batch Normalization layers after convolution layers and after input layer but this modification needs changing in dropout to work correctly so we will change dropout in first convolution layer to 0.1 instead of 0.3 , This modification Achieved an average accuracy 84.42 % fig 3.58 shows results of best accuracy.

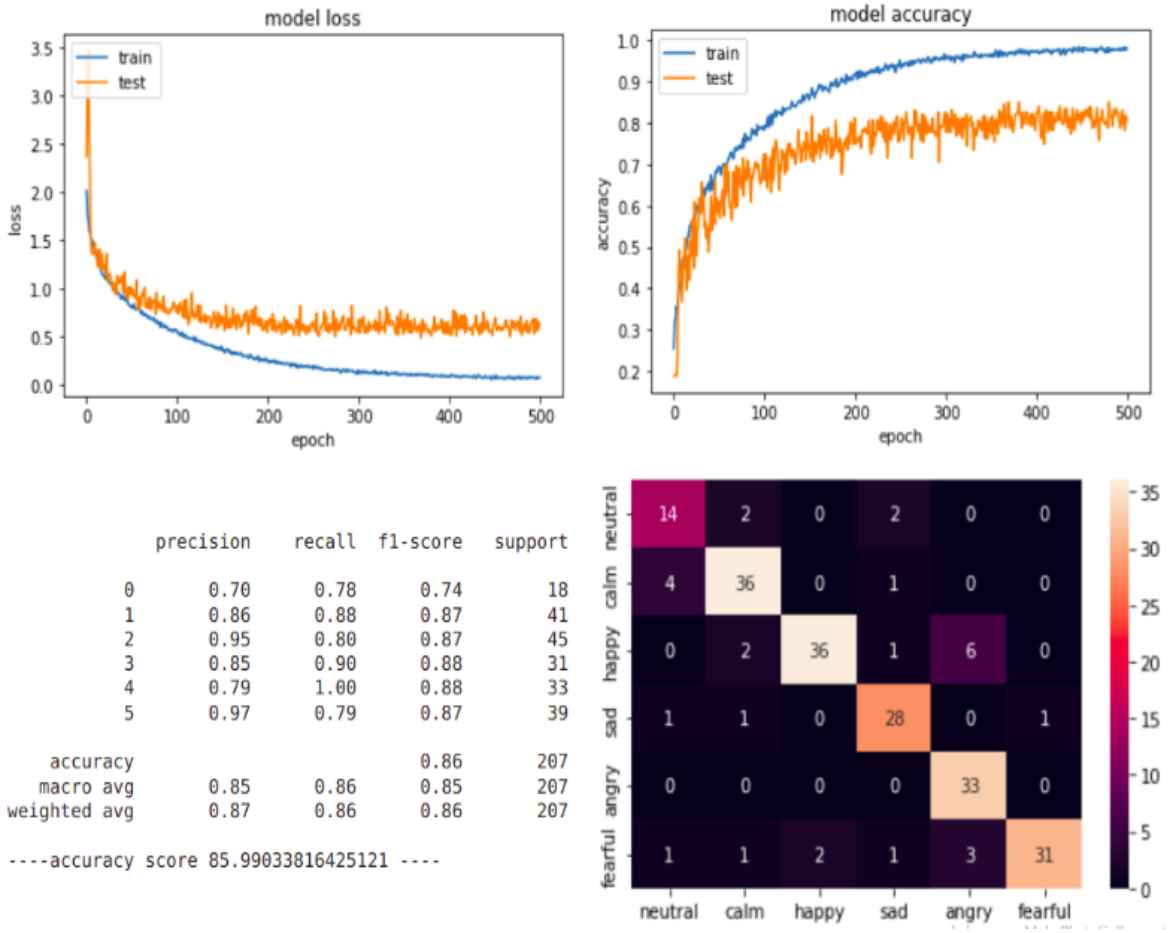


Figure 3-58 : Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **Conclusion:**

Adding Batch normalization layers after convolution layers and input layer increased accuracy by (2.4%). So we will continue with batch normalization.

- **Modification 10:**

After making a lot of modifications, we arrived at a model that achieves an accuracy of 84.5% on average. Now we will try Adam optimizer on this model after the modifications, This modification Achieved an average accuracy 85.26 %. Fig 3.59 shows results of best accuracy.

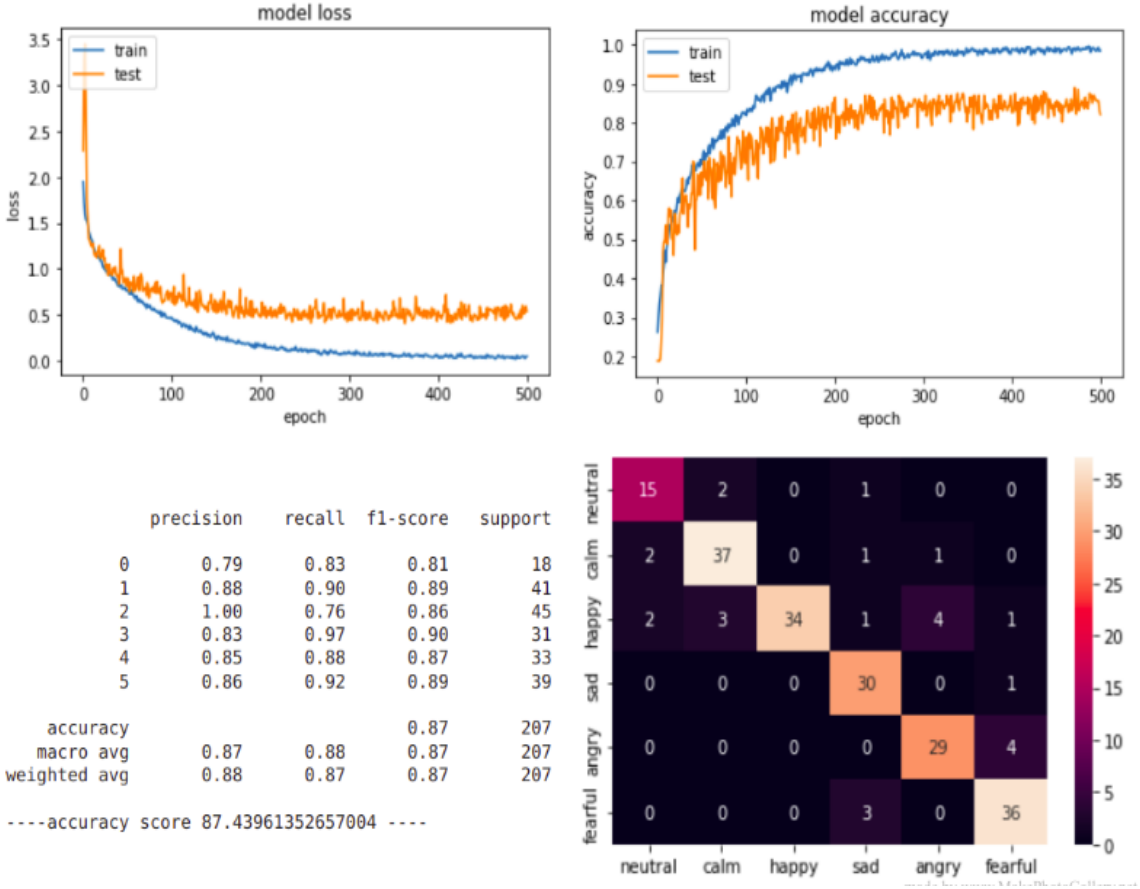


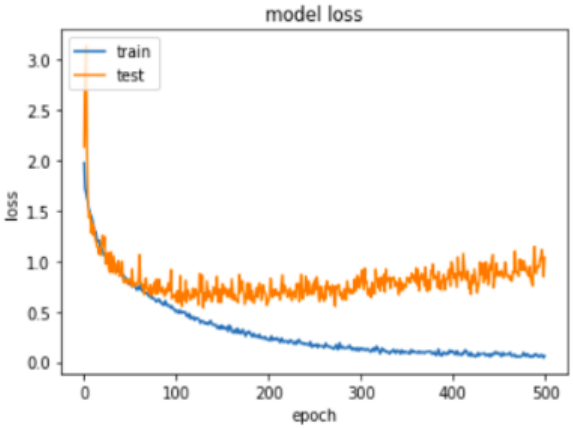
Figure 3-59: Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

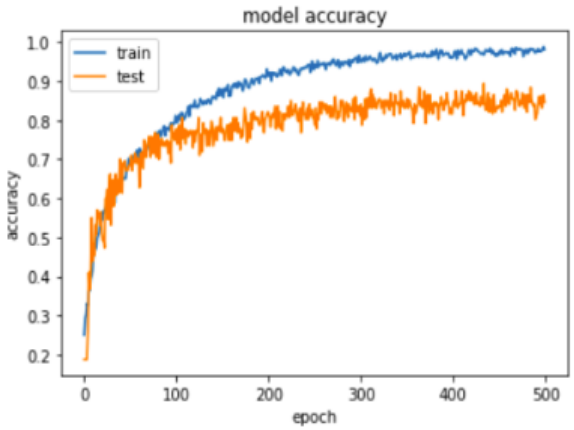
- **Conclusion:**

using Adam optimizer increased accuracy by (1%) on average. So we will continue with Adam optimizer.

- **Modification 11:**

Changing activation function to tanh instead of relu, This modification Achieved an average accuracy 80.79 %. Fig 3.60 shows results of best accuracy.





	precision	recall	f1-score	support
0	0.70	0.89	0.78	18
1	0.85	0.95	0.90	41
2	0.97	0.78	0.86	45
3	0.73	0.97	0.83	31
4	0.93	0.82	0.87	33
5	0.91	0.74	0.82	39
accuracy			0.85	207
macro avg	0.85	0.86	0.84	207
weighted avg	0.87	0.85	0.85	207

----accuracy score 85.02415458937197 ----

Figure 3-60: Shows model loss, accuracy plot, classification report and heatmap of confusion matrix respectively

- **Conclusion:**

Using activation function tanh decreased accuracy by (5%) on average. So we will continue with Relu activation function.

3.3.2.4 Conclusion

After making this modifications model achieved accuracy 85.25% on average and 87.43% on best

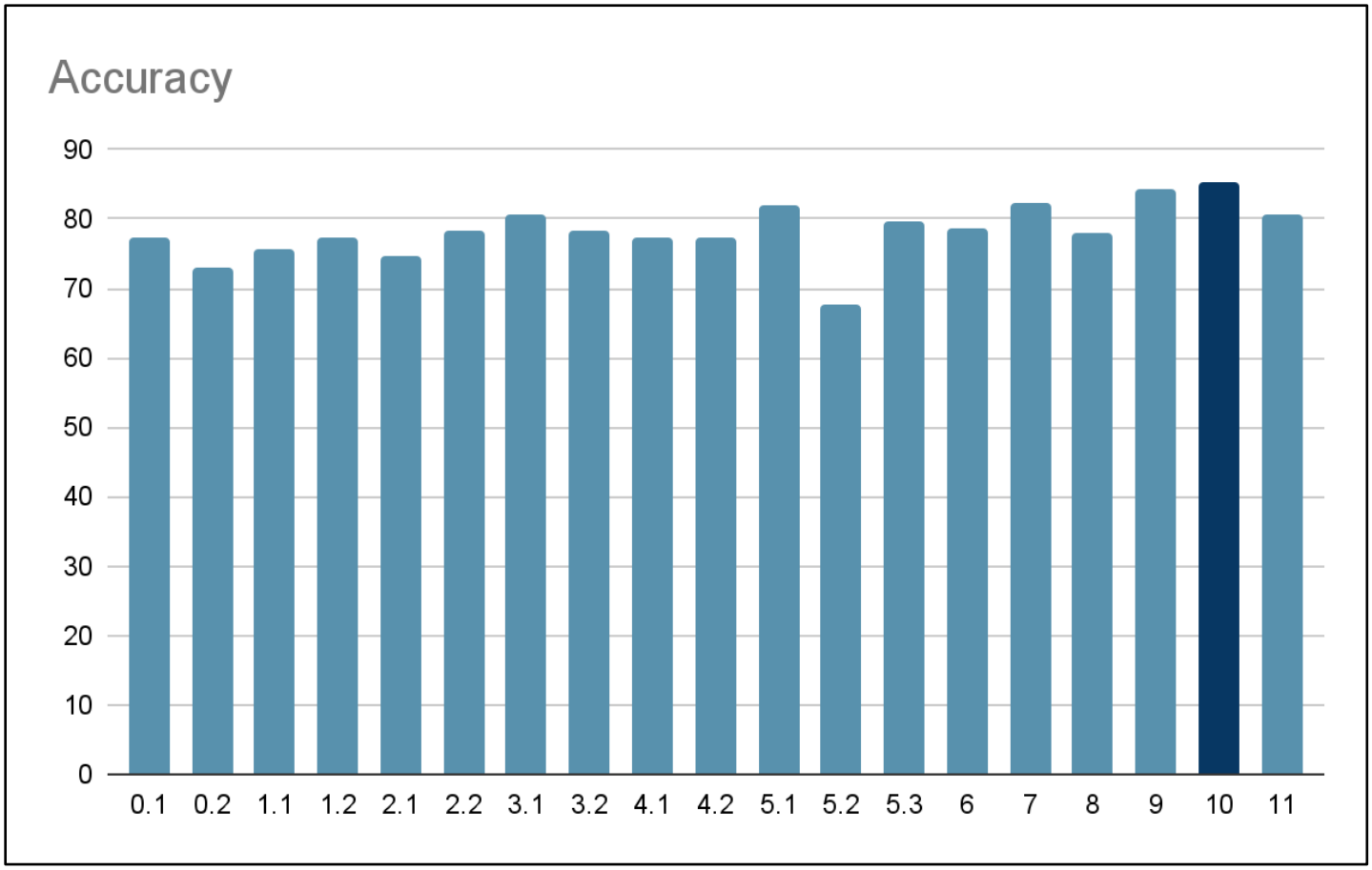


Figure 3-61: Chart shows accuracy of modifications of CNN model

Table 3-4 : Show modifications on CNN model

modification	Dropout	# of Dense layer	Augmentation	batch normalization	# of conv layers	Kernel size	Optimizer	lr	Activation	# of filters	Accuracy
0.1	0.3	3	NO	NO	2	8	RMS	0.00002	Relu	128,256	77.34%
0.2	0.3	3	NO	NO	2	8	Adam	0.00002	Relu	128.256	73.14%
1.1	0.3	3	NO	NO	2	8	RMS	0.0002	Relu	128,256	75.55%
1.2	0.3	3	NO	NO	2	8	Adam	0.0002	Relu	128,256	77.39%
2.1	0.3	3	NO	NO	3	8	RMS	0.0002	Relu	128,256	74.59%
2.2	0.3	3	NO	NO	3	8	Adam	0.0002	Relu	128,256	78.48%
3.1	0.3	1	NO	NO	2	8	RMS	0.0002	Relu	128,256	80.57%
3.2	0.3	1	NO	NO	2	8	Adam	0.0002	Relu	128,256	78.37%
4.1	0.1 0.3	1	NO	NO	2	8	RMS	0.0002	Relu	128,256	77.38%
4.2	0 0.3	1	NO	NO	2	8	RMS	0.0002	Relu	128,256	77.2%
5.1	0.3	1	NO	NO	2	12	RMS	0.0002	Relu	128,256	82%
5.2	0.3	1	NO	NO	2	4	RMS	0.0002	Relu	128,256	67.62%
5.3	0.3	1	NO	NO	2	16	RMS	0.0002	Relu	128,256	79.7%
6	0.3	1	NO	NO	2	12	RMS	0.0002	Relu	64,128	78.8%

7	0.3	1	YES	NO	2	12	RMS	0.0002	Relu	128,256	82.2%
8	0.1 0.3	1	NO	YES	2	12	RMS	0.0002	Relu	128,256	78.08%
9	0.1 0.3	1	NO	YES	2	12	RMS	0.0002	Relu	128,256	84.42%
10	0.1 0.3	1	NO	YES	2	12	Adam	0.0002	Relu	128,256	85.26%
11	0.1 0.3	1	NO	YES	2	12	Adam	0.0002	tanh	128,256	80.79%

CNN MODEL ON SAVEE DATASET:

Now we will train and test our final model on a different data set (SAVVE) that mentioned in [chapter two], the model achieved accuracy 79.16 % on average and 83.33% on best. fig 3.62 shows results of best accuracy.

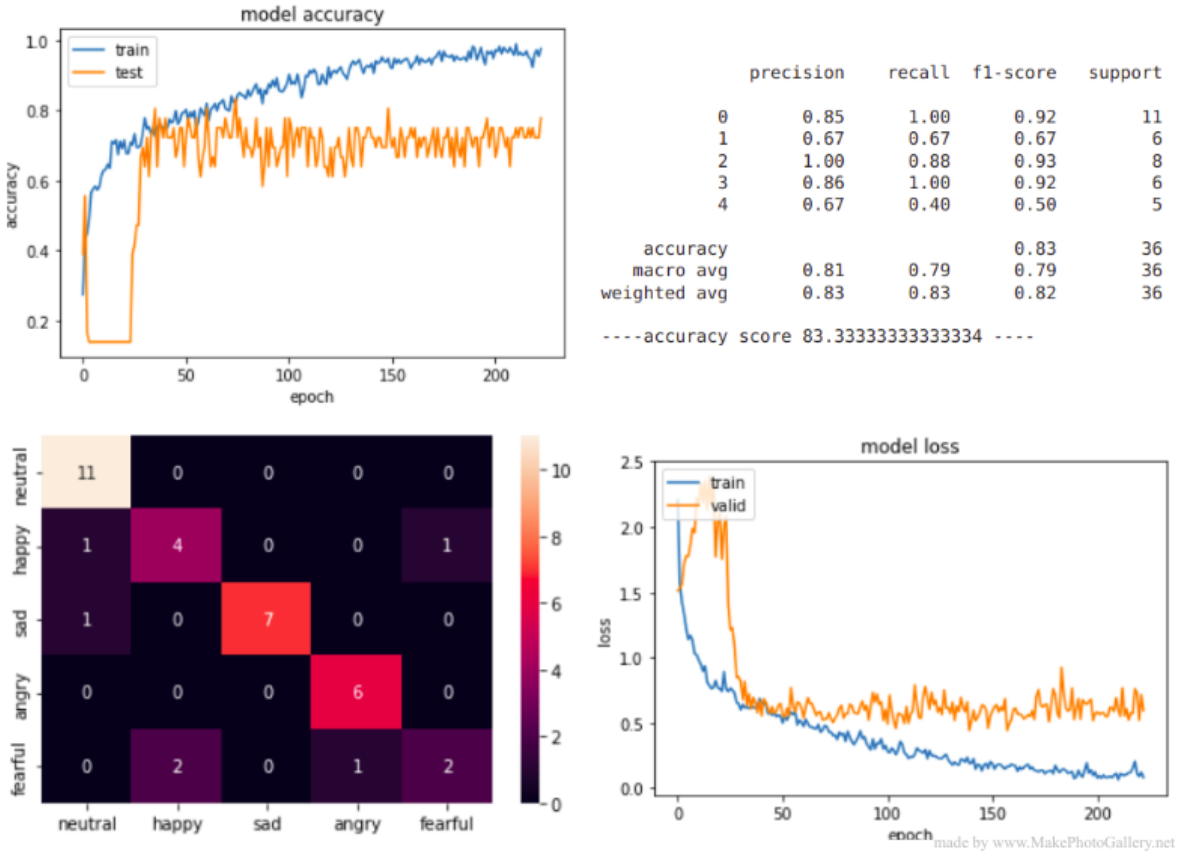


Figure 3-62: Shows model accuracy plot, classification report, heatmap of confusion matrix and loss plot respectively

3.3.3 Machine learning models

We tried different models like a supervised machine learning algorithm such as SVM and also we tried multilayer perceptron (MLP) on the RAVDESS data set on 8 emotions.

- **SVM**

  We trained Support vector machine model on RAVDESS speech and song dataset after scaling the data because scaling makes it easy for a model to learn and understand the problem due to it make values of the features are closer to each other so the algorithm get trained well and faster instead of the data set where the data points or features values have high differences with each other will take more time to understand the data and the accuracy will be lower. In table 3.5 we will show results of SVM using different kernels. Figure 3.63, Figure 3.64, Figure 3.65, Figure 3.66 shows Linear, RBF, Polynomial 3 and polynomial 4 respectively.

Table 3-5 : SVM model using different kernels

SVM kernel	Train accuracy	Test accuracy
Linear	66.61%	59.88%
RBF	79.65%	71.08%
Polynomial = 3	77.61%	65.58%
polynomial = 4	96.61%	53.15%

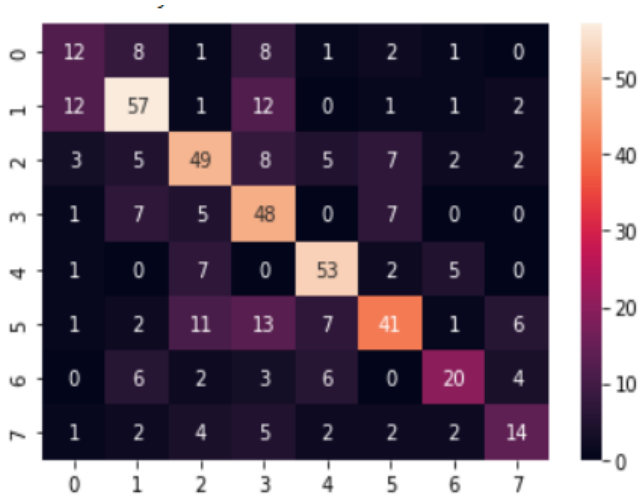


Figure 3-64 : Linear heatmap confusion matrix

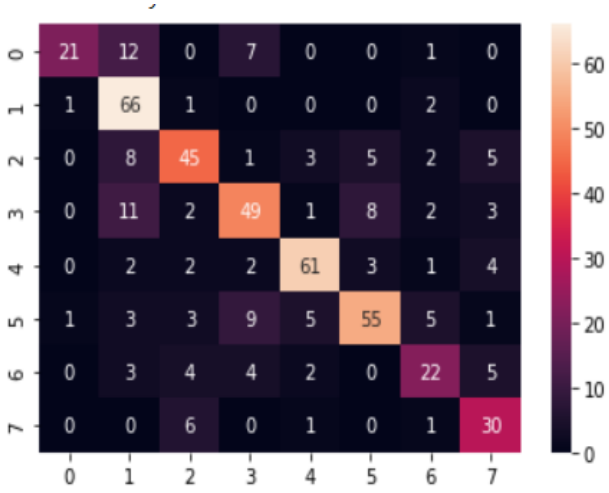


Figure 3-63 : RBF kernel heatmap confusion matrix

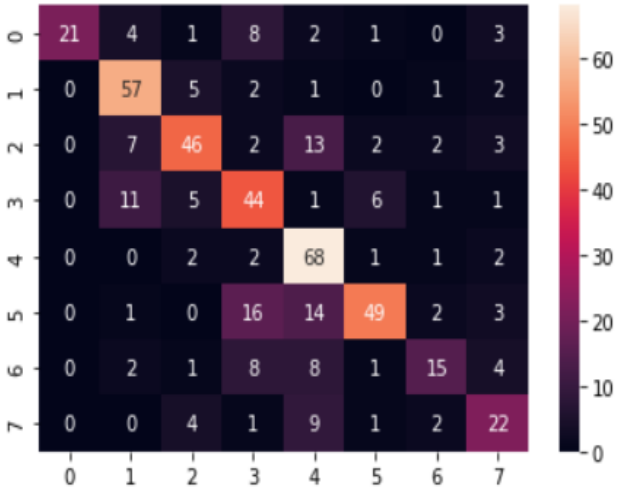


Figure 3-65 : Polynomial = 3 heatmap of confusion matrix

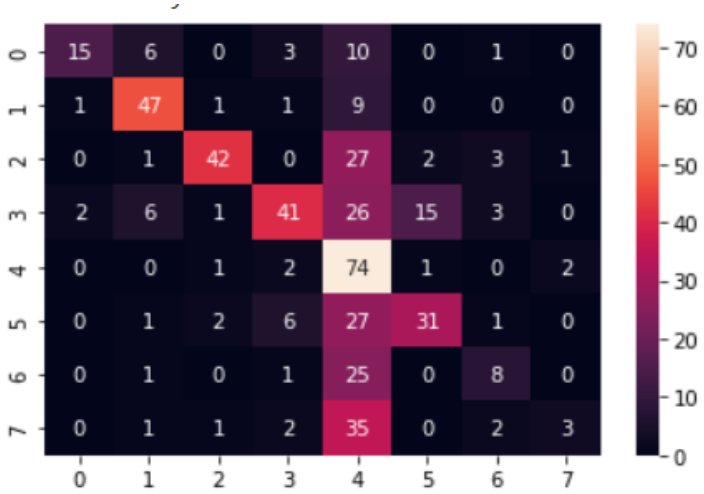


Figure 3-66 : Polynomial = 4 heatmap of confusion matrix

- **MLP**

We trained MLP model on RAVDESS speech only dataset with number of layers equal 200 using relu as an activation function. It achieved 52.77% as a test accuracy. Figure 3.67 shows confusion matrix of MLP model.

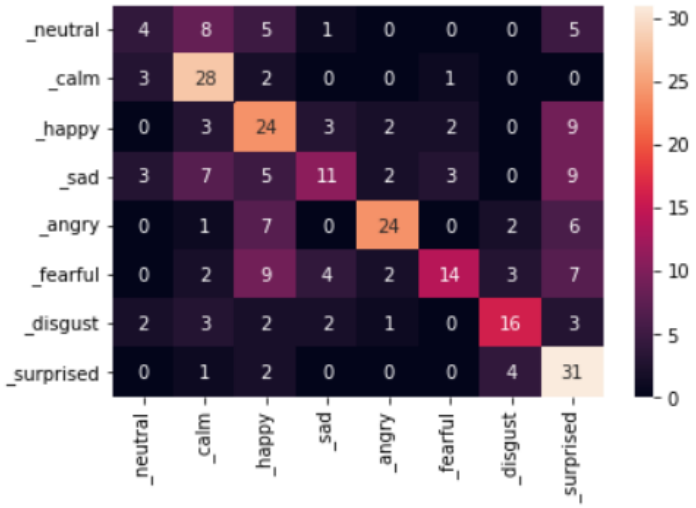


Figure 3-67 : MLP model heatmap of confusion matrix

- **Conclusion:**

After we trained SVM and MLP models on the RAVDESS dataset we noticed that the highest accuracy achieved is by using SVM (RBF kernel) which is 71.08% testing accuracy.

3.3.4 Model Work Conclusion

After we tried a machine learning model like SVM model with different kernels and also trying multilayer perceptron (MLP) in section 3.3.3 we concluded that the highest accuracy achieved is 71.08% using SVM which is not enough to be our test accuracy and also we cannot improve the model in order to achieve a high enough accuracy to be our test accuracy. Therefore we followed deep learning algorithms. So we tried two different models, two parallel CNN (3.3.1), CNN (3.3.2) and made many modifications to them to achieve high accuracy, the outputs were as shown in table 3.6.

Table 3-6 : Show accuracy comparison between reference model and our model

Model	Reference accuracy	Implementation of reference model	Our model
Two parallel CNN	75%	73.23%	80.6%
CNN	70.38%	72.15%	87%

From the result shown in table 3.6 we obtained that our modifications on two parallel CNN model improved accuracy by 6% approximately. and our modifications on CNN model improved accuracy by 17% approximately. We chose CNN model to be our final model due to the higher accuracy and less complication from the Two parallel CNN model.

OUR FINAL MODEL ARCHITECTURE:

Our final model is 2 convolution layers, First convolution layer with filter size 128 and kernel size 12 and dropout ratio 0.1 and batch normalization after it, second convolution layer with filter size 256 and kernel size 12 and dropout ratio 0.3 and batch normalization after it, and batch normalization after input layer, using 1 dense layer and adam optimizer with learning rate 0.0002 and relu activation function. Figure 3.68 shows a diagram of our final model architecture.

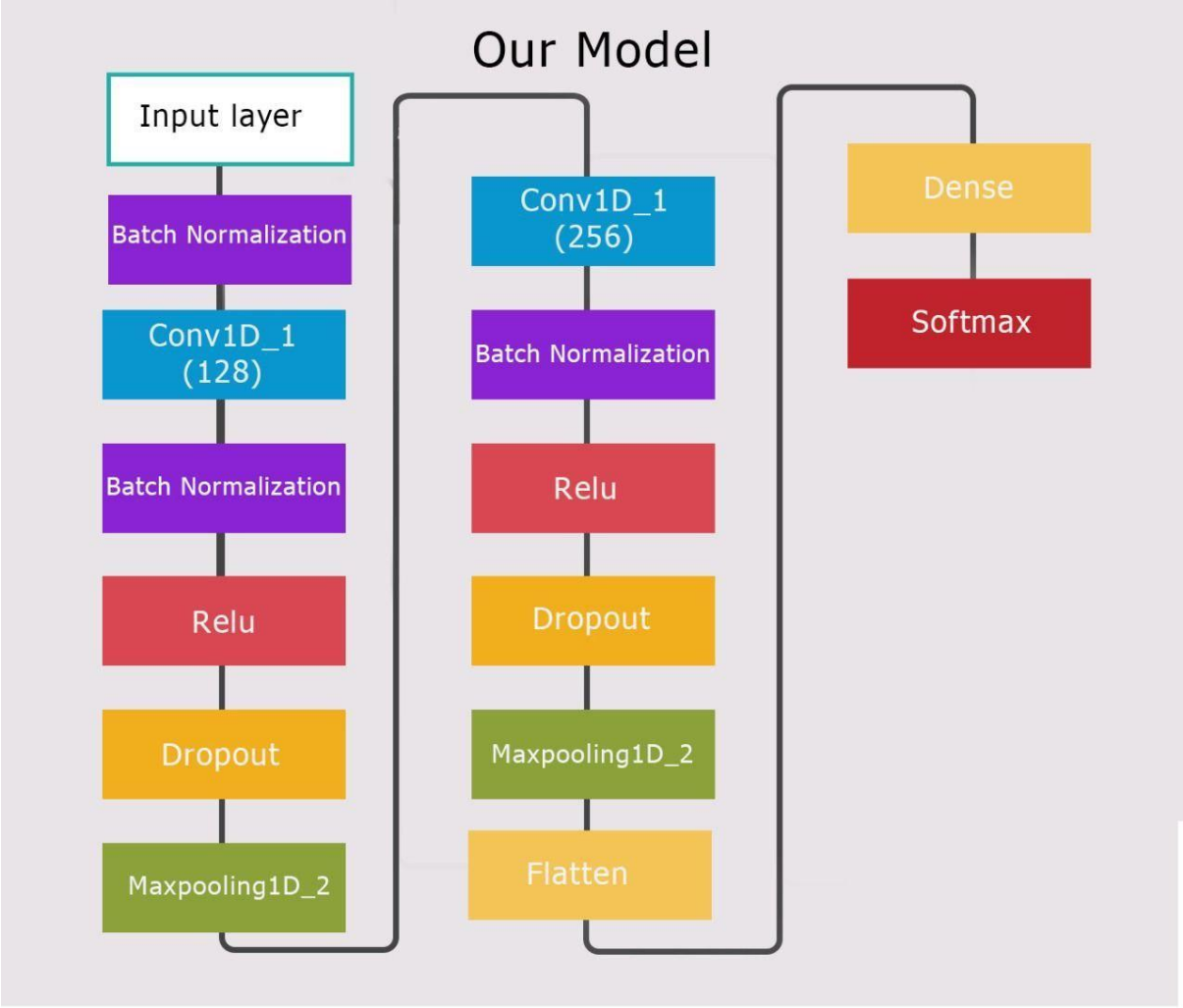


Figure 3-68 : Our Final model ARCHITECTURE

Chapter 4 **Visual Emotion Recognition System**

4.1 Introduction

In this chapter we will explain our visual emotion recognition system. And how to deal with video data sets and how to extract features to feed into the network. We Will clearly explain our network which used to classify our emotion from videos in Detail. And we will show our results on both RAVDESS and SAVEE data sets. We Will clearly explain all these in sections in detail. In section 4.2 show how to deal with Videos and extract features. In section 4.3 explain our model's structure LRCN Model [32] and ConvLstm model [33] in detail. In section 4.4 explain ConvLstm model, How to Improve it and results on RAVDESS and SAVEE data sets. In section 4.5 Explain LRCN model, how to improve it and results on RAVDESS and SAVEE data Sets. In Section 4.6 explain our conclusion.

4.2 Features Extraction

In this section the preprocessing done on data will be discussed, Section 4.2.1 discuss a general introduction about data preprocessing and why it is important Then what steps we used in our project. section 4.2.2 is a section about landmarks And the RAVDESS facial landmarks dataset that we used to process the frames. In Section 4.2.3 we discuss in details what preprocessing we done on RAVDESS Dataset, Then section 4.2.4 is about what preprocessing done on SAVEE.

4.2.1 Data Preprocessing

Data Preprocessing includes the steps we need to follow to transform or Encode data so that it may be easily parsed by the machine; it is a common first step In the deep learning workflow to prepare raw data in a format that the network can Accept. For example, you can resize image input to match the size of an image input Layer. You can also preprocess data to enhance desired features or reduce artifacts That can bias the network. For example, you can normalize or remove noise from Input data[34].Common data preprocessing methods are: resizing, face detection, Cropping, adding noises, and data normalization consists of local normalization, Global contrast normalization and histogram equalization[35]. Figure 4.1 shows the Data preprocessing is an early stage in any machine learning project.

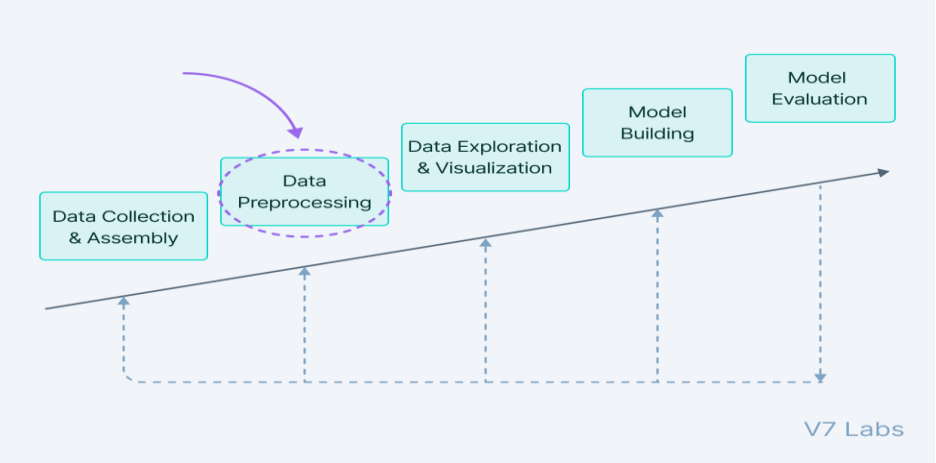


Figure 4-1: Model training steps.

Preprocessing methods that are used on our dataset are:

1. Divide videos into labels representing the emotions.
2. Divide each video into frames and label them.
3. Crop each frame using landmarks.
4. Resize each frame into 64*64.
5. Normalized each frame.
6. Convert labels for each video to one hot encoded.

4.2.2 Landmarks

Facial landmark detection is the task of detecting key landmarks on the face and Tracking them. Facial landmark detection algorithms help to automatically identify The Locations of the facial key landmark points on a facial image or from a video. The key Landmark points normally include the facial regions like nose tip, eye Corner, Eyebrows and chin tip. Some applications of facial landmark detection are Face Swap, head pose detection, detecting facial gestures, gaze direction [36].

RAVDESS Landmark Dataset

The Landmarks used to crop frames in the RAVDESS database. Landmark is A dataset containing tracked facial landmark movements. This dataset is found in [37][38] .Motion tracking of actors' faces was produced by OpenFace 2.1.0. Tracked Information includes: facial landmark detection, head pose estimation, facial action Unit recognition, and eye-gaze estimation.This data set contains tracking for all 2452 RAVDESS trials. All tracking movement data are contained in 2452 CSV files. Each Actor has 104 tracked trials (60 speech, 44 song). Note, there are no song files for Actor 18.Total Tracked Files = (24 Actors x 60 Speech trials) + (23 Actors x 44 Song Trials) = 2452 files. Tracking results for each trial are provided as individual comma Separated value files (CSV format). File naming convention of tracked files is Identical to that of the RAVDESS. For example, the tracked file "01-01-01-01-01-01-01.csv" corresponds to RAVDESS audio-video file "01-01-01-01-01-01-01.mp4". Fig 4.2 and 4.3 are a sample of that dataset from 01-01-01-01-01-01-01.csv file, each column in this file represents a specific Landmark which is shown in table 4.1.

```
frame, timestamp, confidence, success, gaze_0_x, gaze_0_y, gaze_0_z, gaze_1_x, gaze_1_y, gaze_1_z, pose_Tx, pose_Ty, p
1, 0, 0.94848, 1, 0.10917, 0.147619, -0.983001, -0.166114, 0.136956, -0.97655, 21.184, -4.04434, 422.08, 0.0528253, -0
2, 0.04, 0.953299, 1, 0.107503, 0.150627, -0.982728, -0.165798, 0.140331, -0.976124, 21.3134, -4.05283, 421.827, 0.061
3, 0.08, 0.955743, 1, 0.10642, 0.153098, -0.982464, -0.165396, 0.142754, -0.975841, 20.9377, -3.801, 420.152, 0.077066
4, 0.12, 0.963871, 1, 0.106936, 0.154229, -0.982231, -0.164536, 0.145976, -0.97551, 20.4166, -3.87284, 418.836, 0.0831
...
...
204, 8.12, 0.952574, 1, 0.121502, 0.17169, -0.97763, -0.143931, 0.155402, -0.97731, 12.8349, -5.59191, 410.328, 0.1077
205, 8.16, 0.943502, 1, 0.119735, 0.171356, -0.977906, -0.141987, 0.154965, -0.977663, 13.3425, -6.40826, 410.029, 0.0
```

Figure 4-2: Sample of 01-01-01-01-01-01-01.csv file.

```
e_Ty, pose_Tz, pose_Rx, pose_Ry, pose_Rz, x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_10, x_11, x_12, x_13, x_14
253, -0.0474321, 0.0169401, 129.32, 129.568, 130.737, 132.621, 136.572, 143.131, 151.472, 161.39, 172.219, 182.459, 191.
, 0.0619525, -0.0490535, 0.0168232, 129.336, 129.567, 130.75, 132.65, 136.563, 143.075, 151.445, 161.462, 172.38, 182.67
.077066, -0.0420931, 0.0169737, 129.401, 129.594, 130.767, 132.676, 136.57, 143.019, 151.292, 161.226, 172.174, 182.559,
 0.0831857, -0.0378122, 0.0119711, 129.144, 129.391, 130.663, 132.677, 136.624, 143.065, 151.28, 161.147, 172.084, 182.4


 0.107749, -0.058971, 0.0502874, 124.423, 124.206, 125.189, 127.019, 130.81, 136.986, 144.871, 154.67, 165.76, 176.364,
29, 0.0806764, -0.0697097, 0.0482961, 124.486, 124.252, 125.217, 127.038, 130.833, 137.022, 144.961, 154.903, 166.055, 1
```

Figure 4-3: Sample of 01-01-01-01-01-01-01.csv file.

Table 4-1: Content of the csv file for facial landmarks.

Columns	Represents
Columns 1-3	Timing and Detection Confidence
Columns 4-291	Eye Gaze Detection
Columns 292-297	Head pose
Columns 298-433	Facial Landmarks locations in 2D
Columns 434-637	Facial Landmarks locations in 3D
Columns 638-677	Rigid and non-rigid shape parameters
Columns 687-712	Facial Action Units[5]

4.2.3 Preprocessing On RAVDESS

Data preprocessing steps performed on RAVDESS are discussed in this Section. First is labeling and how we classified the dataset by labels in 4.2.3.1. Second in 4.2.3.2 frames are extracted from videos, then in section 4.2.3.3 is cropping these Frames with landmarks, lastly performing one hot encoding in section 4.2.3.4.

4.2.3.1 Labeling

Instead of having 24 folders (one for each actor), each contains 60 speech Videos and 44 song videos (except for actor 18), we found it would be easier and Better to classify these videos into only 6 folders (neutral, happy, angry, calm, fearful And sad), each one represents an emotion (which are the 6 emotions that are Common between speech and song) and contains all videos that belongs to that Emotion from all actors.

Now each folder contains (8 no. of emotion videos x 24 actors =192 speech video) + (8 no. of emotion videos x 23 actors= 184 song video) = 376 videos.

Neutral is an exception as each actor has only 8 neutral video (4 for speech & 4 for Song), therefore, the total number of videos in neutral folder is (4 no. of emotion Videos x 24 actors =96 speech videos) + (4 no. of emotion videos x 23 actors= 92 Song video) = 188 videos.

4.2.3.2 Frames

Each video is then divided into frames, 30 frames are taken from each video. 30 frames from each video will then be fed to the model as one sequence. A frame Shown in fig 4.4 for Actor1 with calm emotion.

Figure 4-4: Sample of Actor1 frame.

4.2.3.3 Cropping With Landmarks

The facial Landmarks dataset mentioned above was then used to crop each One of these frames so that only the face of the actor is focused at and the rest of The Background is neglected. This is shown in fig 4.5 that shows a frame for actor1 After Being cropped. each frame is then resized to a height and width equal 64.

Figure 4-5: Cropped Actor1 frame.

4.2.3.4 One Hot Encoding

It is a common way of preprocessing categorical features for machine learning Models. This type of encoding creates a new binary feature for each possible Category and assigns a value of 1 to the feature of each sample that corresponds to Its original category. In our project we used Keras's to_categorical method to convert Labels into one-hot-encoded vectors.[39]

4.2.4 Preprocessing On SAVEE

Data preprocessing steps performed on SAVEE are discussed in this section. First is labeling and how we classified the dataset by labels in 4.2.4.1. Second in 4.2.4.2 Frames are extracted from videos, lastly performing one hot encoding in section 4.2.4.3.

4.2.4.1 Labeling

Instead of having 4 folders (one for each actor), each contains 90 speech Videos , we found it would be easier and better to classify these videos into only 5 Folders (neutral, happy, angry, fearful and sad), each one represents an Emotion(which are the 5 emotions that are found common with RAVDESS) and Contains all videos that belongs to that emotion from all actors.

Now each folder contains (15 no. of emotion videos x 4 actors =60 speech videos).

Neutral is an exception as each actor has 30 neutral videos, therefore , the total Number of videos in the neutral folder is (30 no. of emotion videos x 4 actors =120 Speech videos).

4.2.4.2 Frames

Each video is then divided into frames, 30 frames are taken from each video, Each frame is resized to a height and width equal 64.30 frames from each video will Then be fed to the model as one sequence. Same as done on RAVDESS.A frame Shown in fig 4.6 for KL with sad emotion.

Figure 4-6: Sample for Actor KL

4.2.4.3 One Hot Encoding

The same hot encoding was done here on the SAVEE data set as Keras's To_categorical Method is used to convert labels into one-hot-encoded vectors.

4.2.5 Dataset Splitting

Data splitting is commonly used in machine learning to split data into a train, Test, or validation set. This approach allows us to find the model hyper-parameter and also estimate the generalization performance. Our dataset was at first split to 80% for training and 20% for testing, and the validation is split as 20% of the Training data. After this we tried another split which resulted in better performance which is 90% for training and 10% for test, and the validation is 10 % of the training.

4.3 LRCN & ConvLstm Structure Model

In this section we will talk about our LRCN and ConvLstm models structure And run original LRCN and ConvLstm on RAVDESS and SAVEE datasets. Section 4.3.1 shows the LRCN model structure. Section 4.3.1.1 shows LRCN architecture Implementation. Section 4.3.1.2 shows LRCN model results on the RAVDESS dataset And SAVEE dataset. Section 4.3.2 shows the ConvLstm model structure. Section 4.3.2.1 shows LRCN architecture implementation. Section 4.3.2.2 shows LRCN Model results on the RAVDESS dataset and SAVEE dataset.

4.3.1 LRCN Model

LRCN (Long-Recurrent Convolutional Network) approach combines Convolution and LSTM layers in a single model. Another approach can be to use CNN model and LSTM model trained separately. But in this approach we combine CNN and LSTM in one model. The Convolutional layers are used for spatial feature Extraction from the Video frames, and the extracted spatial features are fed to LSTM Layer(s) at each time-step for temporal sequence modeling. This way the network learns spatiotemporal features directly in an end-to-end training, resulting in a Robust Model. Figure 4.7 shows the LRCN network which takes as an input video Frames and outputs the emotion or classification.

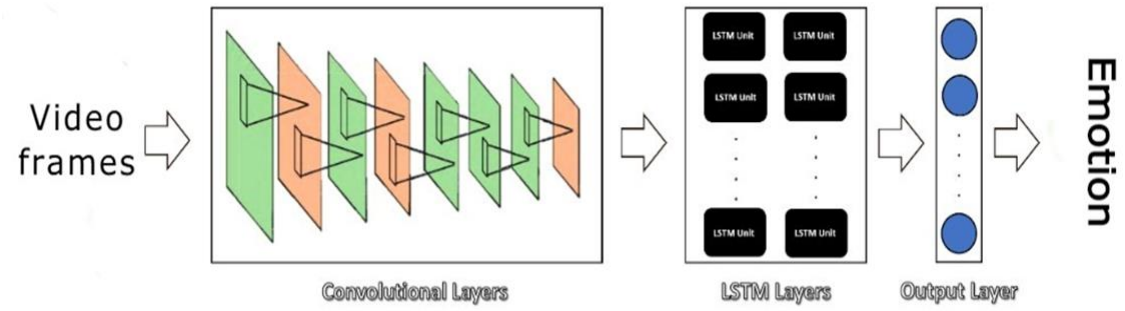


Figure 4-7: LRCN network.

This LRCN architecture is introduced or proposed by the Long-term Recurrent Convolutional Networks for Visual Recognition and this LRCN model used in “” Long-term Recurrent Convolutional Networks for Visual Recognition and Description “”paper[32]. We used a Time Distributed wrapped layer so that every frame of video Can be applied to the same layer independently. So it makes a layer capable of Taking input of shape (No of frames, width, height, No of channels). if the input Shape was (width, height, No of channels) it allows to input the whole video into the Model in a single shot. In our work we used input shape (No of frames, width, height, No of channels). In our work with LRCN model we used 30 frame and 64*64 (width * Height) as mentioned in the preprocessing section and No of channels equals 3.

4.3.1.1 LRCN Architecture Implementation

To implement our LRCN architecture, we will use time-distributed **Conv2D** Layers which will be followed by **MaxPooling2D** and **Dropout** layers. The feature Extracted from the **Conv2D** layers will be then flattened using the **Flatten** layer and will be fed to a **LSTM** layer. The **Dense** layer with Softmax activation will then use the output from the **LSTM** layer to predict the emotion. Our model diagram is shown in figure 4.8.

Figure 4-8: Original LRCN model structure.

4.3.1.2 LRCN Model Results

In this section we will show the results when applying the LRCN model on the RAVDESS dataset and SAVEE dataset. Figure 4.9 shows the model report and Number of parameters and trainable parameters in the LRCN model.

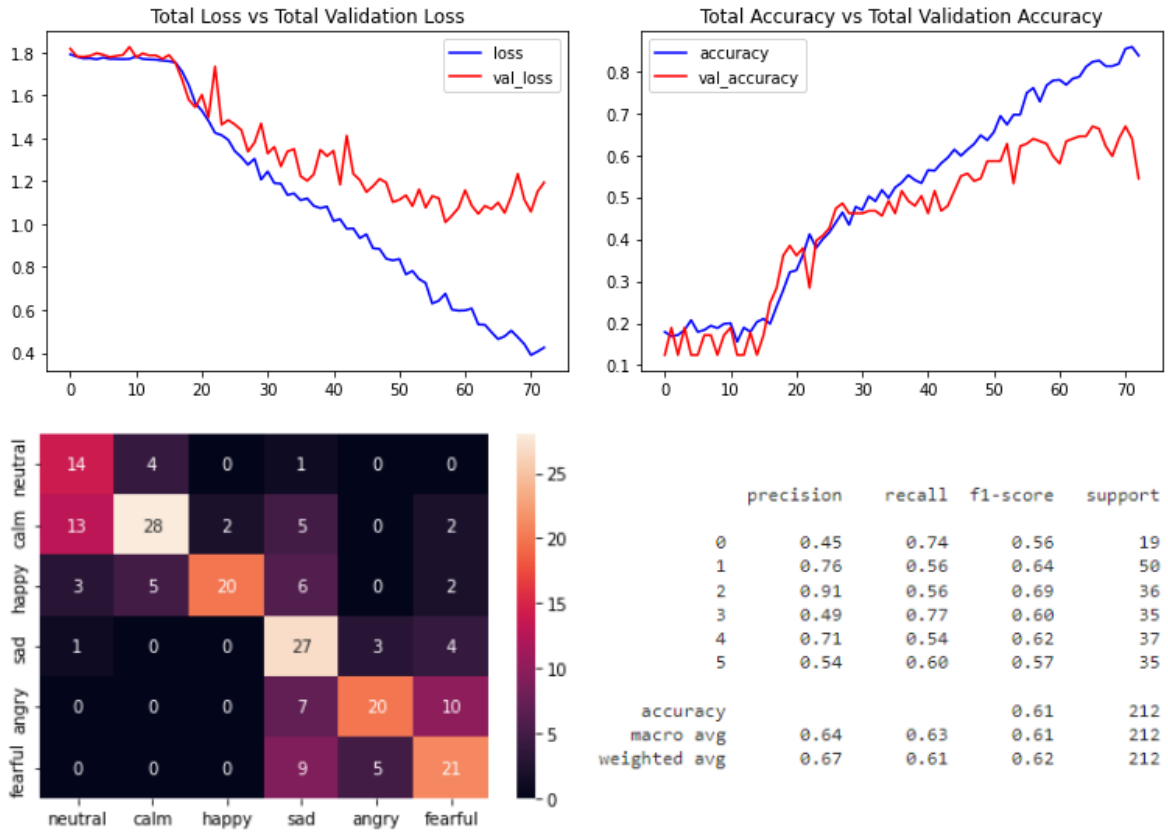
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed (TimeDistr  (None, 30, 64, 64, 16)   448
 ibuted)

 time_distributed_1 (TimeDis  (None, 30, 16, 16, 16)   0
 tributed)

 time_distributed_2 (TimeDis  (None, 30, 16, 16, 16)   0
 tributed)

 time_distributed_3 (TimeDis  (None, 30, 16, 16, 32)   4640
 tributed)

 time_distributed_4 (TimeDis  (None, 30, 4, 4, 32)     0
 tributed)

 time_distributed_5 (TimeDis  (None, 30, 4, 4, 32)     0
 tributed)

 time_distributed_6 (TimeDis  (None, 30, 4, 4, 64)     18496
 tributed)

 time_distributed_7 (TimeDis  (None, 30, 2, 2, 64)     0
 tributed)

 time_distributed_8 (TimeDis  (None, 30, 2, 2, 64)     0
 tributed)

 time_distributed_9 (TimeDis  (None, 30, 2, 2, 64)     36928
 tributed)

 time_distributed_10 (TimeDi  (None, 30, 1, 1, 64)     0
 stributed)

 time_distributed_11 (TimeDi  (None, 30, 64)           0
 stributed)

 lstm (LSTM)                 (None, 32)                12416

 dense (Dense)               (None, 6)                 198

=================================================================
Total params: 73,126
Trainable params: 73,126
Non-trainable params: 0
```

Figure 4-9: Original LRCN model structure report.

RAVDESS Dataset

Speech

We first split the dataset into a train and test set with test size 20%.Then Validation split from the train set 20%. We trained the model with optimizer Adam and categorical_crossentropy as a loss function which is used for multi-class Classification models. We trained the LRCN model with these hyper parameters. Learning rate used is the default for the optimizer Adam in Kera's which is 0.001, Number of epochs equals 90 with early stopping function to prevent overfitting. The Model stopped training at epoch number 73. The LRCN model achieved 61.32%. We Show loss and accuracy graph, heatmap of confusion matrix and classification report In figure 4.10.

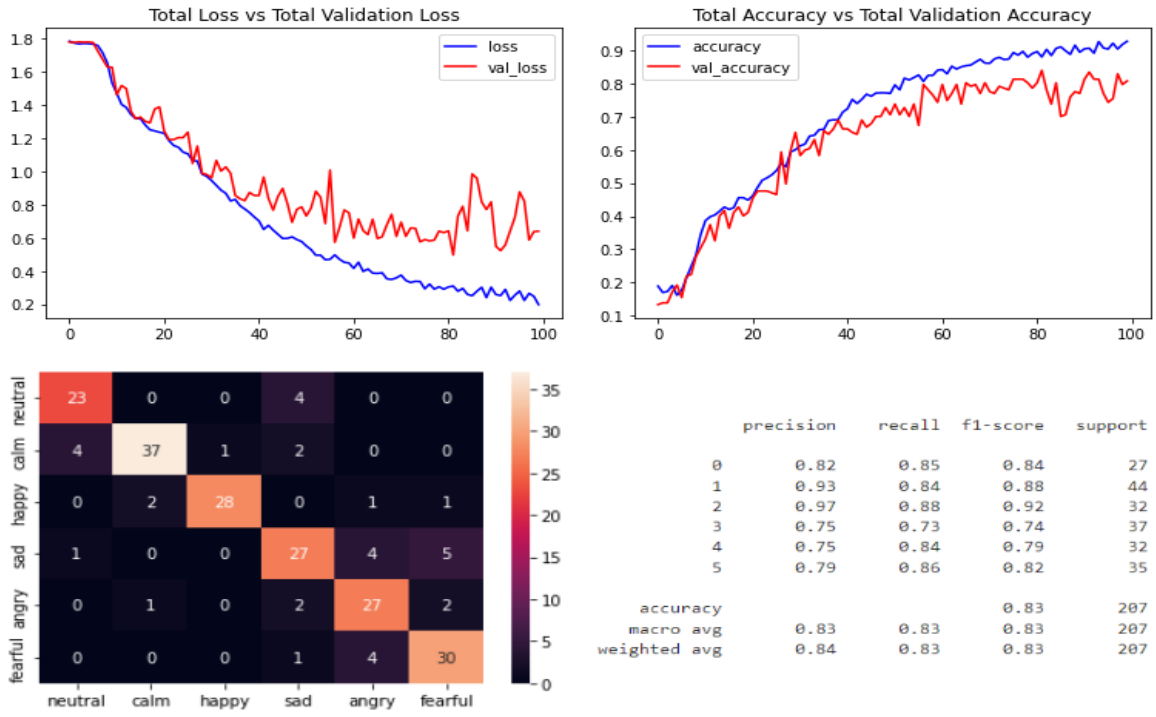


	precision	recall	f1-score	support
0	0.45	0.74	0.56	19
1	0.76	0.56	0.64	50
2	0.91	0.56	0.69	36
3	0.49	0.77	0.60	35
4	0.71	0.54	0.62	37
5	0.54	0.60	0.57	35
accuracy			0.61	212
macro avg	0.64	0.63	0.61	212
weighted avg	0.67	0.61	0.62	212

Figure 4-10: Results of original LRCN model on RAVDESS speech only.

Speech and Song

We first split the dataset into a train and test set with test size 10%.Then Validation split from the train set 10%. We trained the model with optimizer Adam And categorical_crossentropy as a loss function which is used for multi-class Classification models. We trained the LRCN model with these hyperparameters. Learning rate used is the default for the optimizer Adam in Keras which is 0.001, Number of epochs equals 100 with early stopping function to prevent overfitting. But The model is trained with all epochs instead of an early stopping function. The LRCN Model achieved 83.09% as the best accuracy. And achieved 80.67% ± 2.18%. We Show loss and accuracy graph, heatmap of confusion matrix and classification report in figure 4.11.



	precision	recall	f1-score	support
0	0.82	0.85	0.84	27
1	0.93	0.84	0.88	44
2	0.97	0.88	0.92	32
3	0.75	0.73	0.74	37
4	0.75	0.84	0.79	32
5	0.79	0.86	0.82	35
accuracy			0.83	207
macro avg	0.83	0.83	0.83	207
weighted avg	0.84	0.83	0.83	207

Figure 4-11: Results of original LRCN model on RAVDESS.

SAVEE Dataset

We first split the dataset into a train and test set with test size 20%.Then Validation split from the train set 20%. We trained the model with optimizer Adam And categorical_crossentropy as a loss function which is used for multi-class Classification models. We trained the LRCN model with these hyperparameters. Learning rate used is the default for the optimizer Adam in Keras which is 0.001, Number of epochs equals 100 with early stopping function to prevent overfitting. But The model is trained with all

epochs instead of an early stopping function. The LRCN Model achieved 90.28% as the best accuracy.The LRCN model achieved 87% ± 2.78%.we present loss and accuracy graph, heatmap of confusion matrix and Classification report in figure 4.12.

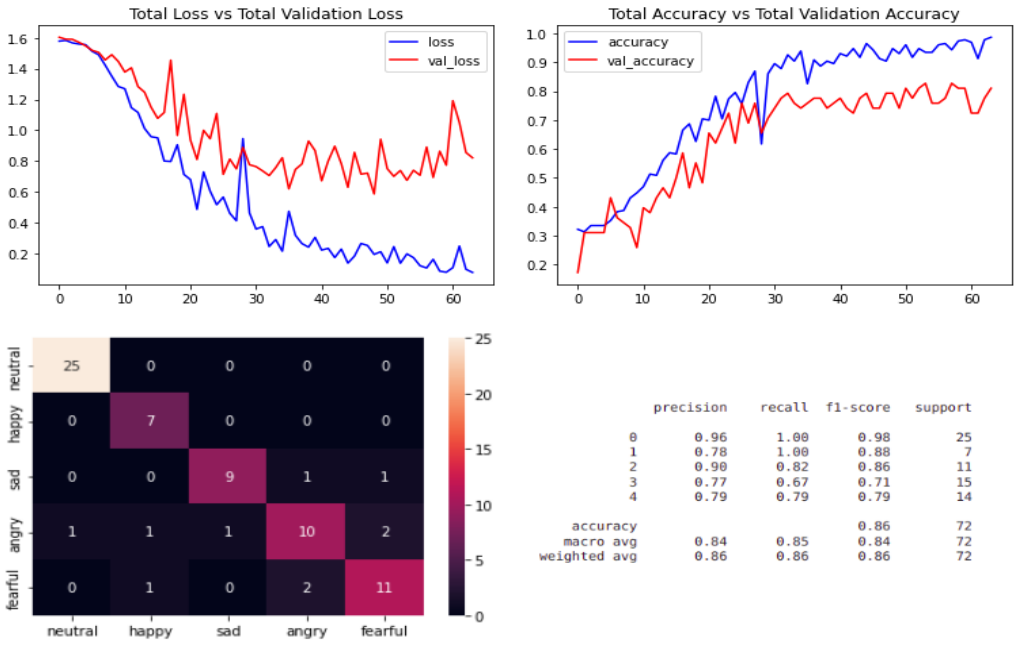


Figure 4-12: Results of original LRCN on SAVEE

4.3.2 ConvLstm Model Approach

We will implement the first approach by using a combination of ConvLstm Cells. A ConvLstm cell is a variant of an Lstm network that contains convolutions Operations in the network. It is an Lstm with convolution embedded in the Architecture, which makes it capable of identifying spatial features of the data while Keeping into account the temporal relation. And this ConvLstm model approach is Used in "Convolutional Lstm Network: A Machine Learning Approach for Precipitation Nowcasting "paper [33]. Show ConvLstm cell in figure 4.13.

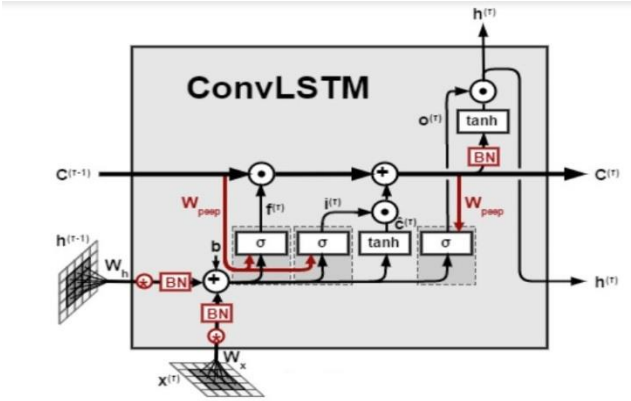


Figure 4-13: ConvLstm cell.

4.3.2.1 ConvLstm Model Implementation

To construct the model, we will use Keras **ConvLstm2D** recurrent layers. The **ConvLstm2D** layer also takes in the number of filters and kernel size required For Applying the convolutional operations. The output of the layers is flattened in the end and is fed to the **Dense** layer with softmax activation which outputs the Probability of each action category. We will also use **MaxPooling3D** layers to reduce the dimensions of the frames and avoid unnecessary computations and **Dropout** layers to prevent overfitting the Model on the data. The architecture is a simple one and has a small number of Trainable parameters. This is because we are only dealing with a small subset of the Dataset which does not require a large-scale model. Our model diagram is shown in Figure 4.14.

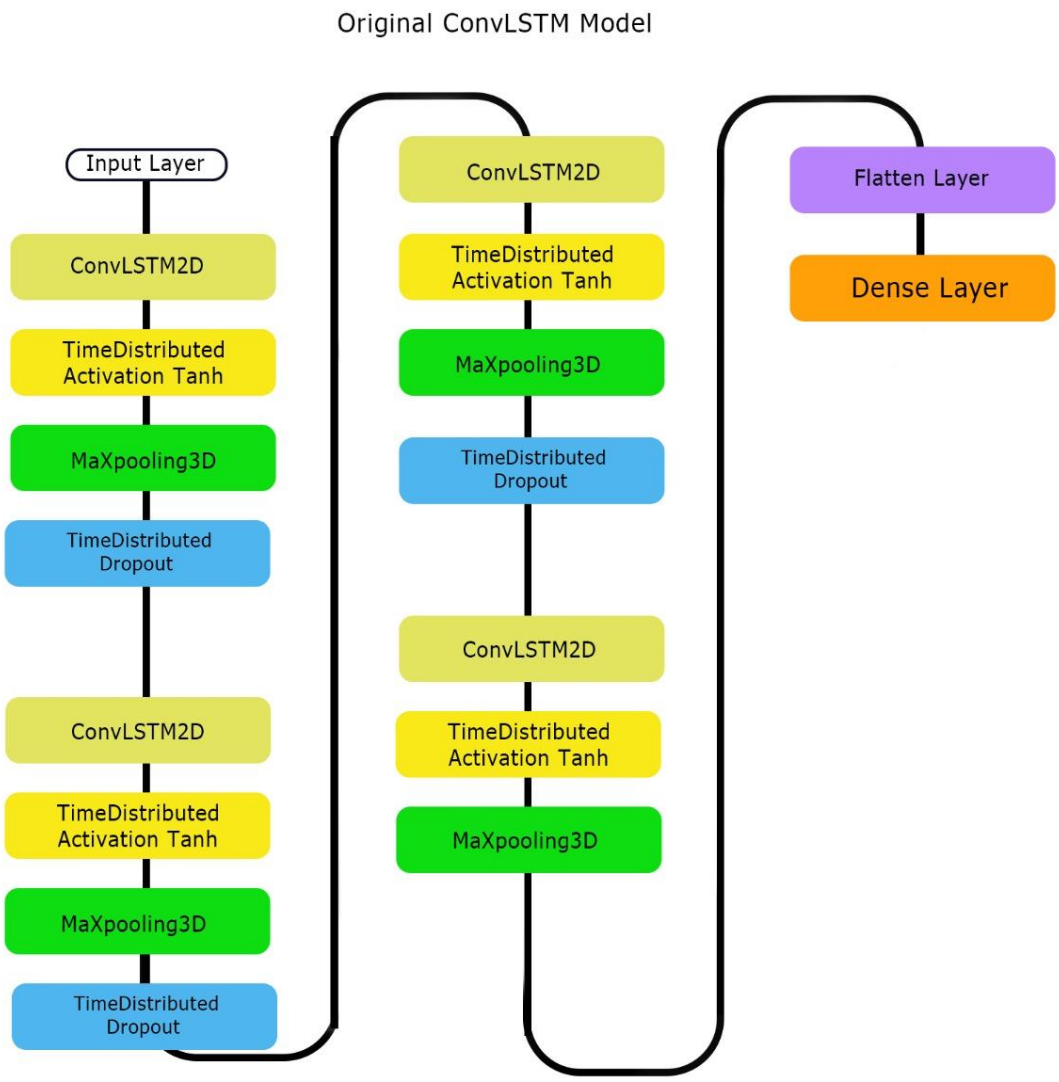


Figure 4-14: Original ConvLstm structure

4.3.2.2 ConvLstm Model Results

In this section we will show the results when applying the ConvLstm model On The RAVDESS dataset. First we will show the results of the model on RAVDESS Dataset speech only then we will show the results using speech & song dataset. Figure 4.15 shows the ConvLstm model report and number of parameters and Trainable parameters in the ConvLstm model.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv_lstm2d (ConvLSTM2D)    (None, 30, 62, 62, 4)     1024

 max_pooling3d (MaxPooling3D  (None, 30, 31, 31, 4)    0
 )

 time_distributed (TimeDistr  (None, 30, 31, 31, 4)    0
 ibuted)

 conv_lstm2d_1 (ConvLSTM2D)  (None, 30, 29, 29, 8)     3488

 max_pooling3d_1 (MaxPooling  (None, 30, 15, 15, 8)    0
 3D)

 time_distributed_1 (TimeDis  (None, 30, 15, 15, 8)    0
 tributed)

 conv_lstm2d_2 (ConvLSTM2D)  (None, 30, 13, 13, 14)    11144

 max_pooling3d_2 (MaxPooling  (None, 30, 7, 7, 14)     0
 3D)

 time_distributed_2 (TimeDis  (None, 30, 7, 7, 14)     0
 tributed)

 conv_lstm2d_3 (ConvLSTM2D)  (None, 30, 5, 5, 16)      17344

 max_pooling3d_3 (MaxPooling  (None, 30, 3, 3, 16)     0
 3D)

 flatten (Flatten)           (None, 4320)              0

 dense (Dense)               (None, 6)                 25926

=================================================================
Total params: 58,926
Trainable params: 58,926
Non-trainable params: 0
```

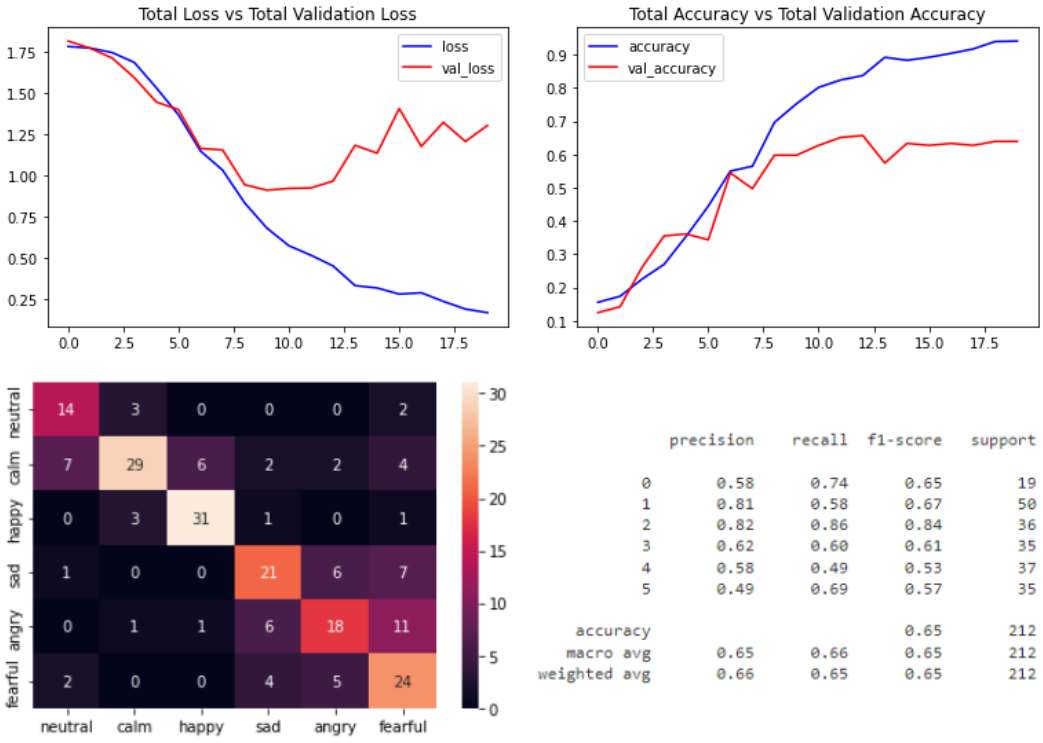
Figure 4-15: Original ConvLstm model report.

RAVDESS Dataset

We split the dataset to 60% train 20% validation data 20% test data, set hyperparameter learning rate is default(0.001) ,optimizer is Adam.

Speech

It achieved accuracy 64.62%. Loss and accuracy graphs, heatmap of Confusion matrix and classification report show in figure 4.16.

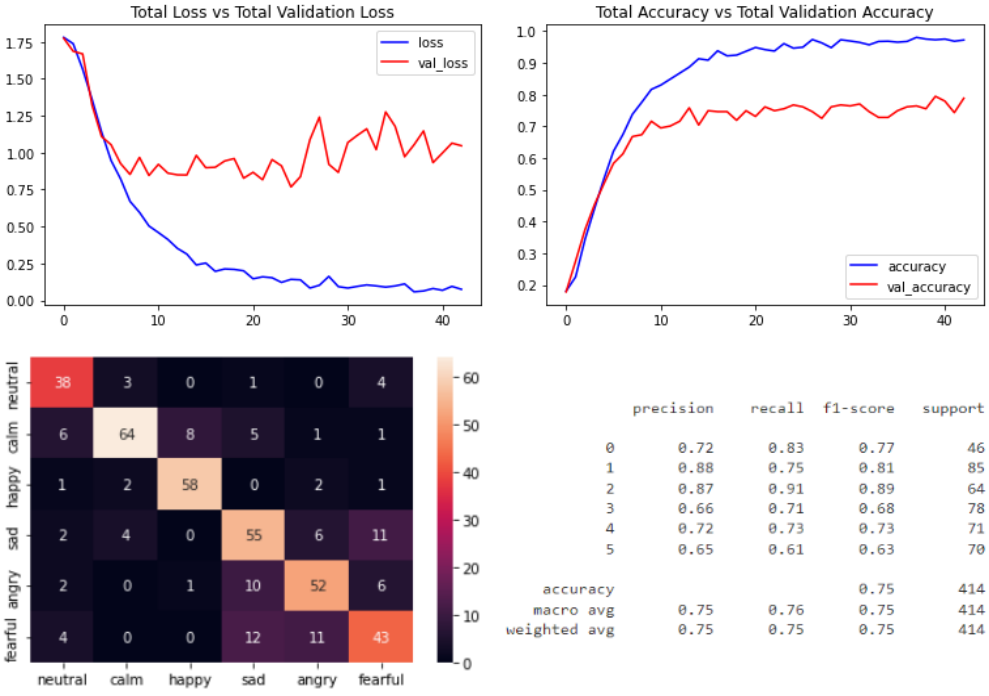


	precision	recall	f1-score	support
0	0.58	0.74	0.65	19
1	0.81	0.58	0.67	50
2	0.82	0.86	0.84	36
3	0.62	0.60	0.61	35
4	0.58	0.49	0.53	37
5	0.49	0.69	0.57	35
accuracy			0.65	212
macro avg	0.65	0.66	0.65	212
weighted avg	0.66	0.65	0.65	212

Figure 4-16: Results of Original ConvLstm model on RAVDESS speech.

Speech and Song

It achieved accuracy 74.8%. Loss and accuracy graphs, heatmap of confusion Matrix and classification report show in figure 4.17.

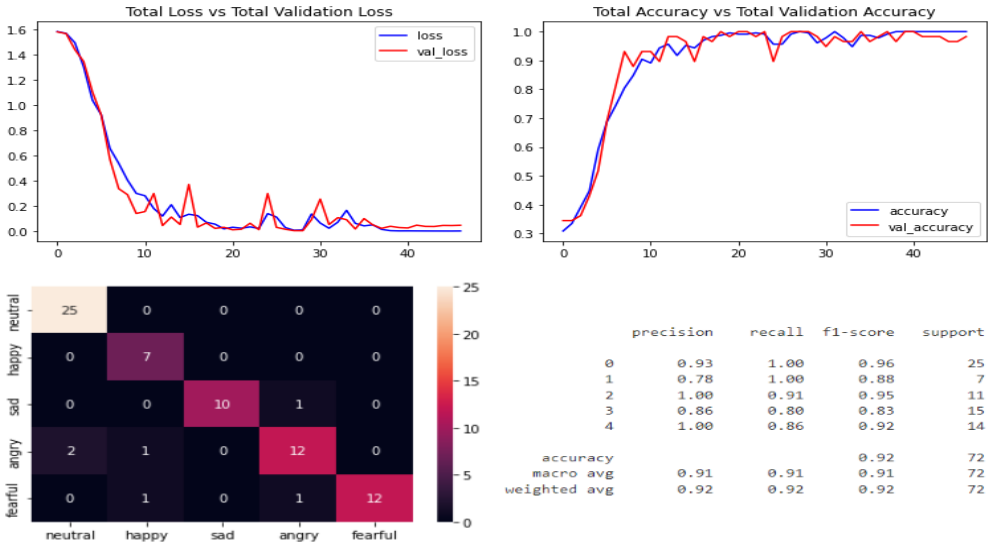


	precision	recall	f1-score	support
0	0.72	0.83	0.77	46
1	0.88	0.75	0.81	85
2	0.87	0.91	0.89	64
3	0.66	0.71	0.68	78
4	0.72	0.73	0.73	71
5	0.65	0.61	0.63	70
accuracy			0.75	414
macro avg	0.75	0.76	0.75	414
weighted avg	0.75	0.75	0.75	414

Figure 4-17: Result of Original ConvLstm model on RAVDESS.

SAVEE Dataset

We first split the dataset into a train and test set with test size 20%.Then Validation split from the train set 20%. The learning rate is default(0.001), optimizer Is Adam, number of epochs equals 70 with early stopping function to prevent Overfitting. The model is stopped at epoch number 29. The ConvLstm model Achieved 91.67%. We present loss and accuracy graph, heatmap of confusion Matrix and classification report in figure 4.18.



	precision	recall	f1-score	support
0	0.93	1.00	0.96	25
1	0.78	1.00	0.88	7
2	1.00	0.91	0.95	11
3	0.86	0.80	0.83	15
4	1.00	0.86	0.92	14
accuracy			0.92	72
macro avg	0.91	0.91	0.91	72
weighted avg	0.92	0.92	0.92	72

Figure 4-18: Results of original ConvLstm model on SAVEE.

4.4 ConvLstm Model

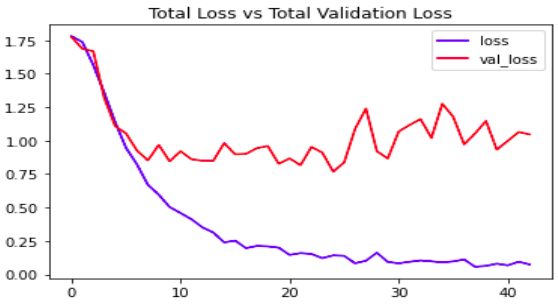
In this section we will talk about our experiments and improvements on the ConvLstm model using RAVDESS data set and show our results of our best ConvLstm model on both RAVDESS and SAVEE data sets. Section 4.4.1 shows our Experiments on the ConvLstm model. Section 4.4.2 shows our best model and Results on both RAVDESS and SAVEE datasets.

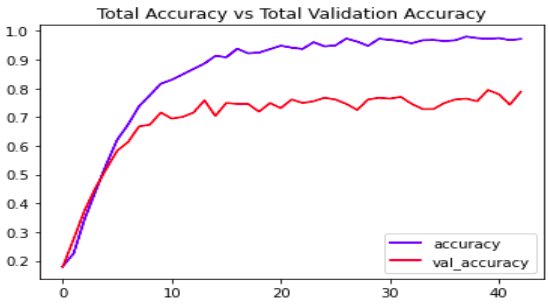
4.4.1 Improvement Of ConvLstm Model

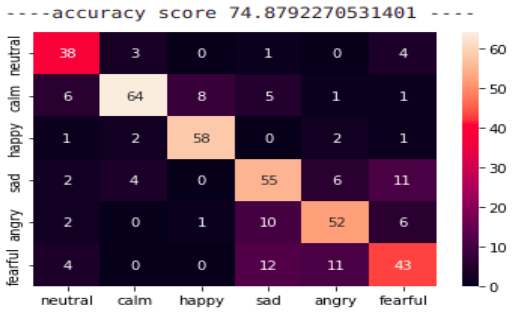
In this section we try some experiments on the original Convlstm model and Adding some modifications such as adding batch normalization layer after and Before activation function and changing dropout layers. We will show the accuracy Results of this section in table 4.2. Graphs of loss, accuracy, heatmap of confusion Matrix and classification report of this table will be in figures 4.19.

Table 4-2: Accuracy of our modification on ConvLSTM model and RAVDESS data set.

Expert_No	Modification	Accuracy
4.1	Original ConvLstm with adam optimizer and default lr	74.80%
4.2	Original ConvLstm with Batch normalization before Activation function	71.98%
4.3	Original ConvLstm with Landmark with Batch normalization	84.54%
4.4	Original ConvLstm with landmark with Batch normalization with change dropout	83.09%

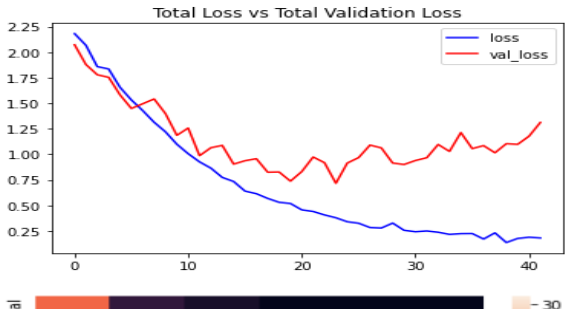


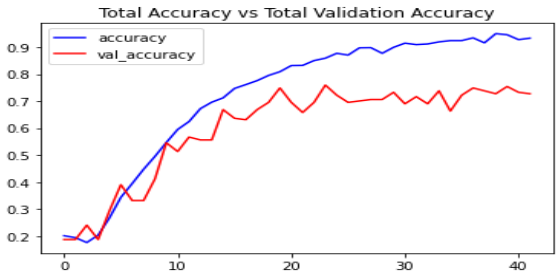




	precision	recall	f1-score	support
0	0.72	0.83	0.77	46
1	0.88	0.75	0.81	85
2	0.87	0.91	0.89	64
3	0.66	0.71	0.68	78
4	0.72	0.73	0.73	71
5	0.65	0.61	0.63	70
accuracy			0.75	414
macro avg	0.75	0.76	0.75	414
weighted avg	0.75	0.75	0.75	414

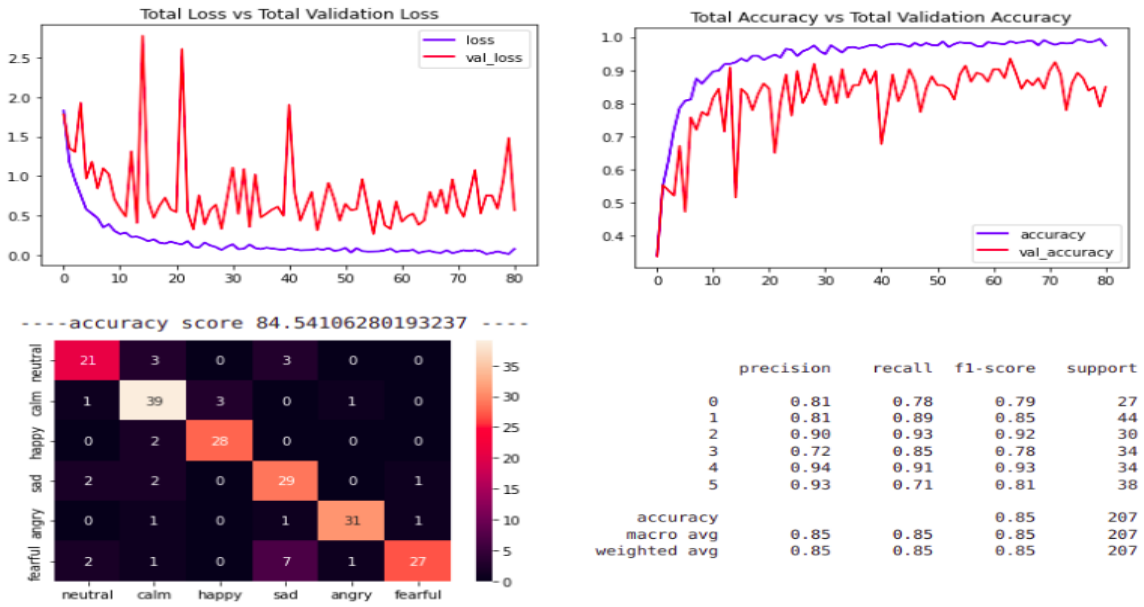
(a)



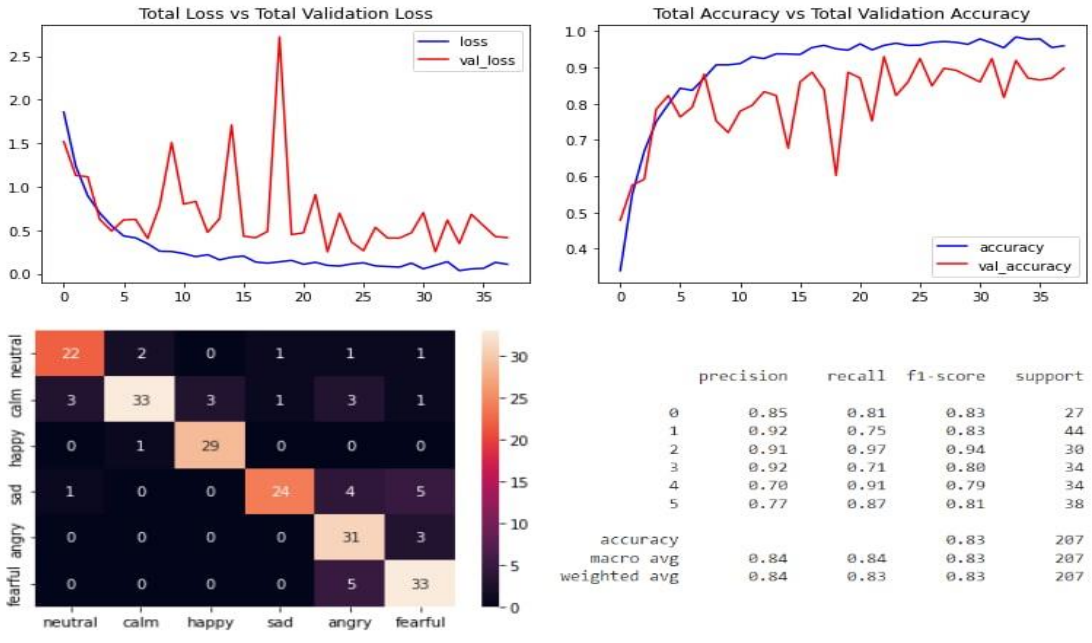


	precision	recall	f1-score	support
0	0.62	0.78	0.69	27
1	0.84	0.70	0.77	44
2	0.71	0.94	0.81	32
3	0.95	0.49	0.64	37
4	0.73	0.75	0.74	32
5	0.60	0.71	0.65	35
accuracy			0.72	207
macro avg	0.74	0.73	0.72	207
weighted avg	0.75	0.72	0.72	207

(b)



(c)



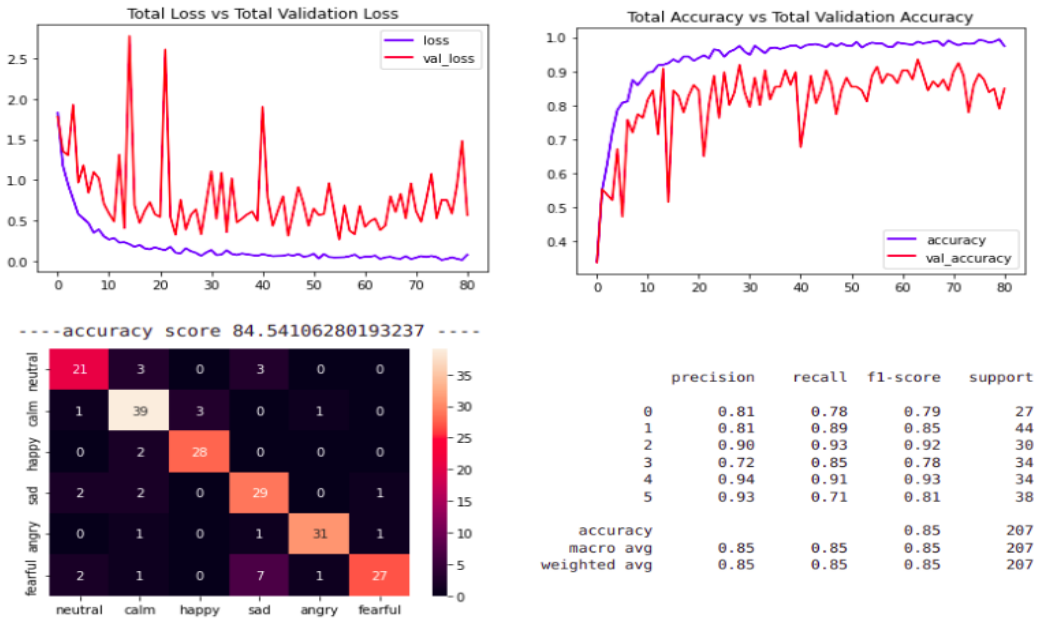
(d)

Figure 4-19: Results of experiments in table. 4.2 : (a) results of original ConvLstm with adam optimizer and default lr, (b) results of original ConvLstm with Batch normalization before Activation function, (c) results of original ConvLstm with Landmark with Batch normalization, (d) results of original ConvLstm with landmark with Batch normalization with change dropout.

4.4.2 Results of final ConvLstm Model

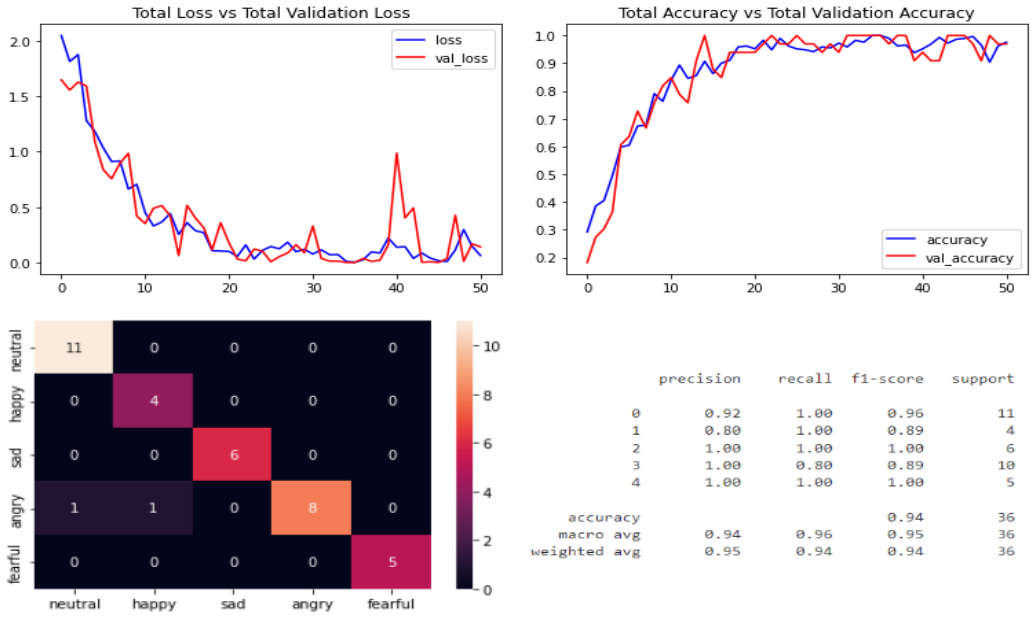
In previous sections we presented all our experiments on ConvLstm model Results. And now we will choose our best model based on best accuracy. We will find that expert_no 4.3 in table 4.2 has the best accuracy 84.54%. Our best model is ConvLstm model [33] with adding batch normalization, dropout rate is 0.2. This Model Achieved best accuracy on a RAVDESS data set 84.54% and on a SAVEE Data set 94.44%. We will show our best ConvLstm model in figure 4.20. We will Show graphs of loss, accuracy, heatmap of confusion matrix and classification report of RAVDESS And SAVEE results on our best ConvLstm model in figures 4.21.

Figure 4-20: Final ConvLstm model.



	precision	recall	f1-score	support
0	0.81	0.78	0.79	27
1	0.81	0.89	0.85	44
2	0.90	0.93	0.92	30
3	0.72	0.85	0.78	34
4	0.94	0.91	0.93	34
5	0.93	0.71	0.81	38
accuracy			0.85	207
macro avg	0.85	0.85	0.85	207
weighted avg	0.85	0.85	0.85	207

(a)



	precision	recall	f1-score	support
0	0.92	1.00	0.96	11
1	0.80	1.00	0.89	4
2	1.00	1.00	1.00	6
3	1.00	0.80	0.89	10
4	1.00	1.00	1.00	5
accuracy			0.94	36
macro avg	0.94	0.96	0.95	36
weighted avg	0.95	0.94	0.94	36

(b)

Figure 4-21: Results of final ConvLstm on RAVDESS and SAVEE : (a) Results on RAVDESS, (b) Results of SAVEE.

4.5 LRCN Model

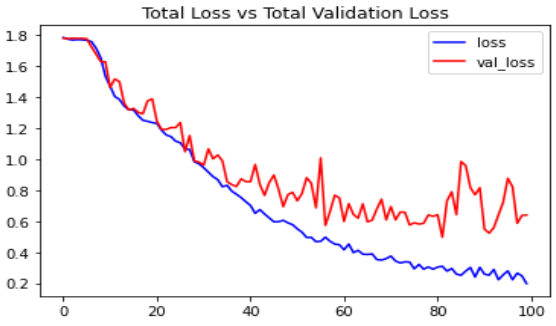
In this section we try to improve the LRCN model accuracy through some Modification on it and using different hyperparameter values using a RAVDESS data Set. We will present our work in the following sections. Section 4.5.1 shows some Modification without using RAVDESS landmarks[35]. Section 4.5.2 shows some Modification using RAVDESS landmarks. Section 4.5.3 comparison between results Of two previous sections 4.5.1 and 4.5.2. Section 4.5.4 shows the effect of changing Hyperparameter values. Section 4.5.5 shows different modifications but doesn't Improve our model accuracy significantly. Section 4.5.6 shows results of our best LRCN model on both RAVDESS and SAVEE data sets.

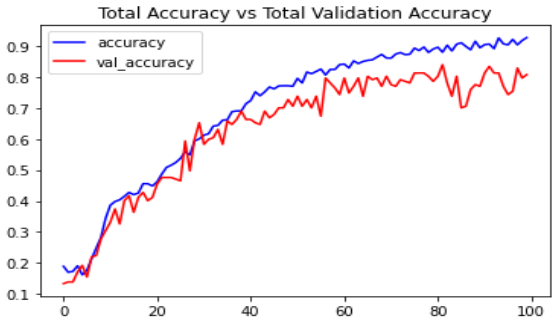
4.5.1 Some Experiments Without Using RAVDESS Landmarks

In this section we try some experiments on the LRCN model [32] without using RAVDESS landmarks. Adding some modification such as adding batch normalization Layer and changing hyperparameter values such as optimizer, learning rate, dropout Rate and # units of LSTM. We will show the accuracy results of this section in table 4.3, loss and accuracy graphs, heatmap of confusion matrix and classification report in figures 4.22.

Table 4-3: Accuracy of experiments on LRCN without using RAVDESS landmarks.

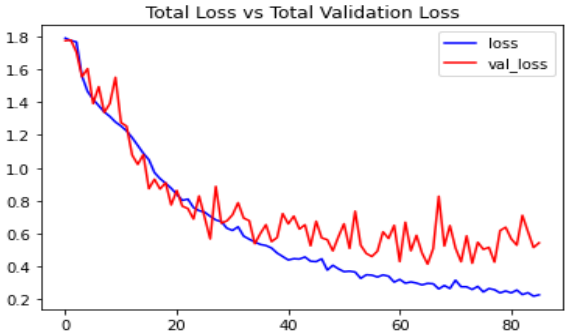
Expert_No	Modification	Worst	Avg	Best
4.5	LRCN_model with adam opt & default lr	78.74%	80.67%	83.09%
4.6	LRCN_model with rms opt & default lr	79.7%	80.34%	81.64%
4.7	LRCN_model with adam opt & 0.0001 lr	56.52%	63.93%	68.12%
4.8	LRCN_model with batch normalization	84.5%	84.5%	84.54%
4.9	LRCN_model with change in dropout rate	78.26%	79.55%	81.64%
4.10	LRCN_model with batch normalization & change in dropout rate	84.5%	86.56%	88.89%
4.11	LRCN_model with batch normalization & change in dropout rate & 64 lstm units	70.53%	80.36%	86.47%

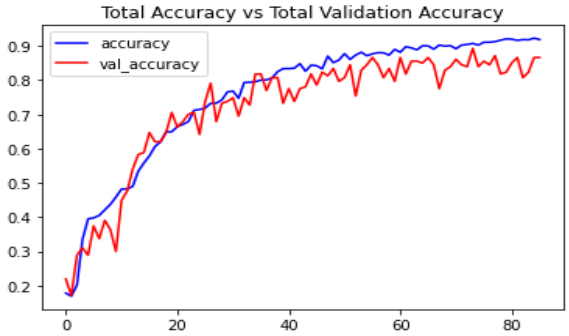




	precision	recall	f1-score	support
0	0.82	0.85	0.84	27
1	0.93	0.84	0.88	44
2	0.97	0.88	0.92	32
3	0.75	0.73	0.74	37
4	0.75	0.84	0.79	32
5	0.79	0.86	0.82	35
accuracy			0.83	207
macro avg	0.83	0.83	0.83	207
weighted avg	0.84	0.83	0.83	207

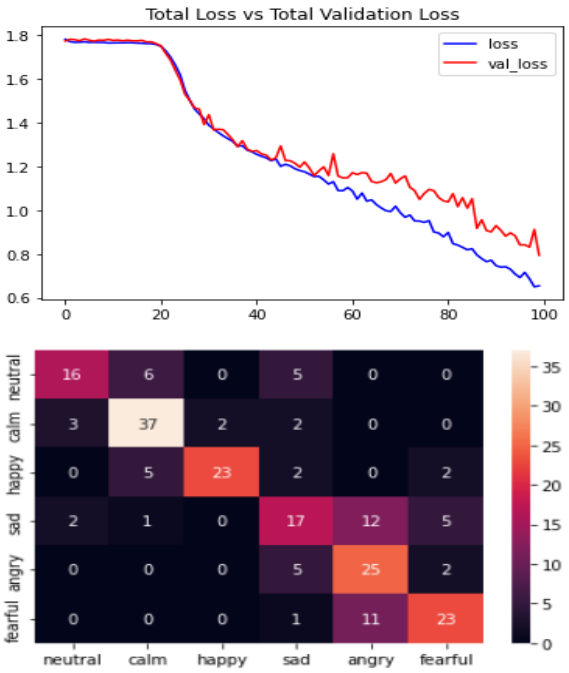
(a)

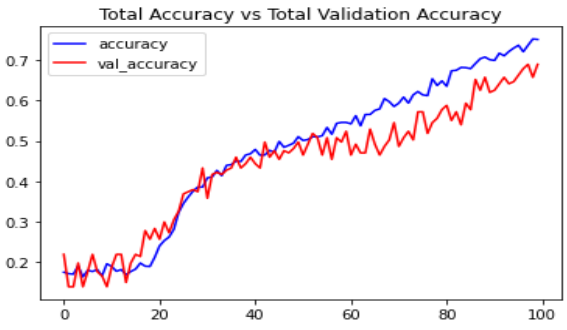




	precision	recall	f1-score	support
0	0.79	0.85	0.82	27
1	0.84	0.82	0.83	44
2	0.93	0.88	0.90	32
3	0.74	0.68	0.70	37
4	0.91	0.91	0.91	32
5	0.72	0.80	0.76	35
accuracy			0.82	207
macro avg	0.82	0.82	0.82	207
weighted avg	0.82	0.82	0.82	207

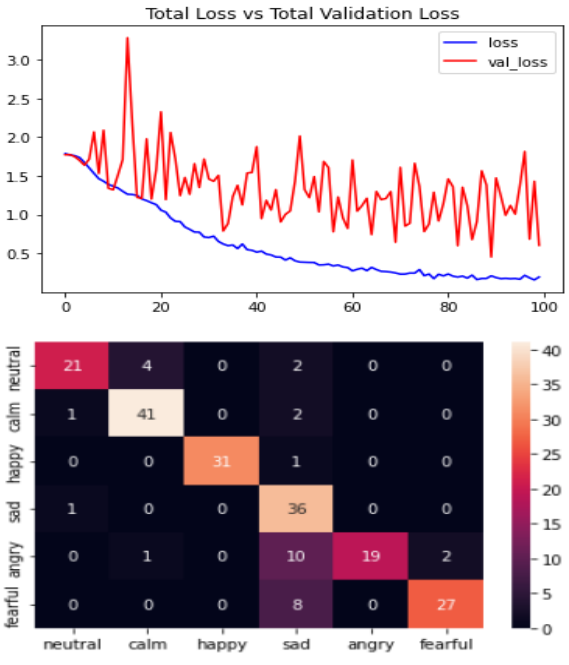
(b)

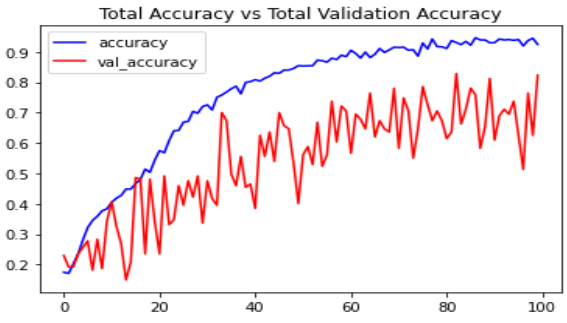




	precision	recall	f1-score	support
0	0.76	0.59	0.67	27
1	0.76	0.84	0.80	44
2	0.92	0.72	0.81	32
3	0.53	0.46	0.49	37
4	0.52	0.78	0.62	32
5	0.72	0.66	0.69	35
accuracy			0.68	207
macro avg	0.70	0.68	0.68	207
weighted avg	0.70	0.68	0.68	207

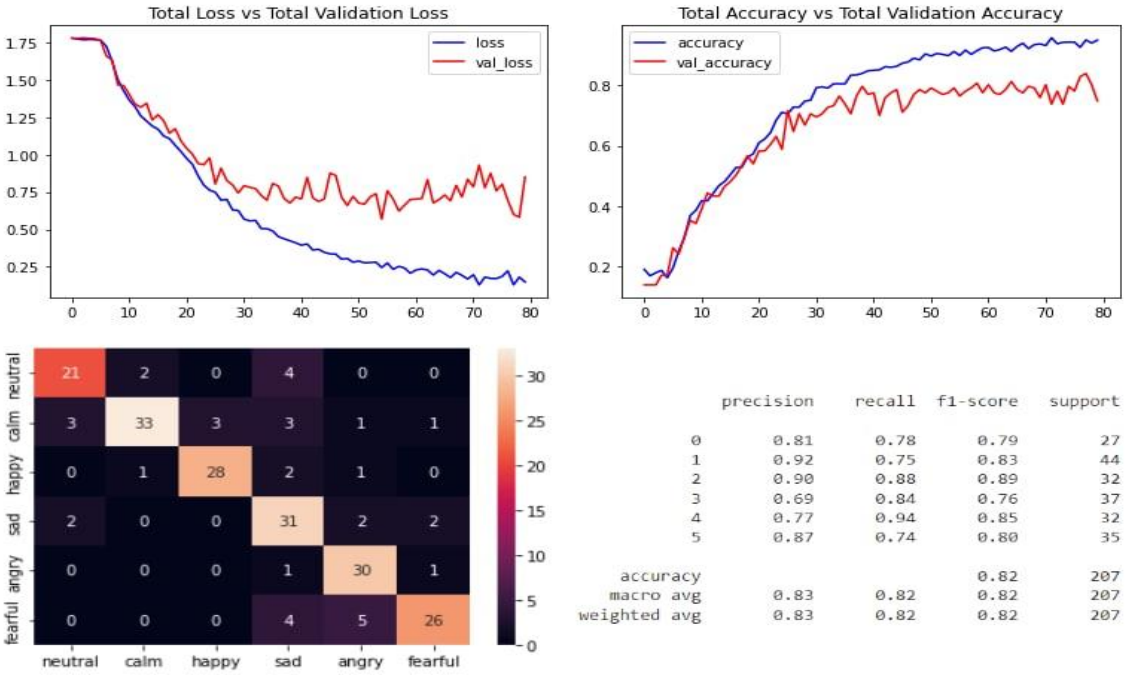
(c)





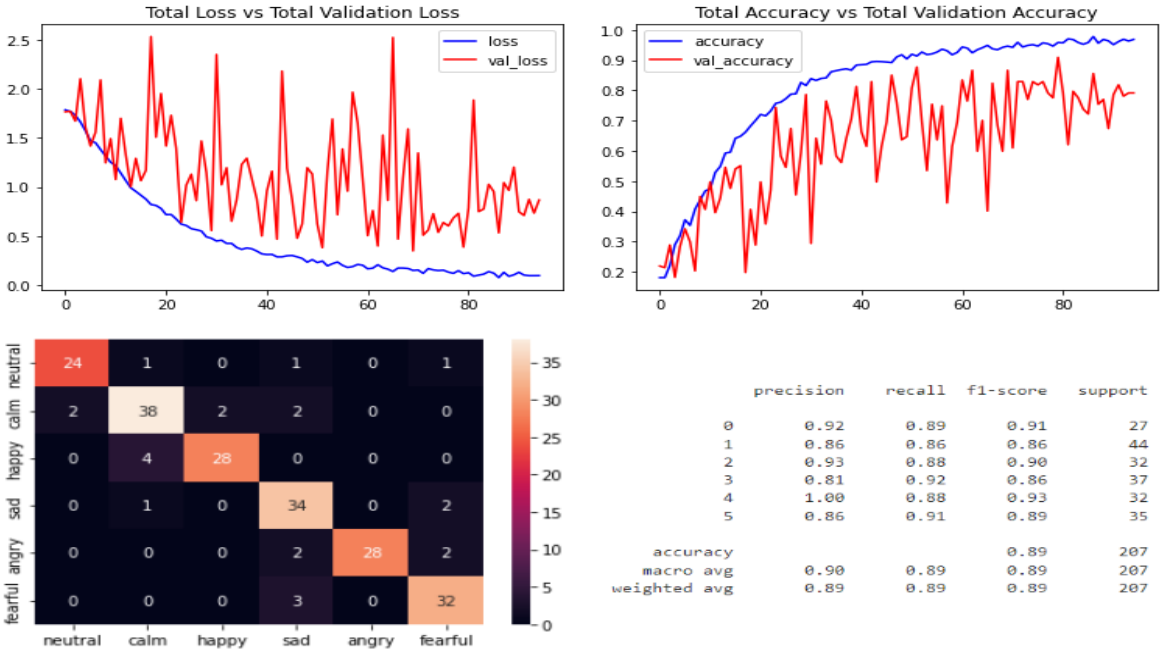
	precision	recall	f1-score	support
0	0.91	0.78	0.84	27
1	0.89	0.93	0.91	44
2	1.00	0.97	0.98	32
3	0.61	0.97	0.75	37
4	1.00	0.59	0.75	32
5	0.93	0.77	0.84	35
accuracy			0.85	207
macro avg	0.89	0.84	0.85	207
weighted avg	0.88	0.85	0.85	207

(d)



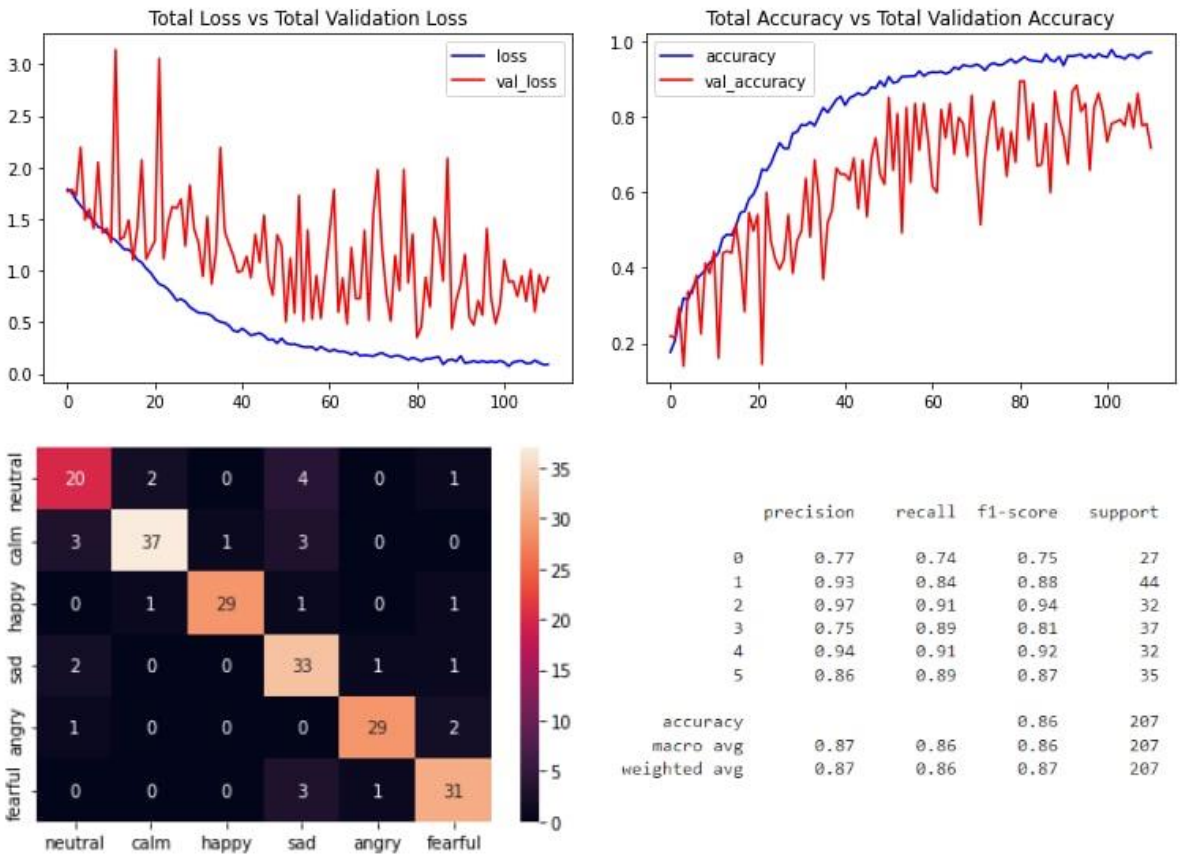
	precision	recall	f1-score	support
0	0.81	0.78	0.79	27
1	0.92	0.75	0.83	44
2	0.90	0.88	0.89	32
3	0.69	0.84	0.76	37
4	0.77	0.94	0.85	32
5	0.87	0.74	0.80	35
accuracy			0.82	207
macro avg	0.83	0.82	0.82	207
weighted avg	0.83	0.82	0.82	207

(e)



	precision	recall	f1-score	support
0	0.92	0.89	0.91	27
1	0.86	0.86	0.86	44
2	0.93	0.88	0.90	32
3	0.81	0.92	0.86	37
4	1.00	0.88	0.93	32
5	0.86	0.91	0.89	35
accuracy			0.89	207
macro avg	0.90	0.89	0.89	207
weighted avg	0.89	0.89	0.89	207

(f)



	precision	recall	f1-score	support
0	0.77	0.74	0.75	27
1	0.93	0.84	0.88	44
2	0.97	0.91	0.94	32
3	0.75	0.89	0.81	37
4	0.94	0.91	0.92	32
5	0.86	0.89	0.87	35
accuracy			0.86	207
macro avg	0.87	0.86	0.86	207
weighted avg	0.87	0.86	0.87	207

(g)

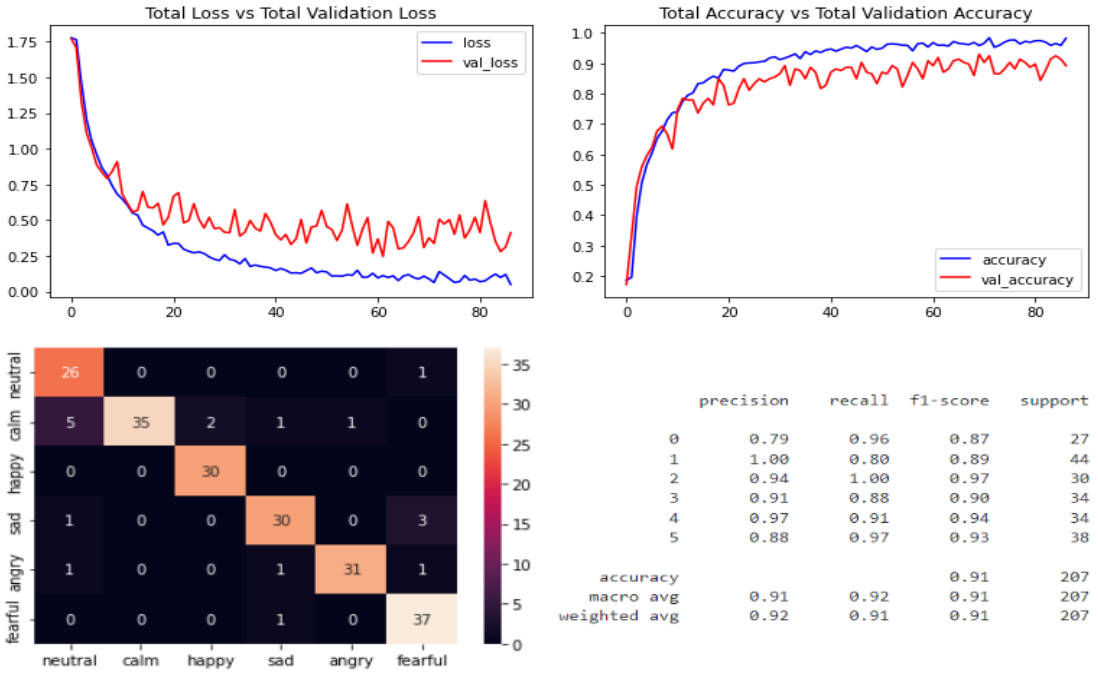
Figure 4-22: experiments of LRCN without RAVDESS landmarks results in table 4.3 : (a) LRCN_model with adam opt & default lr, (b) LRCN_model with rms opt & default lr, (c) LRCN_model with adam opt & 0.0001 lr, (d) LRCN_model with batch normalization, (e) LRCN_model with change in dropout rate, (f) LRCN_model with batch normalization & change in dropout rate, (g) LRCN_model with batch normalization & change in dropout rate & 64 lstm units.

4.5.2 Some Experiments Using RAVDESS Landmarks

In this section we try some experiments on the LRCN model [32] using RAVDESS landmarks[34]. And doing the same experiments such as in previous Section 4.5.1. We will show the accuracy results of this section in table 4.4, loss and Accuracy graphs, heatmap of confusion matrix and classification report in figures 2.23.

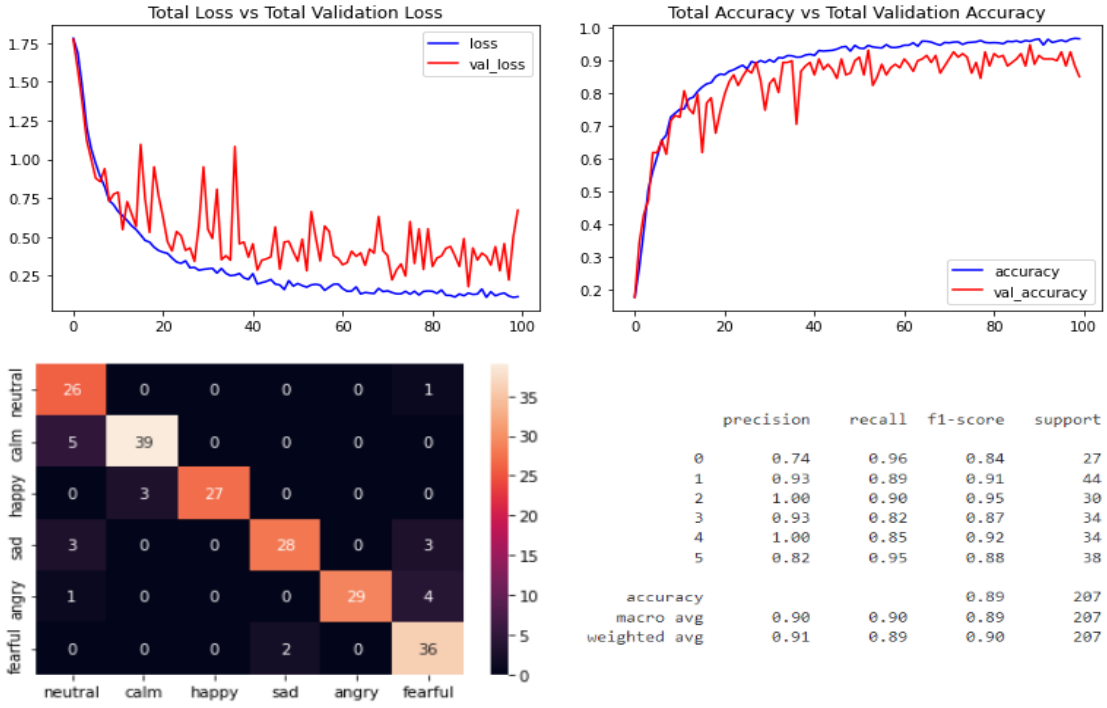
Table 4-4: Accuracy of experiments on LRCN using RAVDESS landmarks.

Expert_No	Modification	Worst	Avg	Best
4.12	LRCN_model with adam opt & default lr	86.47%	89.13%	91.30%
4.13	LRCN_model with rms opt & default lr	88.41%	89.05%	89.37%
4.14	LRCN_model with adam opt & 0.0001 lr	78.74%	78.9%	79.23%
4.15	LRCN_model with batch normalization	84.06%	88.08%	90.83%
4.16	LRCN_model with change in dropout rate	84.54%	85.83%	87.44%
4.17	LRCN_model with batch normalization & change in dropout rate	85.51%	90.34%	92.27%
4.18	LRCN_model with batch normalization & change in dropout rate & 64 lstm units	86.47%	90.63%	94.6%



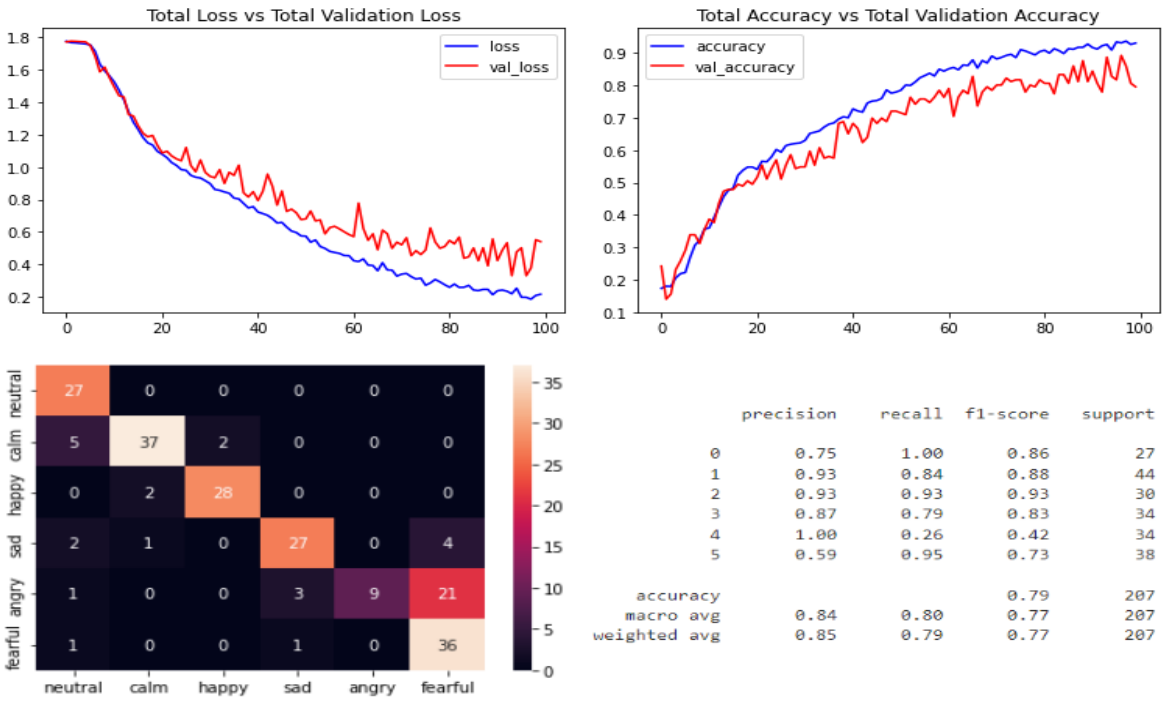
	precision	recall	f1-score	support
0	0.79	0.96	0.87	27
1	1.00	0.80	0.89	44
2	0.94	1.00	0.97	30
3	0.91	0.88	0.90	34
4	0.97	0.91	0.94	34
5	0.88	0.97	0.93	38
accuracy			0.91	207
macro avg	0.91	0.92	0.91	207
weighted avg	0.92	0.91	0.91	207

(a)



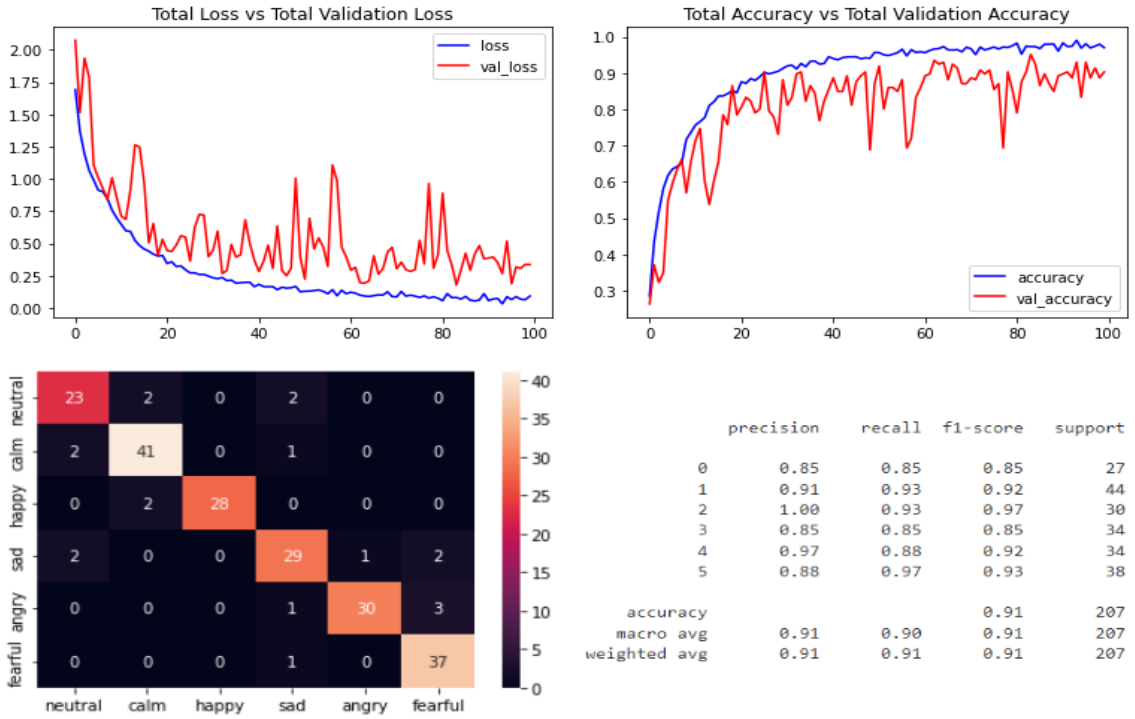
	precision	recall	f1-score	support
0	0.74	0.96	0.84	27
1	0.93	0.89	0.91	44
2	1.00	0.90	0.95	30
3	0.93	0.82	0.87	34
4	1.00	0.85	0.92	34
5	0.82	0.95	0.88	38
accuracy			0.89	207
macro avg	0.90	0.90	0.89	207
weighted avg	0.91	0.89	0.90	207

(b)



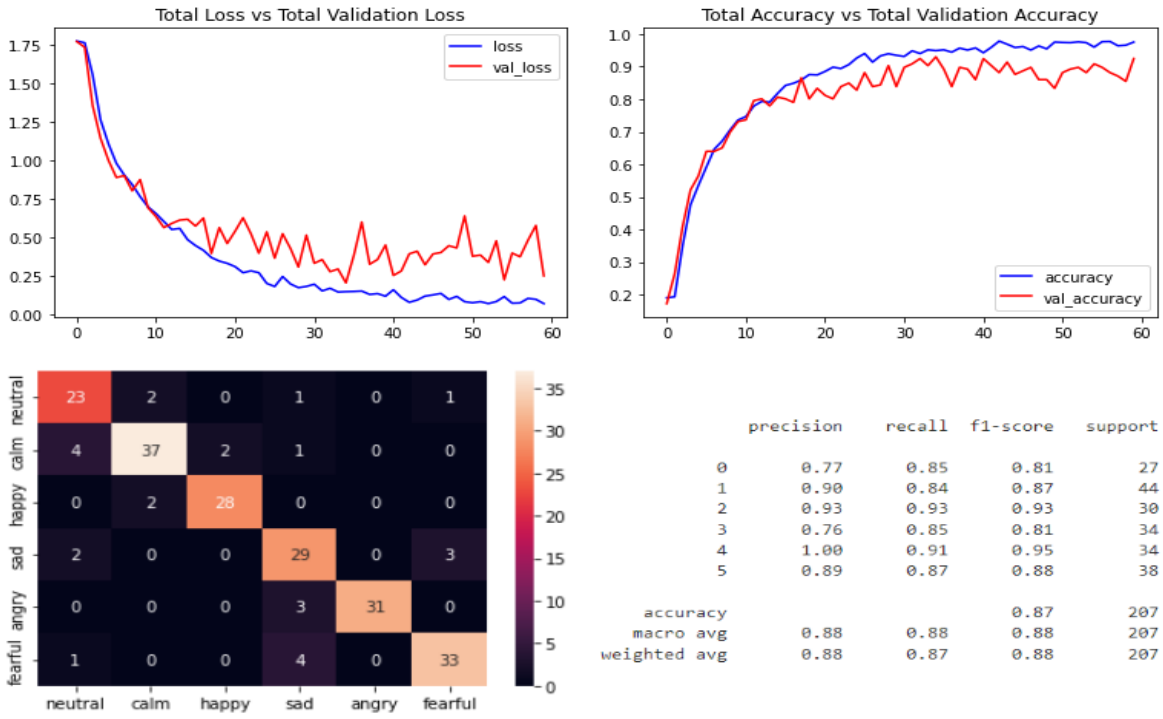
	precision	recall	f1-score	support
0	0.75	1.00	0.86	27
1	0.93	0.84	0.88	44
2	0.93	0.93	0.93	30
3	0.87	0.79	0.83	34
4	1.00	0.26	0.42	34
5	0.59	0.95	0.73	38
accuracy			0.79	207
macro avg	0.84	0.80	0.77	207
weighted avg	0.85	0.79	0.77	207

(c)



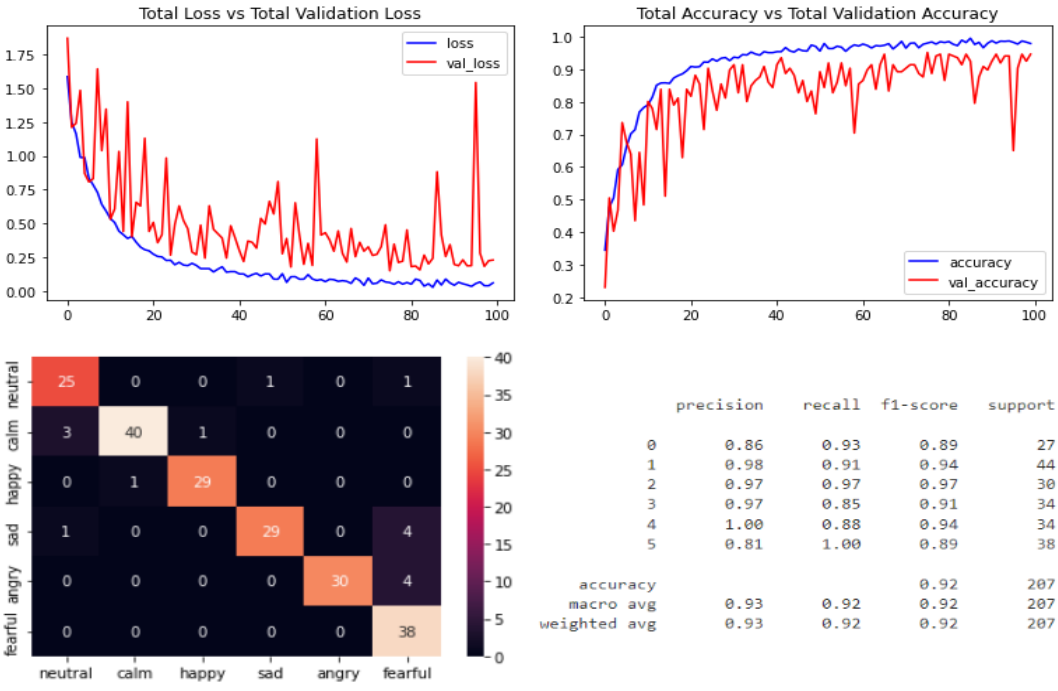
	precision	recall	f1-score	support
0	0.85	0.85	0.85	27
1	0.91	0.93	0.92	44
2	1.00	0.93	0.97	30
3	0.85	0.85	0.85	34
4	0.97	0.88	0.92	34
5	0.88	0.97	0.93	38
accuracy			0.91	207
macro avg	0.91	0.90	0.91	207
weighted avg	0.91	0.91	0.91	207

(d)



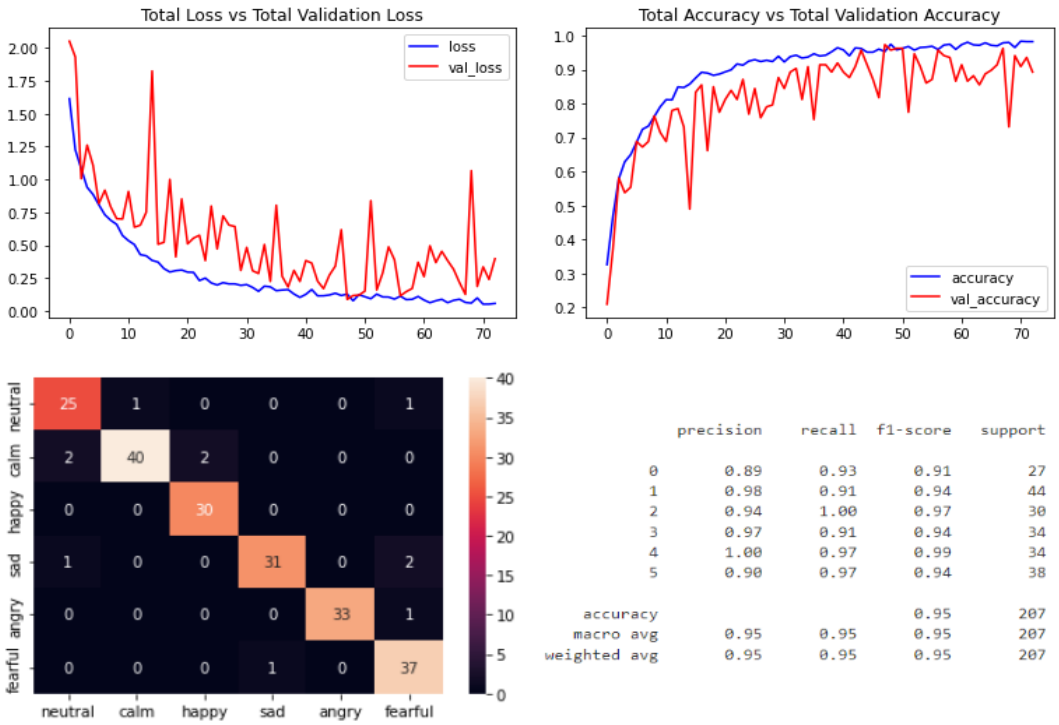
	precision	recall	f1-score	support
0	0.77	0.85	0.81	27
1	0.90	0.84	0.87	44
2	0.93	0.93	0.93	30
3	0.76	0.85	0.81	34
4	1.00	0.91	0.95	34
5	0.89	0.87	0.88	38
accuracy			0.87	207
macro avg	0.88	0.88	0.88	207
weighted avg	0.88	0.87	0.88	207

(e)



	precision	recall	f1-score	support
0	0.86	0.93	0.89	27
1	0.98	0.91	0.94	44
2	0.97	0.97	0.97	30
3	0.97	0.85	0.91	34
4	1.00	0.88	0.94	34
5	0.81	1.00	0.89	38
accuracy			0.92	207
macro avg	0.93	0.92	0.92	207
weighted avg	0.93	0.92	0.92	207

(f)



(g)

Figure 4-23: Experiments of LRCN using RAVDESS landmarks results in table 4.4 : (a) LRCN_model with adam opt & default lr, (b) LRCN_model with rms opt & default lr, (c) LRCN_model with adam opt & 0.0001 lr, (d) LRCN_model with batch normalization, (e) LRCN_model with change in dropout rate, (f) LRCN_model with batch normalization & change in dropout rate, (g) LRCN_model with batch normalization & change in dropout rate & 64 lstm units.

4.5.3 Comparison Between Experiments With And Without RAVDESS Landmarks Based On Best Accuracy.

Based on best accuracy of previous experiments in two sections 4.5.1 and 4.5.2. We will compare between them and show their comparison results in table 4.5

Table 4-5: Comparison between experiments with & without RAVDESS landmarks based on best accuracy.

Experiments	Without landmark	With landmark
LRCN_model with adam opt & default lr	83.09%	91.30%
LRCN_model with rms opt & default lr	81.64%	89.37%
LRCN_model with adam opt & 0.0001 lr	68.12%	79.23%
LRCN_model with batch normalization	84.5%	90.83%
LRCN_model with change in dropout rate	81.64%	87.44%
LRCN_model with batch normalization & change in dropout rate	88.89%	92.27%
LRCN_model with batch normalization & change in dropout rate & 64 lstm units	86.47%	94.6%

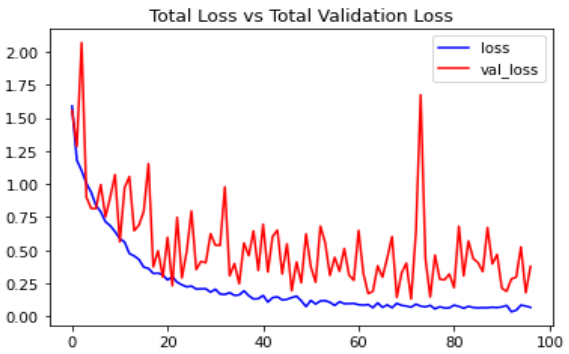
We will find that accuracy of experiments using RAVDESS landmarks improved our Accuracy rather than not using RAVDESS landmarks. Based on that we used Landmarks as a preprocessing for videos which used the same technique in “Video-based person-dependent and person-independent facial emotion recognition” Paper[20]. For now the LRCN_model with adam opt, default lr with adding batch Normalization & change in dropout rate & 64 lstm units using RAVDESS landmarks Has the best accuracy, this will be model(A).

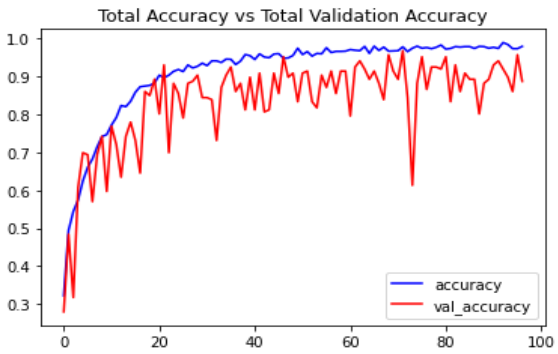
4.5.4 Different Experiments Based on Model(A) and Achieved Accepted Accuracy

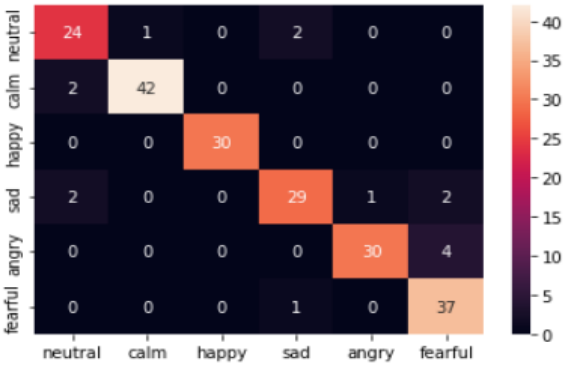
We tried other different experiments. Adding some Modification on our best Experiments such as adding and removing a layer. Changing some hyperparameters Such as kernel size, batch size and input layer Size. We will show the accuracy Results of this section in table 4.6, loss and accuracy Graphs, heatmap of confusion Matrix and classification report in figures 4.24.

Table 4-6: Accuracy of different experiments on model(A).

Expert_No	Modification	Worst	Avg	Best
4.19	Model (A) with adding a new layer	89.37%	90.73%	92.75%
4.20	Model (A) with change kernel size to 4	88.9%	91.2%	93.24%
4.21	Model (A) with change kernel size to 7	88.41%	91.17%	94.69%
4.22	Model (A) with change kernel size to 5	91.30%	93.27%	96.62%
4.23	Model (A) with change kernel size=7 && batch size=12	90.34%	91.79%	93.72%
4.24	Model (A) with batch size=16 & kernel=5	89.86%	91.63%	94.20%
4.25	Model (A) with remove first layer and change three input layers size to 128	88.89%	91.53%	93.72%
4.26	Model (A) with remove first layer and change three input layers size to 128 && kernel size to 5	85.5%	90.33%	93.24%
4.27	Model (A) with change loss function to Poisson()	90.82%	91.14%	91.30%

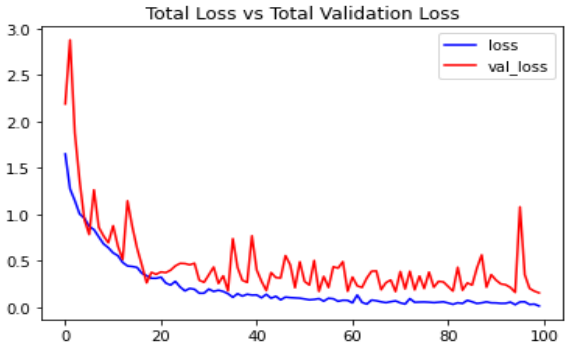


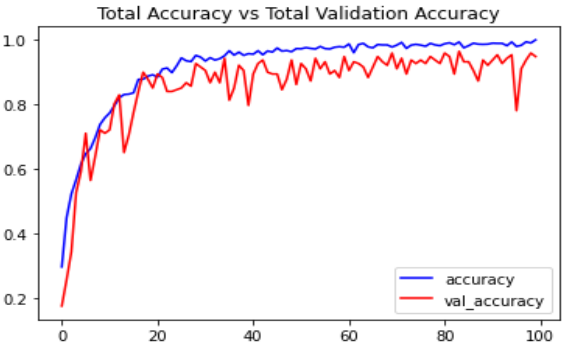


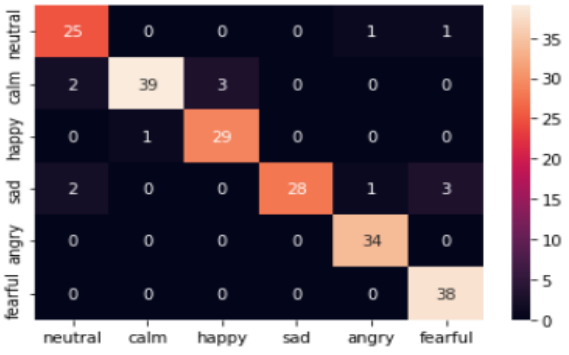


	precision	recall	f1-score	support
0	0.86	0.89	0.87	27
1	0.98	0.95	0.97	44
2	1.00	1.00	1.00	30
3	0.91	0.85	0.88	34
4	0.97	0.88	0.92	34
5	0.86	0.97	0.91	38
accuracy			0.93	207
macro avg	0.93	0.93	0.93	207
weighted avg	0.93	0.93	0.93	207

(a)

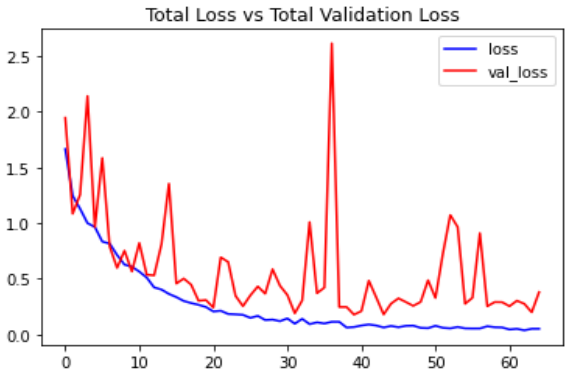


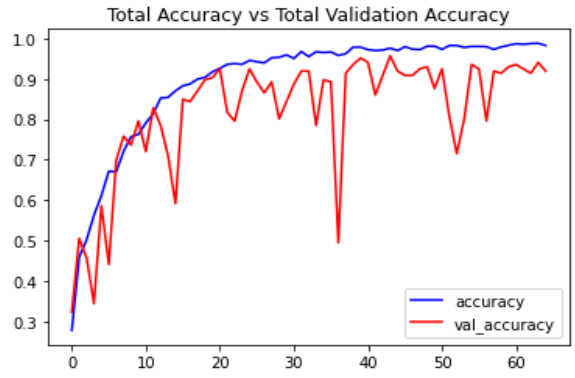




	precision	recall	f1-score	support
0	0.86	0.93	0.89	27
1	0.97	0.89	0.93	44
2	0.91	0.97	0.94	30
3	1.00	0.82	0.90	34
4	0.94	1.00	0.97	34
5	0.90	1.00	0.95	38
accuracy			0.93	207
macro avg	0.93	0.93	0.93	207
weighted avg	0.94	0.93	0.93	207

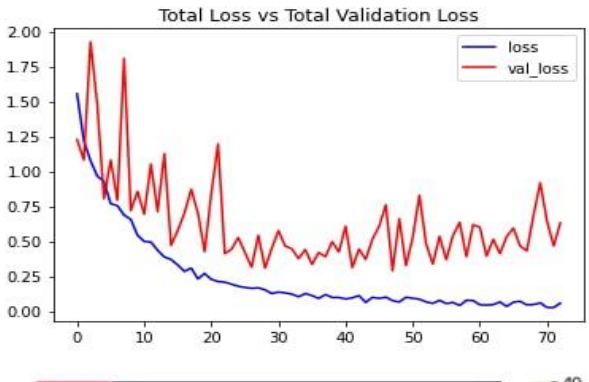
(b)

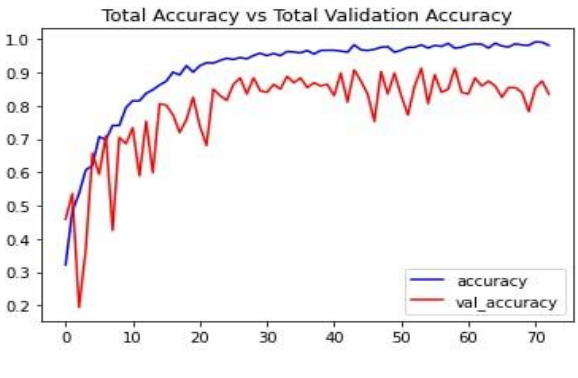




	precision	recall	f1-score	support
0	0.96	0.93	0.94	27
1	0.96	0.98	0.97	44
2	1.00	0.93	0.97	30
3	0.89	0.94	0.91	34
4	1.00	0.94	0.97	34
5	0.90	0.95	0.92	38
accuracy			0.95	207
macro avg	0.95	0.94	0.95	207
weighted avg	0.95	0.95	0.95	207

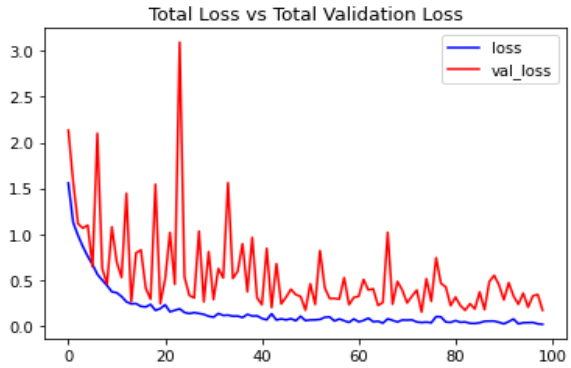
(c)

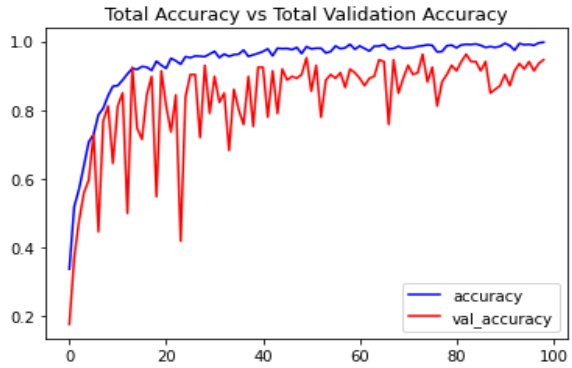




	precision	recall	f1-score	support
0	0.92	1.00	0.96	22
1	1.00	0.97	0.99	37
2	1.00	1.00	1.00	40
3	0.95	0.95	0.95	37
4	1.00	0.94	0.97	35
5	0.92	0.94	0.93	36
accuracy			0.97	207
macro avg	0.96	0.97	0.97	207
weighted avg	0.97	0.97	0.97	207

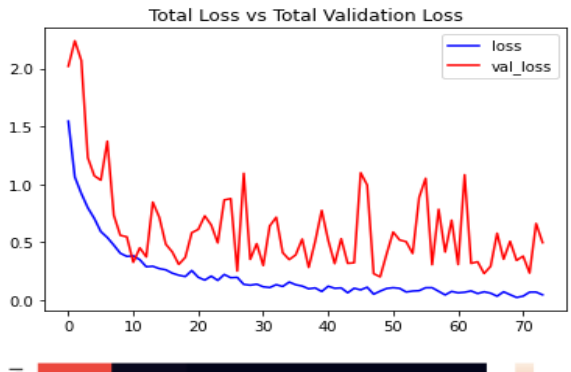
(d)

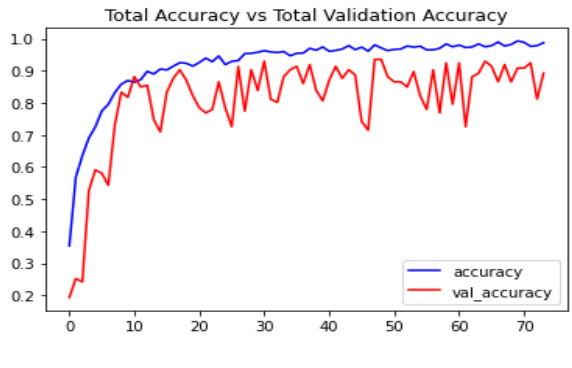




	precision	recall	f1-score	support
0	0.89	0.93	0.91	27
1	0.95	0.93	0.94	44
2	0.97	0.97	0.97	30
3	0.94	0.91	0.93	34
4	0.94	0.94	0.94	34
5	0.92	0.95	0.94	38
accuracy			0.94	207
macro avg	0.94	0.94	0.94	207
weighted avg	0.94	0.94	0.94	207

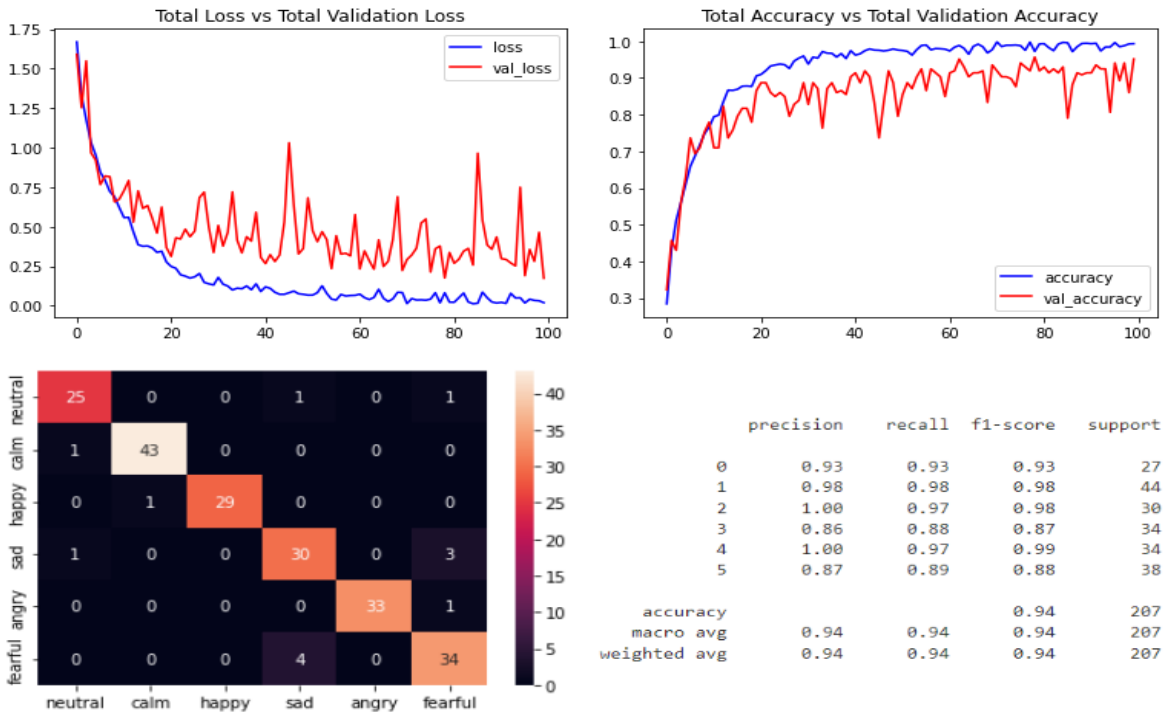
(e)





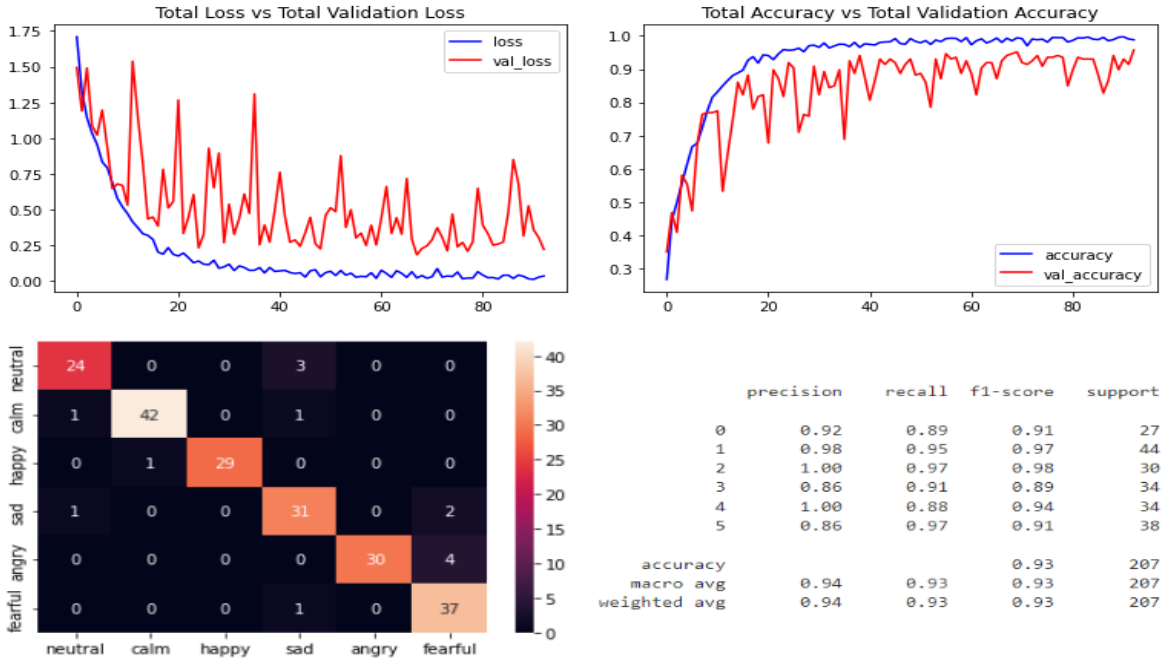
	precision	recall	f1-score	support
0	0.93	0.96	0.95	27
1	0.95	0.95	0.95	44
2	0.94	0.97	0.95	30
3	0.94	0.91	0.93	34
4	0.94	0.97	0.96	34
5	0.94	0.89	0.92	38
accuracy			0.94	207
macro avg	0.94	0.94	0.94	207
weighted avg	0.94	0.94	0.94	207

(f)



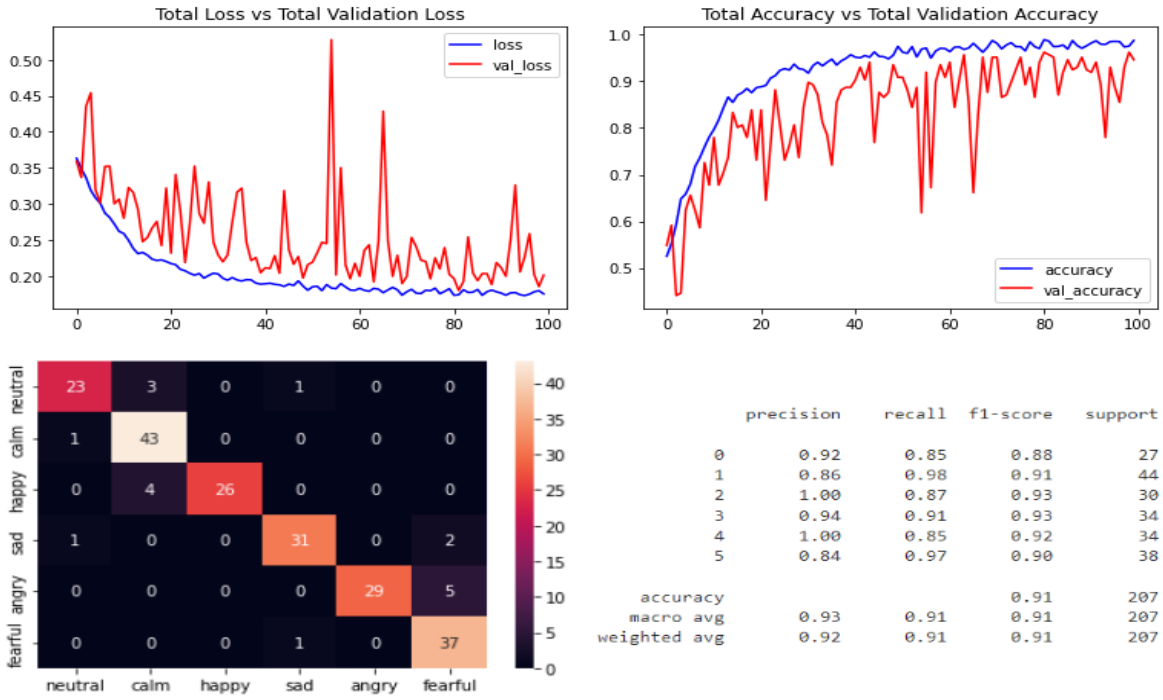
	precision	recall	f1-score	support
0	0.93	0.93	0.93	27
1	0.98	0.98	0.98	44
2	1.00	0.97	0.98	30
3	0.86	0.88	0.87	34
4	1.00	0.97	0.99	34
5	0.87	0.89	0.88	38
accuracy			0.94	207
macro avg	0.94	0.94	0.94	207
weighted avg	0.94	0.94	0.94	207

(g)



	precision	recall	f1-score	support
0	0.92	0.89	0.91	27
1	0.98	0.95	0.97	44
2	1.00	0.97	0.98	30
3	0.86	0.91	0.89	34
4	1.00	0.88	0.94	34
5	0.86	0.97	0.91	38
accuracy			0.93	207
macro avg	0.94	0.93	0.93	207
weighted avg	0.94	0.93	0.93	207

(h)



	precision	recall	f1-score	support
0	0.92	0.85	0.88	27
1	0.86	0.98	0.91	44
2	1.00	0.87	0.93	30
3	0.94	0.91	0.93	34
4	1.00	0.85	0.92	34
5	0.84	0.97	0.90	38
accuracy			0.91	207
macro avg	0.93	0.91	0.91	207
weighted avg	0.92	0.91	0.91	207

(i)

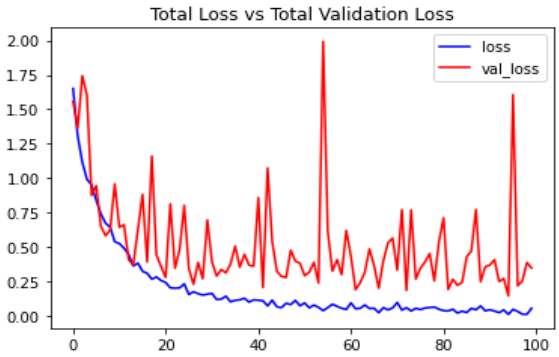
Figure 4-24: Results of experiments on model (A) in table 4.6 : (a) Model (A) with adding a new layer, (b) Model (A) with change kernel size to 4, (c) Model (A) with change kernel size to 7, (d) Model (A) with change kernel size to 5, (e) Model (A) with change kernel size=7 && batch size=12, (f) Model (A) with batch size=16 & kernel=5, (g) Model (A) with remove first layer and change three input layers size to 128, (h) Model (A) with remove first layer and change three input layers size to 128 && kernel size to 5, (i) Model (A) with change loss function to Poisson().

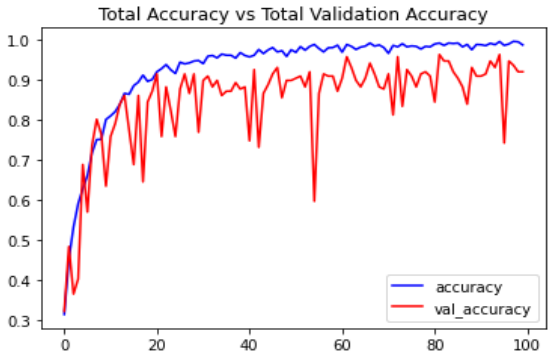
4.5.5 Different Experiments Based On Model (A) And Doesn't Increase Accuracy Significantly

Based on model (A). We tried other different experiments. But it Doesn't Improve our accuracy. We will show the accuracy results of this section in Table 4.7, Loss and accuracy graphs, heatmap of confusion matrix and classification Report in Figure 4.25.

Table 4-7: Accuracy of different experiments on model (A).

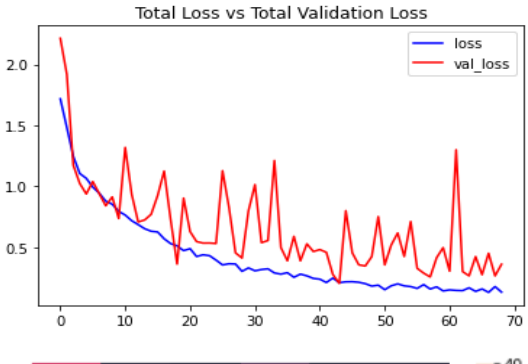
Expert_No	Modification	Worst	Avg	Best
4.28	Model (A) with remove third layer	80.68%	87.08%	89.86%
4.29	Model (A) with change kernel size to 2	86.4%	87.2%	88.41%
4.30	Model (A) with change batch size to 8	81.64%	87.60%	93.24%
4.31	Model (A) with adding dense layer	88.9%	89.2%	89.86%

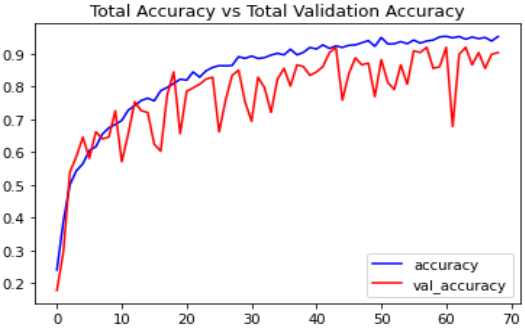




	precision	recall	f1-score	support
0	0.85	0.81	0.83	27
1	0.91	0.95	0.93	44
2	1.00	0.90	0.95	30
3	0.79	0.79	0.79	34
4	1.00	0.97	0.99	34
5	0.85	0.92	0.89	38
accuracy			0.90	207
macro avg	0.90	0.89	0.90	207
weighted avg	0.90	0.90	0.90	207

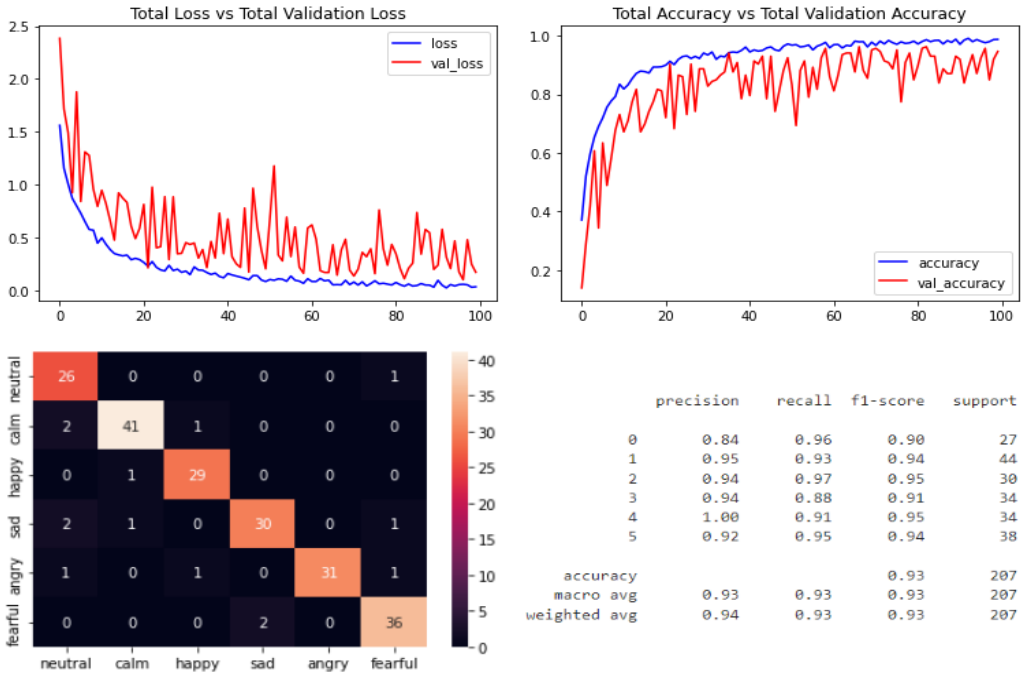
(a)





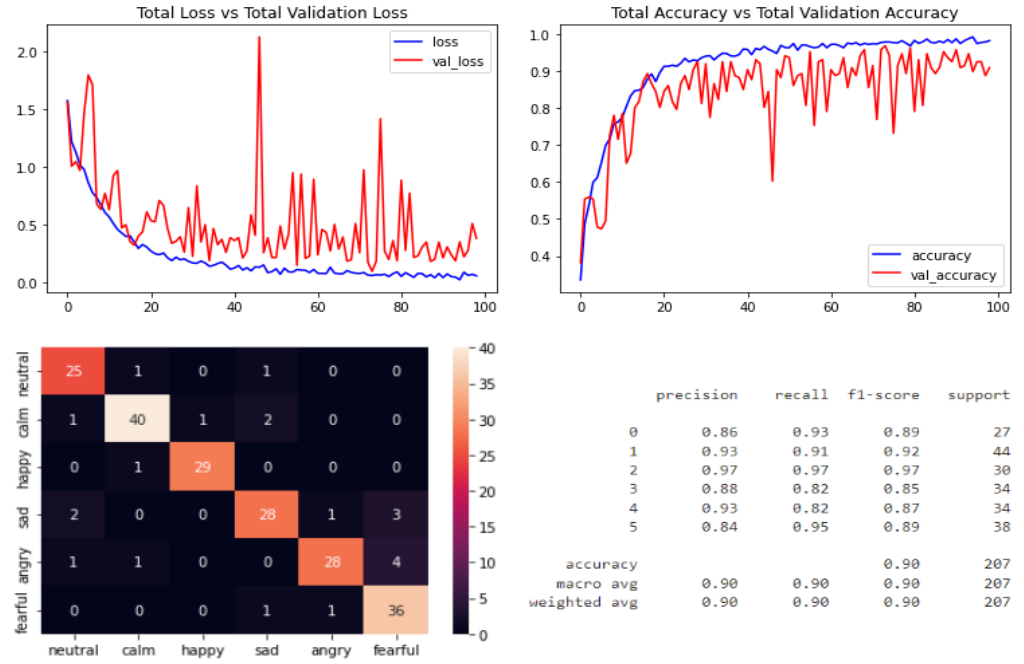
	precision	recall	f1-score	support
0	0.87	0.74	0.80	27
1	0.93	0.91	0.92	44
2	0.97	0.93	0.95	30
3	0.70	0.94	0.80	34
4	1.00	0.88	0.94	34
5	0.92	0.87	0.89	38
accuracy			0.88	207
macro avg	0.90	0.88	0.88	207
weighted avg	0.90	0.88	0.89	207

(b)



	precision	recall	f1-score	support
0	0.84	0.96	0.90	27
1	0.95	0.93	0.94	44
2	0.94	0.97	0.95	30
3	0.94	0.88	0.91	34
4	1.00	0.91	0.95	34
5	0.92	0.95	0.94	38
accuracy			0.93	207
macro avg	0.93	0.93	0.93	207
weighted avg	0.94	0.93	0.93	207

(c)



	precision	recall	f1-score	support
0	0.86	0.93	0.89	27
1	0.93	0.91	0.92	44
2	0.97	0.97	0.97	30
3	0.88	0.82	0.85	34
4	0.93	0.82	0.87	34
5	0.84	0.95	0.89	38
accuracy			0.90	207
macro avg	0.90	0.90	0.90	207
weighted avg	0.90	0.90	0.90	207

(d)

Figure 4-25: Results of experiments on model (A) in table 4.7 : (a) Model (A) with remove third layer, (b) Model (A) with change kernel size to 2, (c) Model (A) with change batch size to 8, (d) Model (A) with adding dense layer.

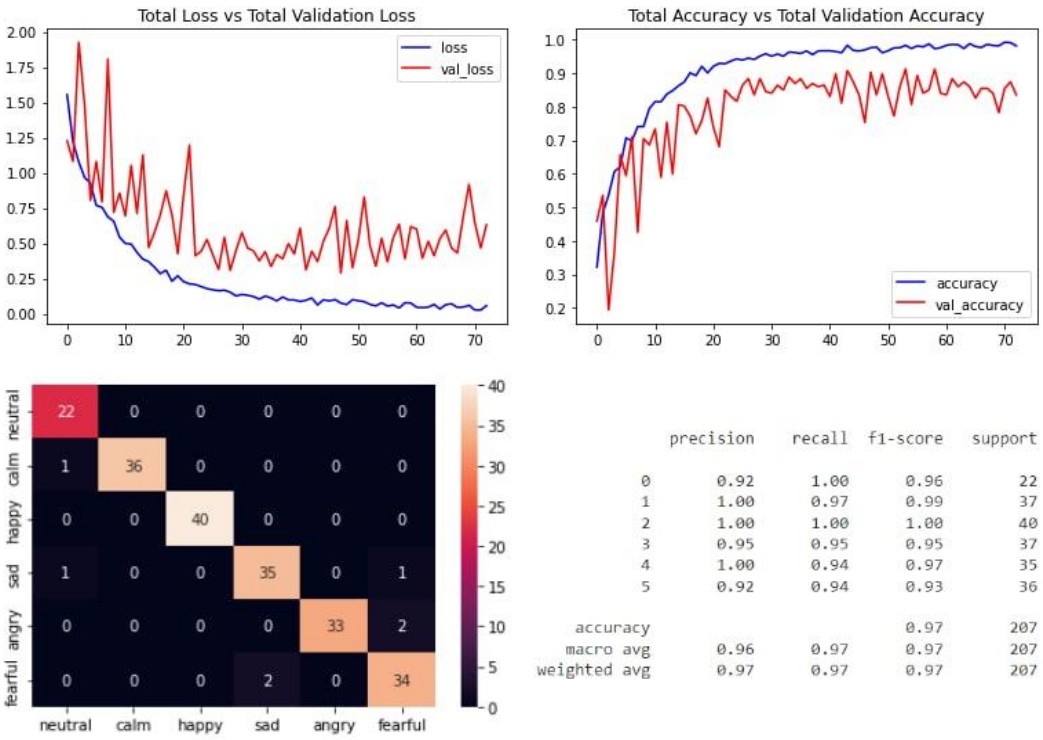
4.5.6 Results Of Final LRCN Model

In previous sections we presented all our experiments on LRCN model Results. And now we will choose our final model based on average accuracy. We Will Find that expert number 4.22 in table 4.6 has a higher average accuracy 93.27%. Our Final Model is LRCN model [32] with adding batch normalization, drop out rate is 0.1:0.2:0.3, # of LSTM units equal 64 and batch size equal 5. This model Achieved Best accuracy on a RAVDESS data set 96.62% and on a SAVEE data set 97.22%. We will find that SAVEE has a higher accuracy, but that is because of some Reasons. We found that the fact that there are only four actors in this database does Not help And the model doesn't receive a great amount of variety. Although the Video files in The test set are not included in the training set, the actors expressing That emotion Have. Therefore every face has already been seen by the trained Model and Therefore some of the facial characteristics will have already been saved And learnt. In [40], they split the dataset in two different ways. At first they split it the Normal way As they separated the files by emotions, and the testing accuracy was Very high (95%), the second way was separating the files by actors, as all files of One actor Went into the test set and the rest went to the training and validation, a 21% Accuracy is obtained which is way below the first try. This proves that the model Was Not learning but remembering certain characteristics of the actors' faces. We Can Observe that the actors in the SAVEE dataset have blue marks painted on their Faces. This could be a reason for the poor results since what the model could be Doing is remembering the positioning of these blue marks for every actor for the Different emotions. Hence, if the model has never seen the actor before, then the Model will fail to recognize the emotions. All these things explain why the SAVEE Dataset has a higher accuracy. We will show hyperparameters of the original and Final LRCN model in table 4.8, the final LRCN model diagram in figure 4.26 and Graphs of Loss, Accuracy, heatmap of confusion matrix and classification report of RAVDESS And SAVEE results on our best LRCN model in figures 4.27.

Table 4-8: Hyperparameter of original and final LRCN model.

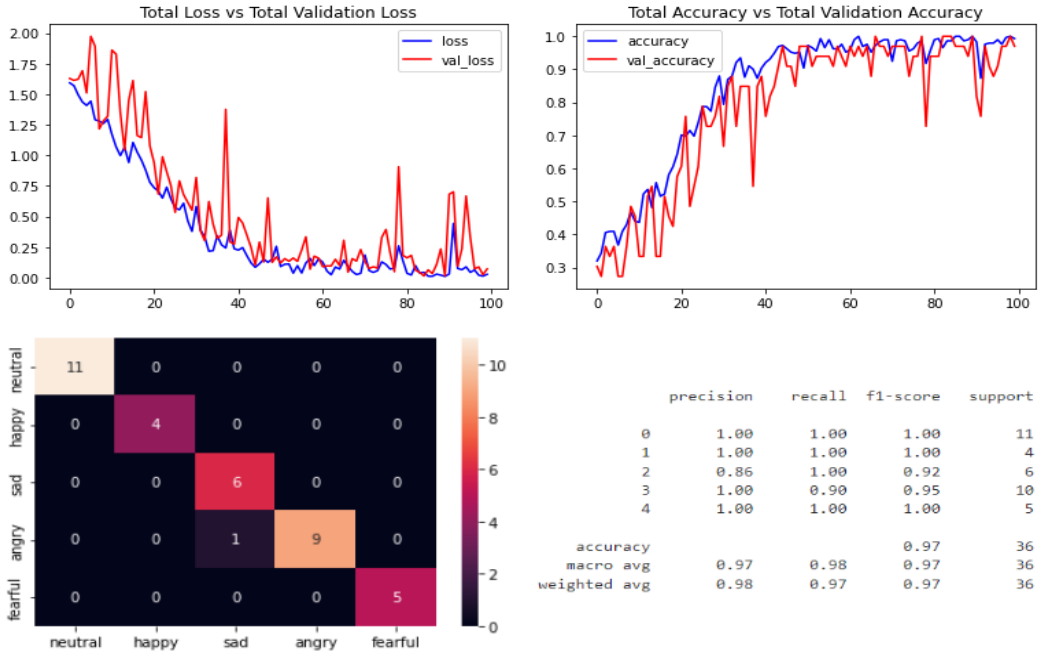
Hyper-parameter	Original LRCN	Final LRCN MODEL
Optimizer	Adam	Adam
# Epochs	70	100
Learning rate	Default	Default
Dropout rate	0.25	0.1
	0.25	0.2
	0.25	0.3

Figure 4-26: Final LRCN model.



	precision	recall	f1-score	support
0	0.92	1.00	0.96	22
1	1.00	0.97	0.99	37
2	1.00	1.00	1.00	40
3	0.95	0.95	0.95	37
4	1.00	0.94	0.97	35
5	0.92	0.94	0.93	36
accuracy			0.97	207
macro avg	0.96	0.97	0.97	207
weighted avg	0.97	0.97	0.97	207

(a)



	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	1.00	1.00	1.00	4
2	0.86	1.00	0.92	6
3	1.00	0.90	0.95	10
4	1.00	1.00	1.00	5
accuracy			0.97	36
macro avg	0.97	0.98	0.97	36
weighted avg	0.98	0.97	0.97	36

(b)

Figure 4-27: Results of final LRCN on RAVDESS and SAVEE: (a) Results on RAVDESS, (b) Results of SAVEE.

4.6 Conclusion

In our project we used two models LRCN and ConvLstm which used to Classify human activity in “Long-term Recurrent Convolutional Networks for Visual Recognition and Description” and “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting” [32][33]. Original ConvLstm model Achieved on RAVDESS data set 74.8% and on a SAVEE dataset 91.67%. The final ConvLstm was achieved on the RAVDESS data set 84.54% and on the SAVEE Dataset 94.44%. Original LRCN model was achieved on a RAVDESS data set 80.67% ± 2.18% and on a SAVEE data set 86.81% ± 2.78%. The final LRCN model Was achieved on a RAVDESS data set 93.27% ± 2.66%. And on SAVEE data set 95.24% ± 2.78%. We will find that the final LRCN model achieved an accepted Accuracy rather than our final ConvLstm model in both RAVDESS and SAVEE data Sets. Our final visual emotion recognition model is LRCN model with adding batch Normalization, dropout rate is 0.1:0.2:0.3, # of LSTM units equal 64 and batch size equal 5.

Chapter 5 **Audio And Visual Fusion System**

5.1 Introduction

After that we build models for audio and video systems to recognize emotions. We need to make fusion between two systems and give us a final prediction for Emotion based on the output of two systems. In this chapter we will explain how we Build a fusion model in detail. Section 5.2 will talk about fusion and its types. Section 5.3 will talk about the data set used for fusion and how to create it. Section 5.4 will Talk about fusion by different machine learning approaches on the RAVDESS data Set. Section 5.5 we will Compare between our models on the RAVDESS data set. Section 5.6 will show our results on the SAVEE data set. Section 5.7 Shows our conclusion.

5.2 Fusion

Fusion strategies using deep learning. Model architecture for different fusion Strategies. In the Section we talked early fusion on 5.2.1 section ,immediate fusion On section 5.2.2,late fusion on 5.2.3.Early fusion considered concatenate original or Extracted features at the input level .Immediate level fusion joins features at the input Level, but the loss is propagated back to the feature extracting model. Late fusion Aggregates predictions at the decision level [41].

5.2.1 Early Fusion

Commonly known as Data level fusion, it refers to the process of joining Multiple input modalities into a single feature vector before feeding into one single Machine learning model for training. Early fusion is applicable on raw data or Pre-processed data. Input modalities can be joined in many different ways, including Concatenation, pooling or by applying a gated unit. Synchronization of data sources Is challenging when one data source is discrete and the others are continuous. Hence, converting data sources into a single feature vector is a significant challenge In early data fusion. Fusing the original features represents early fusion type I, while Fusing extracted features, either from manual extraction, imaging analysis software Or learned representation from another neural network represents early fusion type II. We consider predicted probabilities to be extracted features, thus fusing features With predicted probabilities from different modalities is also early fusion type II. We Will show early fusion type I and type II in figure 5.1 and 5.2 respectively.

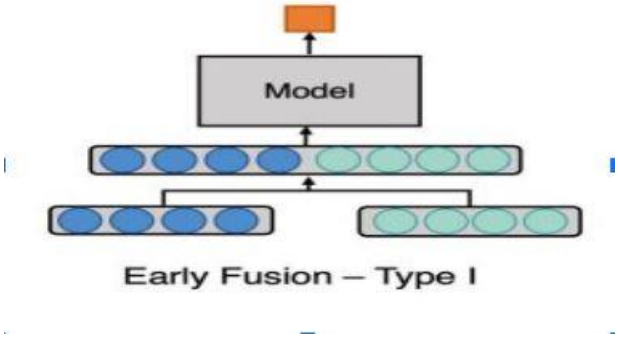


Figure 5-1: Early Fusion type I

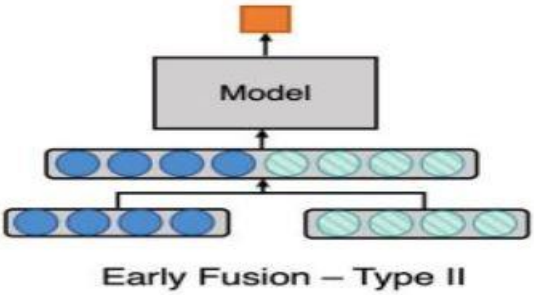


Figure 5-2: Early Fusion type II

5.2.2 Immediate Level Fusion

Is the process of joining learned feature representations from intermediate Layers of neural networks with features from other modalities as input to a final Model The architecture of intermediate fusion is built on the basis of the popular deep neural network. The key difference, compared to early fusion, is that the loss is propagated back to the feature extracting neural networks during training, thus creating better feature representations for each training iteration. It is the most Flexible method, allowing for data fusion at different stages of model training. Neural Network based multimodal data fusion has greatly improved performance. Joint Fusion is implemented with neural networks due to their ability to propagate loss from the prediction model to the feature extraction model(s). When feature Representations are extracted from all modalities, we consider this joint fusion type I. However, not all input features require the feature extraction step to be defined as Joint fusion. We will show joint fusion type I and type II in figure 5.3 and 5.4 respectively.

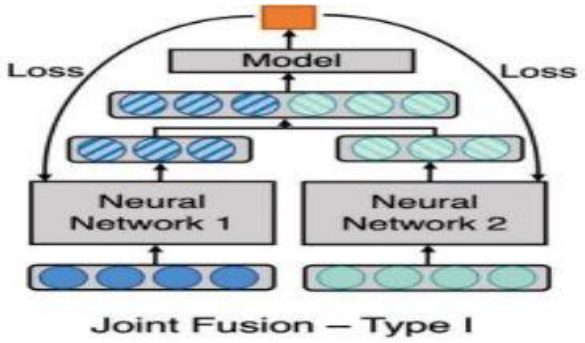


Figure 5-3: Joint Fusion type I

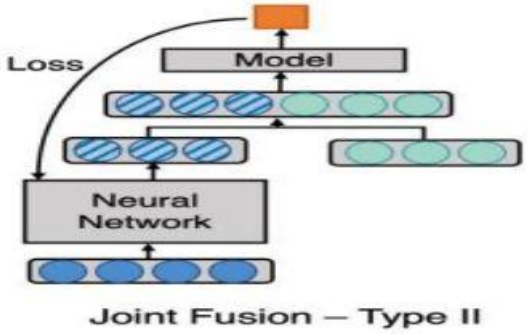


Figure 5-4: Joint Fusion type II

5.2.3 Late Fusion

Referring to the process of leveraging predictions from multiple models to Make a final decision, Late fusion uses data sources independently followed by Fusion at a decision-making stage, which is why it is often known as decision-level Fusion. This technique is much simpler than the early fusion method, particularly when the data sources are significantly varied from each other in terms of sampling Rate, data dimensionality and unit of measurement. Typically, different modalities are used to train separate models and the final decision is made using an aggregation Function to combine the predictions of multiple models. Some examples of Aggregation functions include: averaging, majority voting weighted voting or a Meta-classifier based on the predictions from each model. The choice of the Aggregation function is usually empirical, and it varies depending on the application and input modalities. Late fusion often gives better performance because errors from multiple models are dealt with independently — thus errors are uncorrelated. We will Show late fusion in figure 5.5.

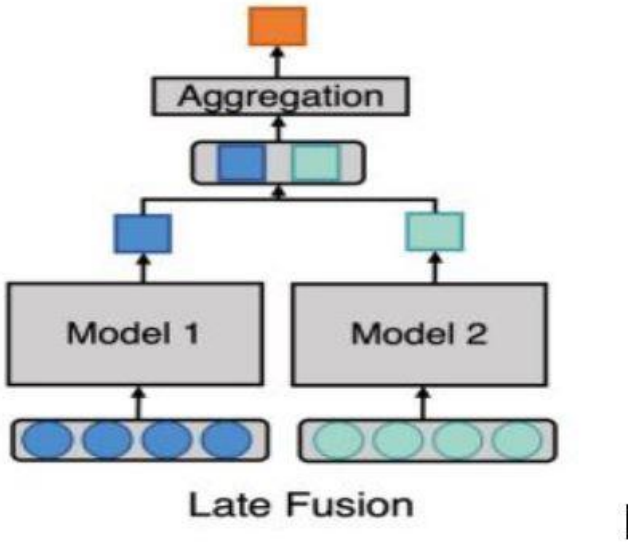


Figure 5-5: Late Fusion

In our project, We used Late fusion by machine learning techniques like Logistic Regression, SVM, KNN, MLP predictions at the decision level.

5.3 Data Set

The first step to make fusion between audio and visual emotion recognition Systems is to create a new data set. This data set will be from the output of two Systems. By using this data we can build a model to make late fusion between two Systems. In this section we will explain how to create our data set for fusion.

5.3.1 Create Data Set

We create our fusion data set from the output of two systems. And doing These steps. We will show a sample from our data set in fig 5.6 .

First In Video Model:

- Save video names of train, valid and test sets in drive.
- Save true labels of train, valid and test sets in drive.
- Training our video model.
- Get predictions of our video model for train, valid and test sets.
- Save these predictions in drive.

Second In Audio Model:

- Load video names which are used in a video model for train, valid and test Sets.
- Load audio files of these names for train, valid and test sets.
- Load labels of train, valid and test sets.
- Training our audio model.
- Get predictions of our audio model for train, valid and test sets.
- Save these predictions in drive.

Third in Fusion Model:

- Load all prediction video files of train, valid and test sets.
- Load all prediction audio files of train, valid and test sets.
- Combine prediction video files and prediction audio files in one dataframe for train, valid and test sets and becomes 6 predictions of audio and video as features and class labels as labels of the data set.
- Doing shuffle on this data set.
- Data set will be ready for ML models.

	OV_0	OV_1	OV_2	OV_3	OV_4	OV_5	OA_0	OA_1	OA_2	OA_3	OA_4	OA_5	labels	video_name
0	9.868231e-01	0.003875	0.000017	0.008416	0.000144	0.000724	9.998703e-01	4.887737e-07	9.956561e-10	0.000088	3.442699e-07	4.116531e-05	0	01-02-01-01-02-01-10.mp4
1	4.480442e-05	0.002354	0.000122	0.969830	0.001741	0.025908	2.210936e-06	2.474808e-08	5.996572e-08	0.934118	1.753438e-06	6.587785e-02	3	01-02-04-02-01-02-22.mp4
2	3.244231e-07	0.000087	0.999900	0.000006	0.000001	0.000006	7.304484e-05	3.128482e-05	9.025565e-01	0.008001	3.209596e-04	8.901695e-02	2	01-01-03-02-02-01-18.mp4
3	7.267597e-05	0.000142	0.000079	0.107955	0.000029	0.891723	1.753516e-02	1.036954e-02	2.639455e-02	0.272578	1.484431e-01	5.246800e-01	5	01-01-06-01-02-02-23.mp4
4	5.066096e-04	0.001949	0.000253	0.029472	0.000321	0.967497	4.825168e-05	1.506449e-02	3.714526e-05	0.941429	4.978651e-06	4.341640e-02	5	01-02-06-01-02-02-10.mp4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
202	5.786679e-05	0.000258	0.000029	0.000694	0.004409	0.994552	9.803350e-07	1.545456e-08	7.291856e-05	0.000120	1.130994e-03	9.986749e-01	5	01-02-06-02-01-02-04.mp4
203	1.949344e-07	0.000052	0.999928	0.000009	0.000001	0.000009	6.893139e-09	2.760478e-08	9.974328e-01	0.000008	2.552663e-03	6.631461e-06	2	01-01-03-02-01-01-24.mp4
204	1.627866e-03	0.004693	0.000329	0.004044	0.831949	0.157358	2.308228e-03	3.455839e-04	4.596739e-02	0.005468	9.189335e-01	2.697734e-02	4	01-01-05-01-01-01-13.mp4
205	4.316892e-05	0.000232	0.000092	0.003479	0.823165	0.172989	5.229506e-08	1.002611e-06	1.239179e-04	0.000012	9.968172e-01	3.045894e-03	4	01-01-05-02-01-01-03.mp4
206	4.524051e-04	0.997560	0.001718	0.000254	0.000008	0.000007	8.807982e-07	9.996986e-01	4.952280e-05	0.000251	5.928453e-09	4.641906e-09	1	01-02-02-02-01-01-06.mp4

Figure 5-6: sample of our fusion data_set

5.3.2 Preprocessing

We do shuffling on the data_set with seed number equal 42. And do MinmaxScaler on the data set with a range from 0 to 1.

5.4 Fusion by ML Approaches On RAVDESS Data Set

As mentioned in the previous section there is more than technique to make Fusion. In our project we will make late fusion by using ML Approach. Such as " A Proposal for Multimodal Emotion Recognition Using Aural Transformers and Action Units on RAVDESS Dataset" paper [23]. We will use The data set which is Created to train different ML models such as Logistic Regression, SVM, KNN and MLP.

5.4.1 Logistic Regression (LR)

Logistic regression is a supervised machine learning classification algorithm Used to predict the probability of a target class. We will use Multinomial Logistic Regression, which can classify more than 2 target classes. In our project we used Default parameters for **LogisticRegression** objects with multi_class equal **'Multinomial'.** We will show our results with different seed values of our final audio, Video model and our LR machine learning model on the RAVDESS data set in Table 5.1. We will show a heatmap of the confusion matrix of **video, audio and LR fusion** respectively, for all experiments of table 5.1 in figure 5.7.

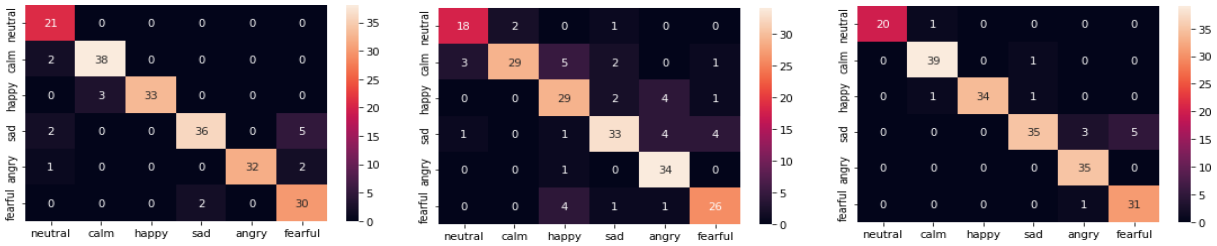
Table 5-1: Show result of audio, video and LR fusion models

# Expert	Seed_Constant	Audio	Video	LR_Fusion
5.1	27	85.99%	91.30%	94.2%
5.2	31	81.64%	91.79%	93.7%
5.3	948088	81.16%	92.75%	91.3%
5.4	271617	83.57%	95.17%	96.6%
5.5	42	85.02%	95.17%	97.1%

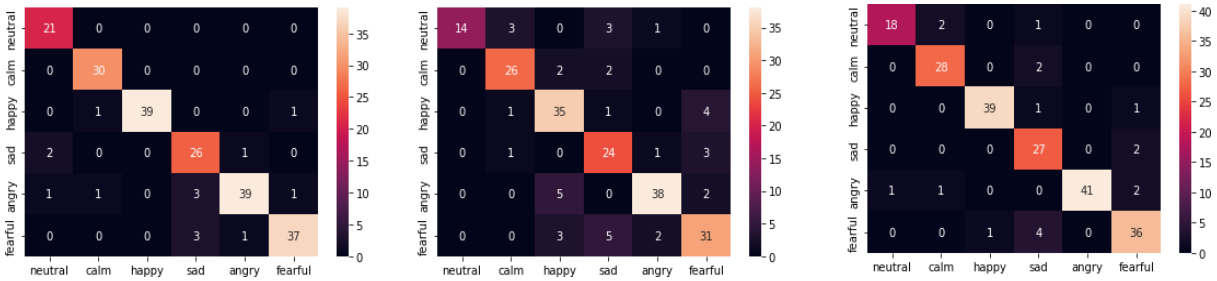
Our audio model achieved 84.27% ± 2.9%, our video model achieved 93.72% ± 2.66% and our LR fusion model achieved 94.87% ± 2.9%.


neutral calm happy sad angry fearful
25 2 0 0 0 0
1 42 1 0 0 0
0 1 29 0 0 0
2 0 0 31 0 1
1 1 0 0 30 2
1 0 0 5 0 32
21 2 2 1 0 1
1 39 3 1 0 0
0 1 27 0 0 2
1 2 2 28 0 1
0 0 0 0 32 2
0 0 1 3 3 31
27 0 0 0 0 0
0 44 0 0 0 0
0 2 26 0 0 2
2 0 0 31 0 1
1 0 0 0 32 1
1 0 0 1 1 35

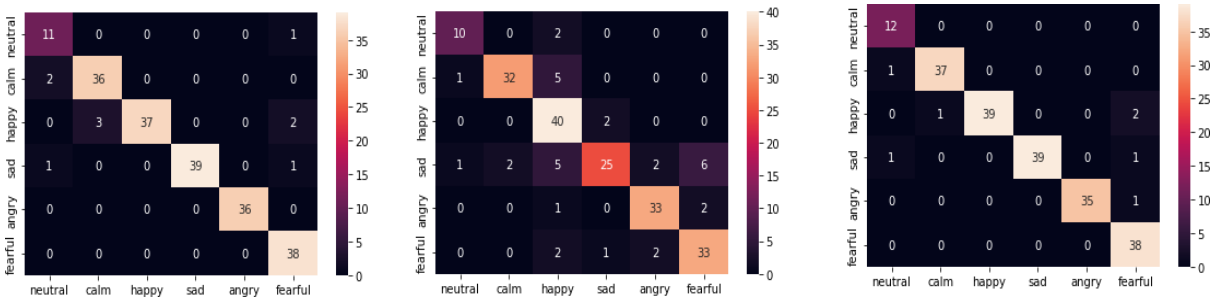
(a)


neutral calm happy sad angry fearful
21 0 0 0 0 0
2 38 0 0 0 0
0 3 33 0 0 0
2 0 0 36 0 5
1 0 0 0 32 2
0 0 0 2 0 30
18 2 0 1 0 0
3 29 5 2 0 1
0 0 29 2 4 1
1 0 1 33 4 4
0 0 1 0 34 0
0 0 4 1 1 26
20 1 0 0 0 0
0 39 0 1 0 0
0 1 34 1 0 0
0 0 0 35 3 5
0 0 0 0 35 0
0 0 0 0 1 31

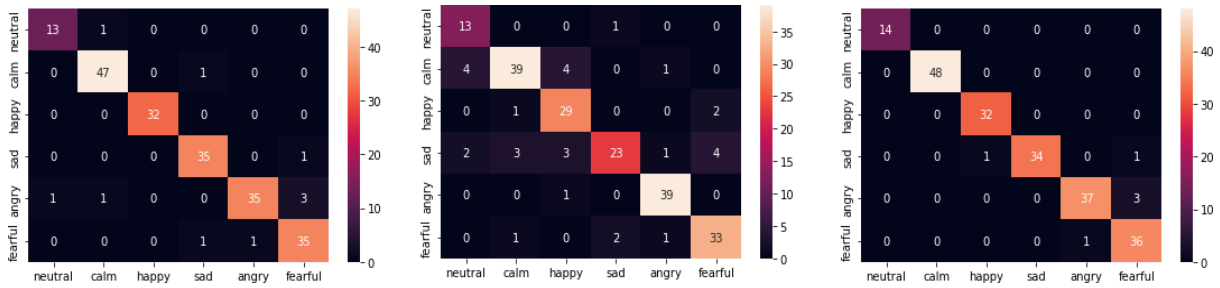
(b)


neutral calm happy sad angry fearful
21 0 0 0 0 0
0 30 0 0 0 0
0 1 39 0 0 1
2 0 0 26 1 0
1 1 0 3 39 1
0 0 0 3 1 37
14 3 0 3 1 0
0 26 2 2 0 0
0 1 35 1 0 4
0 1 0 24 1 3
0 0 5 0 38 2
0 0 3 5 2 31
18 2 0 1 0 0
0 28 0 2 0 0
0 0 39 1 0 1
0 0 0 27 0 2
1 1 0 0 41 2
0 0 1 4 0 36

(c)


neutral calm happy sad angry fearful
11 0 0 0 0 1
2 36 0 0 0 0
0 3 37 0 0 2
1 0 0 39 0 1
0 0 0 0 36 0
0 0 0 0 0 38
10 0 2 0 0 0
1 32 5 0 0 0
0 0 40 2 0 0
1 2 5 25 2 6
0 0 1 0 33 2
0 0 2 1 2 33
12 0 0 0 0 0
1 37 0 0 0 0
0 1 39 0 0 2
1 0 0 39 0 1
0 0 0 0 35 1
0 0 0 0 0 38

(d)



(e)

Figure 5-7: Fig. 5.7 : show results of audio, video and lr models with different seed constants: (a) with 27 seed constant, (b) with 31 seed constant, (c) with 948088 seed constant, (d) with 271617 seed constant and (e) with 42 seed constant.

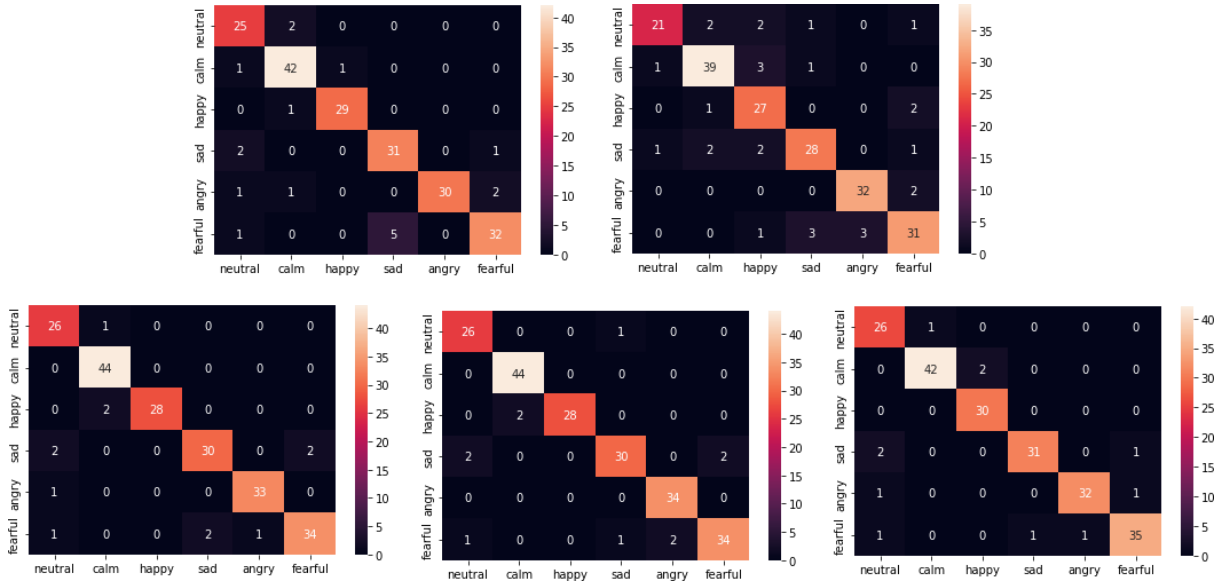
5.4.2 Support Vector Machine (SVM)

SVM is a supervised machine learning algorithm that can be used for both Classification or regression challenges. However, it is mostly used in classification Problems. In the SVM algorithm, we plot each data item as a point in n-dimensional Space (where n is a number of features you have) with the value of each feature Being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well. There are several types of kernel such as linear, rpf, sigmoid and polynomial. In our project we use linear, RBF and sigmoid kernels. We will show our results with different seed values of our final Audio and video model and our SVM machine learning model on the RAVDESS data Set in table 5.2. We will show a heatmap of the confusion matrix of **video, audio, SVM_linear, SVM_rbf and SVM_sigmoid kernels** respectively, for all experiments in table 5.2 in Figure 5.8.

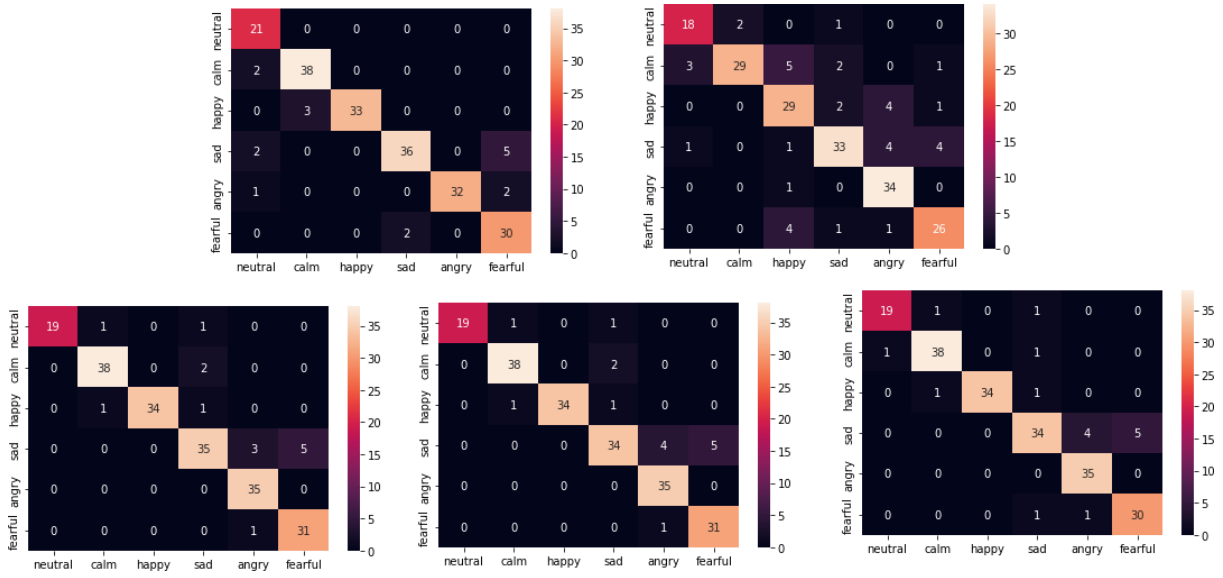
Table 5-2: Show result of audio, video and SVM fusion models.

# Expert	Seed_ Constant	Audio	Video	SVM Linear	SVM Rbf	SVM Sigmoid
5.6	27	85.99%	91.30%	94.2%	94.68%	94.68%
5.7	31	81.64%	91.79%	92.75%	92.27%	91.78%
5.8	948088	81.16%	92.75%	91.78%	91.78%	90.33%
5.9	271617	83.57%	95.17%	96.13%	95.65%	95.65%
5.10	42	85.02%	95.17%	94.68%	95.16%	96.13%

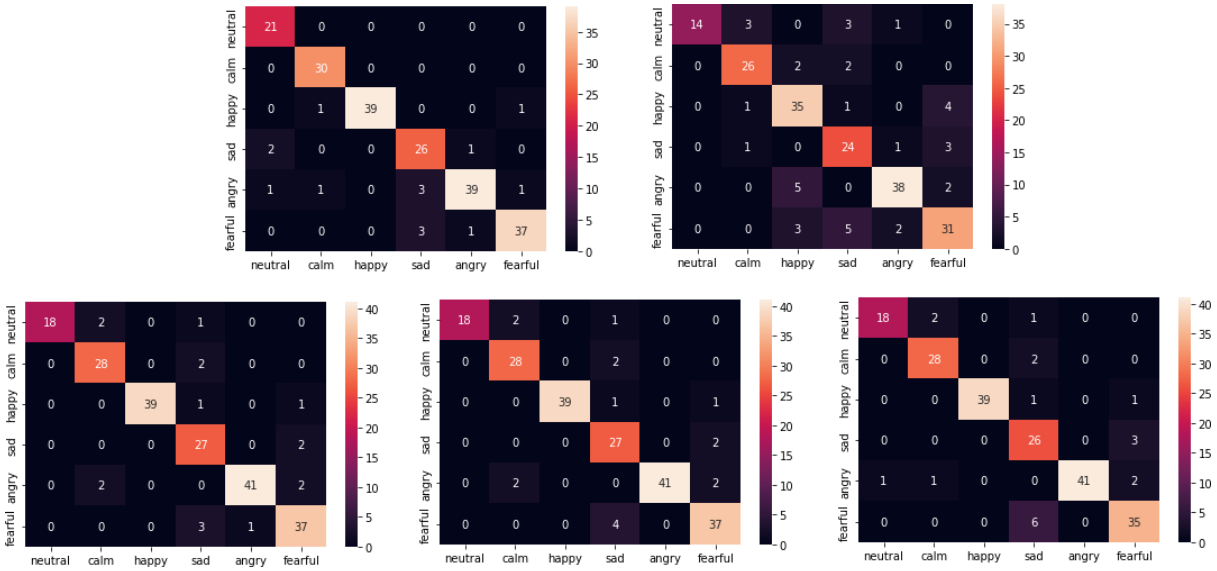
Our audio model achieved 84.27% ± 2.9%, our video model achieved 93.72% ± 2.66%, our SVM fusion model with linear kernel achieved 93.91% ± 2.18%, with rbf Kernel achieved 93.91% ± 1.94% and with sigmoid kernel achieved 93.71% ± 2.9%.



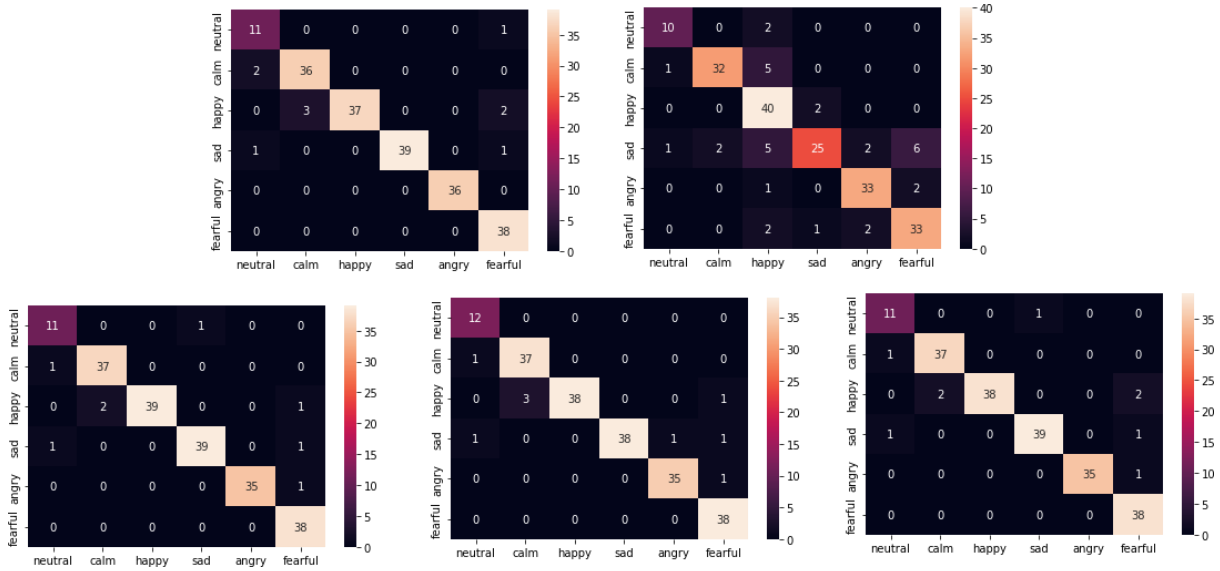
(a)



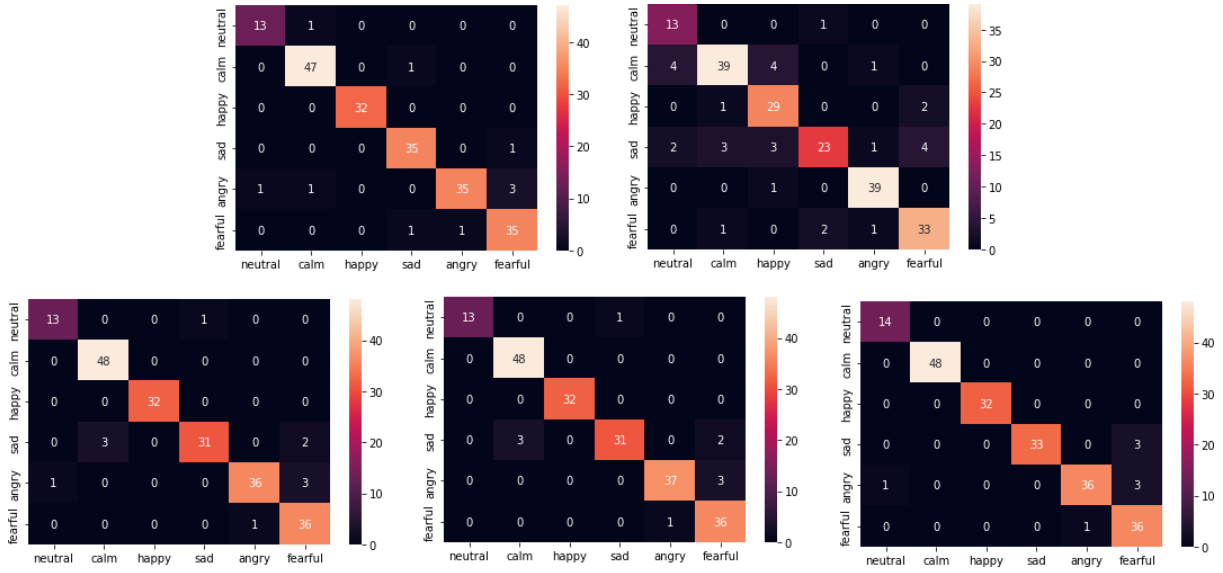
(b)


neutral calm happy sad angry fearful

(c)


neutral calm happy sad angry fearful

(d)



(e)

Figure 5-8: show results of audio,video and svm models with different seed constants: (a) with 27 seed constant, (b) with 31 seed constant, (c) with 948088 seed constant, (d) with 271617 seed constant and (e) with 42 seed constant.

5.4.3 K-Nearest Neighbor (KNN)

KNN is one of the simplest Machine Learning algorithms based on Supervised Learning technique. It stores all the available data and classifies a new data point Based on the similarity.in our project we use different k values.For our model, we will specify a range of values for 'n_neighbors' in order to see which value works best for our model. We will create a dictionary, setting 'n_neighbors' as the key and using numpy to create an array of values from 1 to 24. Our new model using grid search will take in a new k-NN classifier to find the optimal value for 'n_neighbors. After training, we can check which of our values for 'n_neighbors' that we tested performed the best. We will show our Results with different seed values of our final audio and video model and our KNN Machine learning model on the RAVDESS data set in table 5.3. We will show a heatmap of the Confusion matrix of **video, audio and KNN fusion** respectively, for all experiments in table 5.3 in figure 5.9.

Table 5-3: Show result of audio, video and KNN fusion models.

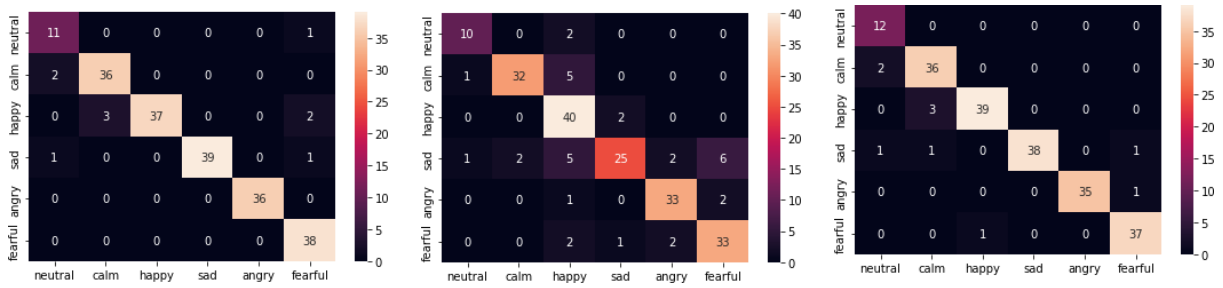
# Expert	Seed_Constant	k_value	Audio	Video	KNN_Fusion
5.11	27	4	85.99%	91.30%	94.20%
5.12	31	1	81.64%	91.79%	91.78%
5.13	948088	4	81.16%	92.75%	89.85%
5.14	271617	11	83.57%	95.17%	95.17%
5.15	42	1	85.02%	95.17%	95.17%

Our audio model achieved 84.27% ± 2.9%, our video model achieved 93.72% ± 2.66% and our KNN machine learning model achieved 93.23% ± 2.66%.

(a)

(b)

(c)



(d)

(e)

Figure 5-9: show results of audio, video and knn models with different k values: (a) k=4, (b) k=1, (c) k=4, (d) k=11 and (e) k=1.

5.4.4 Multilayer Perceptron (MLP)

MLP is known multilayer perceptron in machine learning, it is a fully connected Class of feedforward artificial neural network (ANN).it used to solve simple Regression or classification problems as it helps us to get a sneak pic about Underlying reasons in the ultra-modern models of deep learning and also obtain Information about it. It also requires a large number of parameters to operate Multi-dimensional data, due to its efforts in remembering the patterns of data. In our Project we use 40 layers and 400 iterations to train our MLP model. We will show our Results with different seed values of our final audio and video model and our MLP Machine learning model on the RAVDESS data set in table 5.4. We will show a Heatmap of the Confusion matrix of **video, audio and MLP fusion** respectively, for all experiments of table 5.4 in figure 5.10.

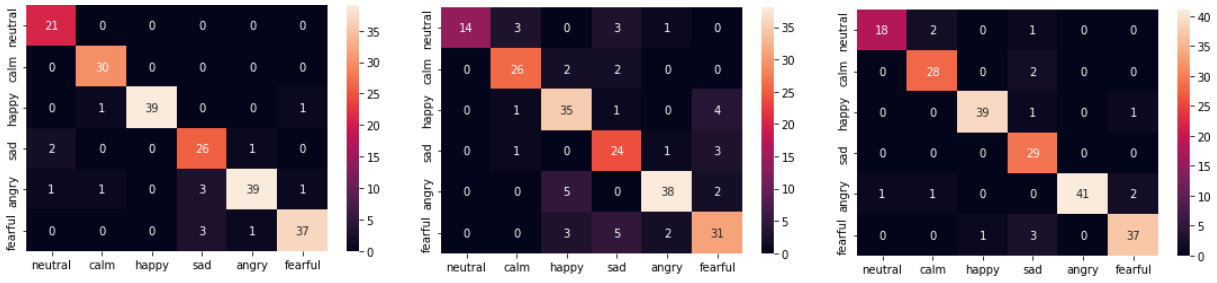
Table 5-4: Show result of audio, video and MLP fusion models.

# Expert	Seed_Constant	Audio	Video	MLP_Fusion
5.16	27	85.99%	91.30%	96.13%
5.17	31	81.64%	91.79%	94.20%
5.18	948088	81.16%	92.75%	92.75%
5.19	271617	83.57%	95.17%	96.61%
5.20	42	85.02%	95.17%	97.10%

Our audio model achieved 84.27% ± 2.9%, our video model achieved 93.72% ± 2.66% and our MLP machine learning model achieved 95.36% ± 2.18%.

(a)

(b)



(c)

(d)

(e)

Figure 5-10: show results of audio,video and mlp models with different seed constants: (a) with 27 seed constant, (b) with 31 seed constant, (c) with 948088 seed constant, (d) with 271617 seed constant and (e) with 42 seed constant.

5.5 Comparison Between Our ML Models on RAVDESS Data Set

We will compare between our ML models on the RAVDEE data set Based on average accuracy. We will find that MLP has the best average accuracy. We will show our comparison on table 5.5.

Table 5-5: Show comparison between ML models on RAVDESS dataset.

ML_Models		Avg_Accuracy
LR		94.87% ± 2.9%
SVM	Linear	93.91% ± 2.18%
	RBF	93.91% ± 1.94%
	Sigmoid	93.71% ± 2.9%
KNN		93.23% ± 2.66%
MLP		95.36% ± 2.18%

5.6 Fusion by Using MLP ML Approach on SAVEE Data Set

We run our best audio, video and fused them using MLP ML model. The Audio Model achieved 79.17 +- 4.17%. Video model achieved 95.24% ± 2.78%. Our MLP machine learning model approach achieved 95.83% ± 2.81%. We will show video, audio and MLP heatmap of the confusion matrix respectively in figure 5.11.

Figure 5-11: confusion matrix of final audio,video and MLP models on SAVEE

5.7 Conclusion

In this chapter our target is to make fusion between our Video and Audio Model by building ML model and has an accepted accuracy. So we build different ML Models such as LR, SVM, KNN and MLP. The LR model achieved 94.87% ± 2.9%, SVM with Linear kernel achieved 93.91% ± 2.18%, SVM with Rbf kernel achieved 93.91% ± 1.94%, SVM with Sigmoid kernel achieved 93.71% ± 2.9%, KNN Achieved 93.23% ± 2.66% and MLP achieved 95.36% ± 2.18% on RAVDESS data Set. We will note that MLP has the best average accuracy, achieved 95.36% ± 2.18% on the RAVDESS data set and 95.83% ± 2.81% on the SAVEE data set. Compare With " A Proposal for Multimodal Emotion Recognition Using Aural Transformers And Action Units on RAVDESS Dataset" paper [23] which achieved 86.70% on RAVDESS data set, our ML fusion model (MLP) achieved 95.36% ± 2.18% on RAVDESS data set. We will find that our ML has higher accuracy.

Chapter 6 **Conclusion & Future work**

6.1 Conclusion

Our project consists of 3 phases: audio, video and fusion. In each phase we used different techniques and tried to improve other related work accuracy.
In audio Phase, first we extract audio features using two methods Mel Spectrogram and MFCC. We tried different models using Mel spectrogram and MFCC and found that the range of accuracy of both methods are 58% - 69% and 69% - 73% respectively. we found that MFCC has higher accuracy than Mel spectrogram, so we choose MFCC to extract features from our audio files. After extracting features by MFCC, which feed into our network. We worked on two models and tried to improve them. The first model consists of two parallel CNN [28] and achieved 73% as an average accuracy. We tried to improve it and our modifications on it improved its accuracy by 7% Approximately. Our improved model achieved 80.86% as the best accuracy on the RAVDESS dataset. The second model, we implement the model architecture in "Emotion Recognition from Speech Signals Using Machine Learning and Deep Learning Techniques" paper [10], Original model achieved 70.83% on the RAVDESS dataset and classified 8 emotions. Our modifications On CNN model improved accuracy by 17% approximately. Our model achieved 87.4% as the best accuracy on the RAVDESS dataset and classified 6 emotions. And 83.3% as the best accuracy on SAVEE dataset.

Table 6-1 : Show our results on audio phase.

Paper	Year	Dataset	Paper accuracy	Our accuracy
Emotion Recognition from Speech Signals Using Machine Learning and Deep Learning Techniques [10]	2021	RAVDESS	70.38% (8 emotions)	87% (6 emotions)
		SAVEE	61.46%	83.3%

In the Video phase we used landmarks to extract frames from video. We Used two models LRCN and ConvLSTM which used to Classify human activity in Two papers [32][33]. The original ConvLstm model Achieved on RAVDESS data set 74.8% and on a SAVEE dataset 91.67%. Our Modifications on ConvLSTM model improved accuracy by

10% approximately on RAVDESS. The final ConvLstm was achieved on the RAVDESS data set 84.54% and On the SAVEE Dataset 94.44%.

The second model LRCN, the original LRCN model was achieved on the RAVDESS data Set 80.67% ± 2.18% and on the SAVEE data set 86.81% ± 2.78%. Our modifications On LRCN model improved accuracy by 13% approximately on the RAVDESS dataset. The final LRCN model Was achieved on the RAVDESS data set 93.27% ± 2.66%. And on the SAVEE data set 95.24% ± 2.78%.

Table 6-2: show our results on video phase on 6 emotions.

Paper	Year	Dataset	Paper accuracy	Our accuracy
A CNN-LSTM BASED DEEP NEURAL NETWORKS FOR FACIAL EMOTION DETECTION IN VIDEOS [21]	2021	RAVDESS	65.35%	93.27% ± 2.66%
Human Emotion Recognition in Video using Subtraction Pre-Processing [22]	2019	RAVDESS	79.74%	

In the fusion phase, we fused between our Video and Audio Model by building ML models and has An accepted accuracy. So we build different ML Models such as LR, SVM, KNN and MLP. MLP achieved 95.36% ± 2.18% on the RAVDESS data Set. MLP has the best Average accuracy which achieved 95.36% ± 2.18% on the RAVDESS data set and 95.83% ± 2.81% on the SAVEE data set. Compare With " A Proposal for Multimodal Emotion Recognition Using Aural Transformers And Action Units on RAVDESS Dataset" paper [23] which achieved 86.70% on RAVDESS data set, our ML fusion Model (MLP) achieved 95.36% ± 2.18% on RAVDESS data set.

Table 6-3 : show our results on fusion phase.

Paper	Year	Dataset	Paper accuracy	Our accuracy
A Proposal for Multimodal Emotion Recognition Using Aural Transformers and Action Units on RAVDESS Dataset [23]	2021	RAVDESS	86.70% (LR) (8 emotions)	95.36% ± 2.18% (MLP) (6 emotions)
Multimodal Emotion Recognition on RAVDESS Dataset Using Transfer Learning [11]	2021	RAVDESS	80.08% (8 emotions)	

6.2 Future work

We will deploy our audio and visual system in different applications. such as an **Emotion recognition app for online admissions and interviews.** which can be used to understand how candidates feel during interviews and how they react to certain questions. This information Can be used to optimize interview structure for future candidates and Streamline the application process. **Emotion analysis for an online education app.** which can be used for online education is an ideal way to analyze the Online student journey and improve it where necessary. Assess Schools course materials, teaching styles, structure and layout by way of emotional feedback as student's go through each module in Real-time. **Emotion analysis for marketing.** which can be used by different companies to gauge consumer Mood towards their products, brands, marketing efforts, staff or In-location experiences. Understanding customer emotions is vital to Ensure business growth and enhance experiences. **Automotive industry and emotion analysis.** Car manufacturers around the world are increasingly focusing on Making cars more personal and safe for people to drive. Using facial Emotion detection smart cars can alert the driver when he is feeling Drowsy and in turn help to decrease road casualties.
In Research The model can also be improved further to improve its quality through New modifications to the model to get a better accuracy, and the quality of the Current model can be used in transfer learning in different fields.

ABBREVIATIONS

ABBREVIATION	SENTENCE
AO	audio overlapping
VA	video frames augmentation
MFCC	Mel-frequency cepstral coefficients
CNN	convolutional neural network
LSTM	Long short-term memory
RAVDESS	The Ryerson Audio-Visual Database of Emotional Speech and Song
SAVEE	Surrey Audio-Visual Expressed Emotion
WA	weighted accuracy
MLP	Multi layer perceptron
RNN	Recurrent neural network
LRCN	Long-term recurrent convolutional network
ConvLstm	Convolutional Long ShortTerm Memory
Lr	learning rate
Opt	optimizer
LR	Logistic Regression
SVM	Support Vector Machine
MLP	Multilayer Perceptron
KNN	K-Nearest Neighbor
Crema-d	Crowd-sourced Emotional multi-modal Actors Dataset
MRPN	multi-modal Residual Perceptron Network

References

[1] G. Mendels, "Towards Data Science," 18 Nov 2019. [Online]. Available: https://towardsdatascience.com/how-to-apply-machine-learning-and-deep-learning-methods-to-audio-analysis-615e286fcbbc.

[2] L. T. Alberto Albiol, "Video preprocessing for audiovisual indexing," IEEE Computer Society Annual Symposium on VLSI, p. 4, 2014.

[3] N. Ellarby Sánchez, "Emotion Detection through Audio and Video," 2021.

[4] A. H. L. Lu, L. . M. T. Ö. and . E. B. , "Audio Representation," MA: Springer US, p. 8, 2009.

[5] N. E. S. S. M. A. C. Q. F. E. Moncholi, "Final Degree Thesis “Emotion Detection through Audio and Video," June 2021.

[6] W. contributors, 10 February 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Spectrogram&oldid=1070997200.

[7] A. Bajaj, 4 January 2022. [Online]. Available: https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide#:~:text=Performance%20metrics%20are%20a%20part,a%20metric%20to%20judge%20performance..

[8] S. R. a. F. A. R. Livingstone, "The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English," PloS one, 2018.

[9] P. a. S. H. Jackson, "Surrey audio-visual expressed emotion (savee) database," 2014.

[10] G. e. a. Kataria, "Emotion Recognition from Speech Signals Using Machine Learning and Deep Learning Techniques," Springer, p. 10, 2021.

[11] C. Luna-Jiménez, D. Griol, Z. Callejas, R. Kleinlein, J. Montero and F. Fernández-Martínez, "Multimodal Emotion Recognition on RAVDESS Dataset Using Transfer Learning," Sensors, 2021.

[12] B. Puterka and J. Kacur, "Time Window Analysis for Automatic Speech Emotion Recognition," IEEE, p. 4, 16-19 9 2018.

[13] J. A. R. ,. W. B. Abdul Malik Badshah, "Speech Emotion Recognition from Spectrograms with Deep Convolutional Neural Network," International Conference on Platform Technology and Service (PlatCon), February 2017.

[14] X. X. C. a. C. C. Wang, "Human emotion recognition by optimally fusing facial expression and speech feature," 2020.

[15] O. S. A. A. Mohamed, "Arabic Speech Emotion Recognition Employing Wav2vec2. 0 and HuBERT Based on BAVED Dataset," 2021.

[16] A. e. a. Aftab, "Light-SERNet: A lightweight fully convolutional neural network for speech emotion recognition.," 2021.

[17] S. O. S. R. D. S. a. D. M. Sarala Padi, "Improved Speech Emotion Recognition using Transfer Learning and Spectrogram Augmentation," International Conference on Multimodal Interaction (ICMI '21),, p. 8, 2021.

[18] A. M. B. &. N. R. &. N. U. &. J. A. &. K. M. &. M. Y. L. 1. &. S. K. &. S. W. Baik1, springer Science+Business Media, LLC , June-October 2017.

[19] S. M. S. A. e. a. Abdullah, "Multimodal emotion recognition using deep learning," Journal of Applied Science and Technology Trends 2.02, pp. 52-58, 2021.

[20] N. E. B. a. H. D. Hajarolasvadi, "Video-based person-dependent and person-independent facial emotion recognition.," pp. 1049-1056, 2021.

[21] &. S. R. Arnold Sachith A Hans, "A CNN-LSTM BASED DEEP NEURAL NETWORKS FOR FACIAL EMOTION DETECTION IN VIDEOS," INTERNATIONAL JOURNAL OF ADVANCES IN SIGNAL AND IMAGE SCIENCES, pp. 11-20, 2021.

[22] Z. e. a. He, "Human emotion recognition in video using subtraction pre-processing," International Conference on Machine Learning and Computing, November 2019.

[23] *. ,. R. K. 1. ,. D. G. 2. ,. Z. C. 2. ,. J. M. M. 1. F. F.-M. Cristina Luna-Jiménez 1, "A Proposal for Multimodal Emotion Recognition Using Aural Transformers and Action Units on RAVDESS Dataset," ,Grupode Tecnología del Habla y Aprendizaje Automático (THAU Group), p. 23, 30 12 2021.

[24] Y. H. M. C. J. C. P. L. A. K. Yaxiong Ma, "Audio-visual emotion fusion (AVEF): A deep efficient weighted approach,".

[25] ,. D. G. 2. ,. Z. C. 2. ,. R. K. 1. ,. J. M. M. 1. a. F. F.-M. Cristina Luna-Jiménez 1, "Multimodal Emotion Recognition on RAVDESS Dataset Using Transfer Learning," 18 November 2021.

[26] X. Chang and W. Skarbek, "Multi-modal Residual Perceptron Network for Audio-Video Emotion classification," Preprints, 2021.

[27] S. Zhang, S. Zhang, T. Huang, W. Gao and Q. Tian, "Learning Affective Features with a Hybrid Deep Model for Audio-Visual Emotion Recognition," Transactions on Circuits and Systems for Video Technology, 2017.

[28] LinaZenkov, 6 Nov 2020. [Online]. Available: https://github.com/IliaZenkov/transformer-cnn-emotion-recognition.

[29] Kosta-jo, "github," 31 mar 2021. [Online]. Available: https://github.com/Data-Science-kosta/Speech-Emotion-Classification-with-PyTorch.

[30] "github," 13 April 2021. [Online]. Available: https://github.com/mkosaka1/Speech_Emotion_Recognition/blob/master/3.%20Transfer_Learning%20-%20Initial_Model.ipynb.

[31] rajamohanharesh, "github," 14 December 2019. [Online]. Available: https://github.com/rajamohanharesh/Emotion-Recognition/blob/master/Codes/4.6%20Final%20Model%20Ravdess%202D%20CNN%20_Log%20Mel%20Spectogram_6L_64_.ipynb.

[32] L. A. H. M. R. S. V. S. G. S. T. D. Jeff Donahue, "Long-Term Recurrent Convolutional Networks for Visual Recognition and Description," *IEEE,* 2015.

[33] X. S. Z. C. H. W. D.-Y. Yeung, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," 2015.

[34] "MathWorks," [Online]. Available: https://www.mathworks.com/help/deeplearning/deep-learning-data-management-and-preprocessing.html.

[35] D. A. e. a. Pitaloka, "Enhancing CNN with preprocessing stage in automatic emotion recognition.," pp. 523-529, 2017.

[36] "datagen," [Online]. Available: https://datagen.tech/guides/face-recognition/facial-landmarks/.

[37] R. a. L. S. R. a. R. F. A. Swanson, june 2019. [Online]. Available: https://zenodo.org/record/3255102/export/hx#.YqvSfXYzZPY.

[38] "github," 2019. [Online]. Available: https://github.com/TadasBaltrusaitis/OpenFace/wiki/Action-Units.

[39] "Analytics India Magazine," [Online]. Available: https://analyticsindiamag.com/when-to-use-one-hot-encoding-in-deep-learning/#:~:text=of%20a%20model.-,One%20Hot%20Encoding%20is%20a%20common%20way%20of%20preprocessing%20categorical,corresponds%20to%20its%20original%20category.

[40] N. Ellarby Sánchez, "Emotion Detection through Audio and Video.," 2021.

[41] "ResearchGate," [Online]. Available: https://www.researchgate.net/figure/Fusion-strategies-using-deep-learning-Model-architecture-for-different-fusion_fig2_346295743. [Accessed 9 jun 2022].