ECE 9063: Data Analytics Foundations

Final Project

Human Image Sentiment Analysis for Enhanced Mental Health Monitoring

Group 8

Xiaoyun Huang (xhuan427@uwo.ca)

Harshpreet Singh (hsing368@uwo.ca)

Jayakarthiga Arumugam (jarumuga@uwo.ca)

Rhea Sera Rodrigues (rrodri27@uwo.ca)

Supratim Basu (sbasu22@uwo.ca)

Instructor: Dr. Alexandra L'Heureux

Table of Contents

Abstract:

The focus of this report and project is to combine mental health with technology to gain insight into people's emotional and mental states through image processing. We use advanced technology to study different facial expressions and emotions using the FER-2013 dataset to detect potential mental health problems at an early stage through image analysis. Imagine being able to help someone who might be struggling just by looking at his/her photo. Our project aims to do just that.

Human emotions are perceptible states of mind that are spontaneous and have a clear relationship to facial expressions but can vary considerably. Face recognition is a challenging research field, and previously manually defined algorithms such as Histogram of Oriented Gradients (HOG) and Scale Invariant Feature Transform (SIFT) have been used for feature extraction [1]. In recent years, the application of machine learning (ML) and neural networks (NN) has advanced in emotional recognition. In this report, we use fully connected neural networks (FC) and convolutional neural networks (CNN) to extract features from images for emotion recognition. In addition, we optimize the accuracy of feature extraction by extracting facial features using the dlib library. These flag points are combined with the FC model to improve the model performance and compared with the CNN model.

We use grayscale images from the FER-2013 dataset to classify expressions into seven emotions, including happy, sad, neutral, fearful, angry, disgusted, and surprised. We used normalization and hyperparameter tuning to improve model accuracy and avoid overfitting. We implemented Model 1, which prepares real images for Haar Cascading analysis, which transforms images into detailed facial networks to improve the analysis of emotional expression. The output of Model 1 is used as input to the FC and CNN models to test its accuracy. The best pattern recognition skills will ensure effective intervention and have a positive impact on mental health. This project underscores our commitment to leveraging technology to perform careful and accurate sentiment analysis.

The FER-2013 dataset plays a key role in this project, as it provides a diverse range of facial expressions that provide valuable information about human emotions. Our goal is to use this technology to create a simple and accessible way to monitor the condition of people, which contributes to the improvement of mental health. We hope to shape the future so that technology becomes a partner that supports mental health and positively impacts people and lives.

Index terms: Facial emotion recognition; Classification; Fully connected layer; Convolutional neural network [2], model tuning; Hyperparameter Tuning; GridSearchCV, convolutional layer; Connection layer; Hidden layer; Neurons; Learning speed; Stop.

1. Introduction

Our project explores the intersection of mental health and technology and aims to analyze facial expressions using the FER-2013 dataset to improve mental health. We used machine learning and advanced neural network algorithms, specifically fully connected neural networks (FC) and convolutional neural networks (CNN), enhanced by the features of the dlib library. We work with FER-2013 grayscale images and classify the expressions into seven emotions. This report presents our methodology, data pre-processing and model estimation, all aimed at improving mental health.

1.1 Setting the Scene: Understanding the Analogy

In this section, we draw a parallel between the unique challenge of understanding human emotions through facial expressions and the more familiar task of predicting energy consumption in large buildings using sensor data. By creating this analogy, we want to contextualize the innovative approach of sentiment analysis. This comparison serves as a basis for readers to understand the complexity and potential impact of our project.

1.2 Our Mission: Accuracy in predicting emotions

Here, we express our main task of achieving accurate emotion prediction through facial expression analysis. By highlighting the analogy to predicting energy use, we emphasize our goal of reshaping approaches to emotional well-being. The mention of "precise emotion prediction" underlines our commitment to accuracy and reliability in assessing individuals' emotional states.

1.3 Why it matters: transforming mental health support

In this section, we delve into the importance of our project to change the emotional well-being of society. Drawing another parallel to predicting energy use through planning, we highlight the potential of our project to change the paradigm of mental health support through proactive intervention. This underlines the social impact and forward-looking nature of our initiative.

1.4 How we do it: Exploring the methodology

This section describes our methods in detail. We emphasize the departure from traditional methods and the adoption of advanced neural networks, including fully connected neural networks (FC) and convolutional neural networks (CNN). The mention of feature points and the Haar cascade for data preparation underscores our commitment to model robustness. Key steps such as data class balancing, image normalization and efficient data transformation are outlined to give readers an understanding of our approach.

1.5 Structure of the Report: Navigation of the Sections

This subsection provides the reader with a road map that guides the reader through the different sections of the report. Starting with our strategies using Haar cascade and neural networks (Section 2) for data preparation, class imbalance extraction and image transformation (Section 3), we provide a comprehensive overview of our approach. Subsequent sections cover feature creation (Section 4), model evaluation (Section 5), and model and performance reporting (Section 6). Finally, we conclude our study with an understanding of the strong relationship between mental health and technology (Section 7). This organizational overview prepares the reader for the detailed overview they will receive of each section of the report.

2. Background

Our main objective here was to assess the accuracy of two fully adapted models - one fully connected and the other based on convolutional neural networks （CNNs）. This comparative analysis aims to identify the model that performs best in emotion detection for mental health monitoring.

First, we implemented Model 1, which processes real images by Haar cascading. This initial model transforms the image into a detailed facial mesh to enhance the analysis of emotional expression. Then, we use the prepared images to evaluate them by fully connected model and CNN model, respectively. By examining their performance, we can gain more insights into which model has higher accuracy in emotion detection.

This process is intended to inform decision making for mental health support. Optimal model identification ensures effective interventions and helps promote proactive mental health care. This project underscores our commitment to leveraging technology for detailed and accurate sentiment analysis.

2.1 Haar Cascade Classifier

The Haar Cascade algorithm, developed by Viola and Jones in 2001, is renowned for its rapid and effective object detection capabilities. Originating from the field of computer vision, this algorithm utilizes Haar-like features inspired by Haar wavelets. These features play a pivotal role in identifying subtle patterns within images, making it particularly adept at discerning intricate details. Initially designed for object detection, the algorithm's unique characteristics extend its applicability to various scenarios requiring swift analysis of patterns and trends, such as real-time data processing.

One key aspect of the Haar Cascade algorithm is its utilization of Haar-like features. These features serve as detectors for abrupt changes in pixel intensities, enabling the algorithm to identify critical patterns within images. To enhance precision, the algorithm adopts a grid-based segmentation approach, dividing the image into smaller regions for localized analysis. This grid-based strategy allows for a detailed examination of specific areas, contributing to a nuanced understanding of patterns relevant to forecasting.

A distinctive feature of the algorithm is the cascading window technique. This iterative approach involves the algorithm processing various windows within the image, computing Haar-like features, and iteratively classifying them. This iterative refinement ensures a robust identification process, enabling the algorithm to detect objects rapidly and accurately across diverse scales and positions within images. While traditionally recognized for object detection, the algorithm's efficiency, and speed position it as a potentially valuable tool for forecasting applications, particularly in scenarios requiring swift analysis of patterns and trends, such as real-time data processing.

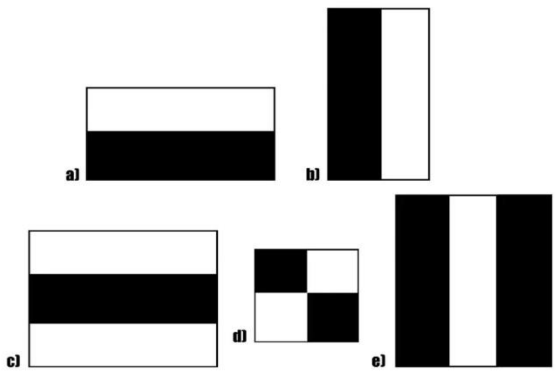
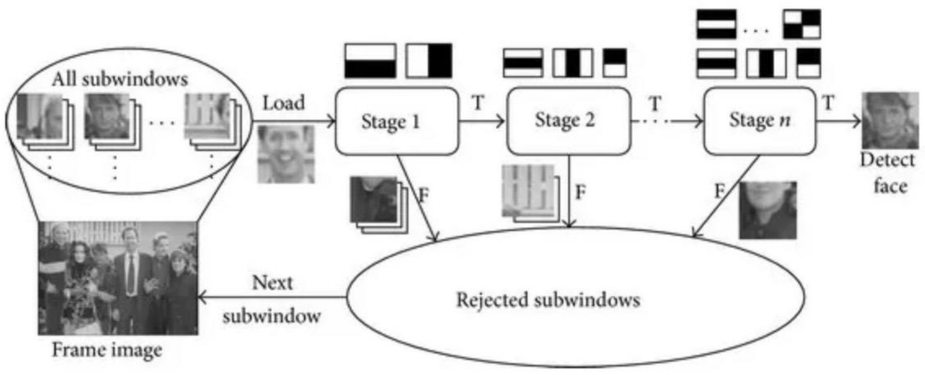


Fig. A sample of Haar features used in the Original Research Paper published by Viola and Jones.

Fig 1: Showing a sample of Haar features in the Research paper by Viola and Jones

In terms of mathematical foundations, the algorithm's Haar value calculation is crucial. This involves determining the difference in average pixel intensities between regions of interest. Integral Images, employed in the algorithm, significantly reduce the time complexity of these calculations. When considering its application to forecasting, defining the problem becomes crucial. This involves determining the Haar-like features relevant to the forecasted patterns and configuring the algorithm accordingly. Parameters such as window size, feature selection, and cascading stages should be tuned based on the specific characteristics of the forecasting task. In conclusion, the Haar Cascade algorithm, with its sophisticated approach to pattern detection, holds promise for forecasting applications, offering a unique blend of speed and precision.



Architecture

Fig 2: Showing Haar Cascade Architecture

2.2 Fully Connected Model

A fully connected layer, often called a dense layer, is the basic building block of artificial neural networks. This layer is usually located at the end of neural network architectures and is aptly called "fully connected". because of its holistic connectivity: every neuron in the previous layer connects to every neuron in the current layer. This connectivity forms a densely woven neural network, making fully connected layers essential for many applications such as image recognition and natural language processing. The nonlinear transformation equation for a fully connected layer is given by:

$$y_{jk}(x) = f\left(\sum_{i=1}^{n_H} w_{jk}x_i + w_{j0}\right)$$

One of the characteristic tasks of fully connected layers, especially in convolutional neural networks (CNNs), is to smooth spatial information. After the convolution and pooling layers, these layers transform the 2D spatial structure of the data into a 1D vector, setting the stage for later tasks such as classification. The importance of the layer lies in its ability to capture complex data patterns and relationships, making it a key element in applications beyond image recognition and natural language processing.

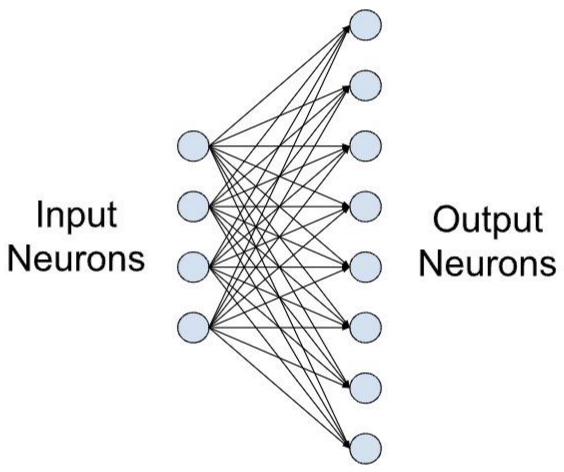


Fig 3: Example of a small fully connected layer with four input and eight output neurons.

An important feature of fully connected layers is the adaptability of their learning parameters. The weights and biases of these layers are set during the training process, allowing the neural network to adapt its parameters to the problem at hand. In addition, the last fully connected layer of a neural network is often ranked with the number of output classes in the classification problem. Each neuron in this layer provides a separate class score, facilitating the classification of input data into predefined classes.

Batch normalization is a valuable technique for improving the performance of fully connected layers. Batch normalization, applied to the layer output before the activation function, normalizes the activations during training. This normalization process, which involves calculating the mean and variance, promotes faster convergence, allows for higher learning rates, reduces sensitivity to weight initialization, and provides a regularization effect. In conclusion, a comprehensive understanding of fully connected layers and their integration with techniques such as group normalization lays the foundation for the construction of efficient neural networks capable of handling various computational tasks.

2.3 CNN Model

Convolutional Neural Networks (CNN) stand out as a special category of deep learning, strategically designed for tasks involving images. These networks develop hierarchically and include crucial elements such as convolutional layers, activation functions, connection layers and fully connected layers. Acting like intelligent feature detectors, convolutional layers use filters to detect unique patterns in images. The introduction of activation functions introduces non-linearity that allows the network to understand complex data relationships. At the same time, combining layers plays a key role in reducing spatial dimensions, which emphasizes valuable information. The final layer, which consists of fully connected layers, combines these learned features to make final predictions. Convolution layer formula:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

The robust performance of CNNs comes from their nuanced ability to decipher complex patterns through layered computations. This typical expertise can be optimally used in tasks such as image classification and object recognition. Architecture and its inherent strength are based on its ability to understand the spatial relationships inherent in the input data. Mathematically, CNNs use convolution functions, activation functions, and pooling mechanisms that synergistically contribute to the network's ability to handle image-related tasks. This comprehensive integration of components highlights the importance and efficiency of CNNs in visual information processing and analysis.

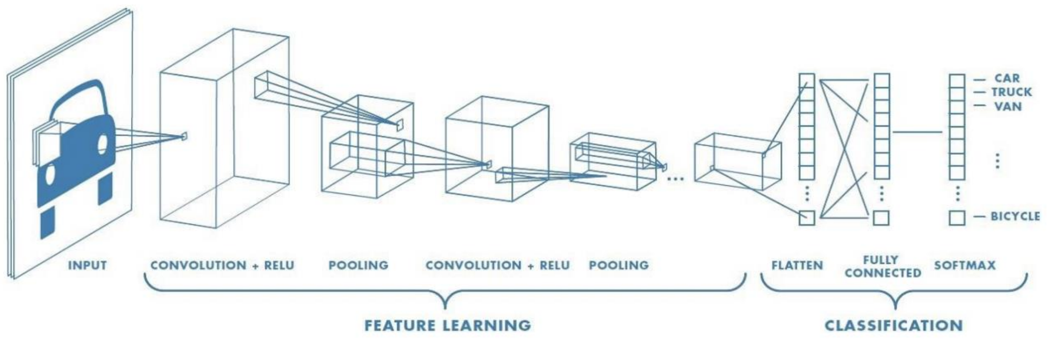


Fig 4: CNN Model working architecture

3. Methodology/process

3.1.1 Methodology 1: Haar Cascading

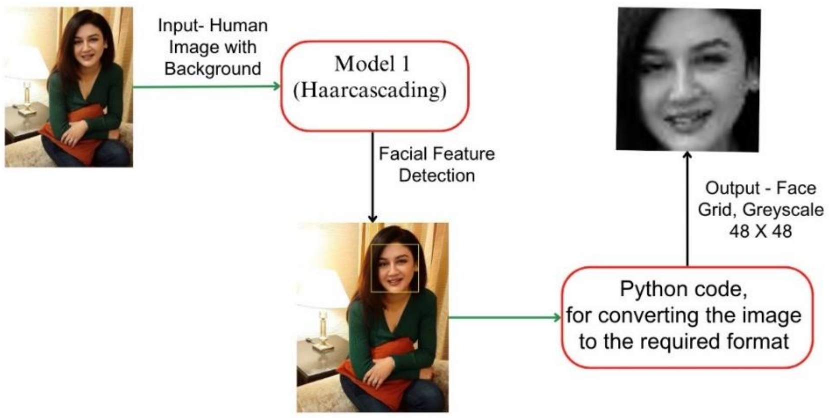

Input- Human Image with Background
Model 1 (Haarcascading)
Facial Feature Detection
Python code, for converting the image to the required format
Output - Face Grid, Greyscale 48 X 48

Fig 5: Showing the working of Model 1, using Haar Cascading

When emotion recognition systems are used in real-world scenarios, especially those that analyze emotions from real-time images captured by cameras, a structured image preprocessing workflow is crucial to optimize emotion classification accuracy. The images obtained by the camera are raw data that must be standardized to meet the requirements of the following advanced analysis models.

For this, a pre-trained Haar Cascading model is used in the pre-processing step, which acts as a reliable face recognition mechanism. The Haar Cascading model's effectiveness is based on its ability to accurately detect and localize facial regions in diverse and dynamic backgrounds. That model has been extensively trained and is adept at delineating boundaries around faces in an image, which is a crucial step in separating facial expression from external visual information.

Once the Haar Cascade classifier detects and labels the face region, custom Python scripts proceed to crop the image to the detected boundaries. This cropping is a targeted removal that ensures that subsequent analysis focuses only on facial expression. After cropping, the image will be converted to grayscale. This transformation removes color from an image and reduces it to various shades of gray, which emphasize intensity and contrast, which are critical for the expression and perception of emotion.

The grayscale image now represents a standardized input consistent with advanced models of emotion recognition—Model 2A, a fully connected network. and Model 2B, Convolutional

Neural Network (CNN). Both models are carefully trained on the FER2013 dataset, a face recognition benchmark consisting of 48x48 pixel grayscale images tagged with a particular emotion tag.

By adapting the images to a 48x48 grayscale format, the system ensures that the data entered by either Model 2A or Model 2B is exactly what the models are trained to interpret. This matching is critical for accurately classifying an image in one of the different emotional categories identified by the models. Without this careful pre-processing, the image may not be correctly evaluated, which may lead to misclassification of the expressed emotion.

In the test phase, raw images are subjected to face detection using a pre-trained Haar Cascade classifier, cropped and grayscale. The standardized 48x48 images are then carefully cataloged for use in emotion detection.

Processed images are stored in two formats for easy access and analysis. First, they are saved to a structured CSV file where each image is flattened into a single row of pixel values. This format is particularly useful for feeding data to machine learning models in a batch process, as it allows large data sets to be loaded quickly and processed efficiently. The CSV file can be easily integrated into various data processing and machine learning pipelines, enabling fast and automated analysis.

Secondly, the images are organized into folders, and each folder usually represents a specific emotional category. This directory-based storage provides an intuitive way to access and manage image data, which can be especially useful in applications that require fast data transfer, such as interactive user interfaces or real-time monitoring systems.

For scenarios where the face is partially obscured (e.g., by hands or objects), it is important to note that the current implementation of the Haar cascade model does not address these situations directly. By design, the system only captures visible faces and does not extend to processing masked or blurred facial features. This deliberate limitation is consistent with the original model's primary goal, which is to identify and extract visible expressions.

An illustrative scenario involved adding 40 images to the system. Among them were cases of a cat picture, a dog picture, and two pictures where the face was covered with hands. In particular, the Haar Cascading model did not capture facial features in images with masked faces, as expected from a deliberate design choice. The model effectively fulfills its main task of capturing and extracting visible expressions.

3.1.2 Methodology 2: FC (Model 2A)

3.1.2.1 Base Model Architecture and Features:

The basic FC model, built without the integration of facial landmarks, represented our first attempt at emotion analysis using the FER-2013 dataset.

- **Input Features:** The base model was run on FER-2013 grayscale images representing different facial expressions. Standard preprocessing steps such as resizing, and normalization were used to ensure consistent input.
- **Neural Network Structure:** The architecture of the FC model consisted of several fully connected layers that allowed the model to extract complex patterns and relationships from the input data. Activation functions such as ReLU introduced non-linearity, helping to capture complex emotional expressions.
- **Loss Function and Optimizer:** Categorical cross entropy loss was used to drive the model and learning process. The Adam optimizer facilitated efficient weight updates during training, contributing to the model's convergence.

Training and Performance:

During the training phase, parameters including dropout rate and the number of neurons in fully connected layers were iteratively adjusted. Performance evaluation metrics included precision, accuracy, recall, and F1 score. These measures helped evaluate the model and its ability to classify facial expressions into seven different emotions. Challenges identified at this stage included the sensitivity of the model to subtle emotional signals and the need to improve strategies to handle class imbalances in the dataset. The model could not extract complex features from the images.

Enhancements to Training Dynamics:

To improve the training dynamics, a learning decay strategy was introduced, resulting in a gradual decrease in the learning rate every 10 epochs. This adaptive adjustment was to strike a balance between rapid convergence in the initial stages and subtle adjustments as the exercise progressed. In addition, the training process included an early termination mechanism based on the loss of validation. This feature ensured that training was stopped if no significant improvement was observed after a certain time, preventing overfitting, and promoting the generalizability of the model.

3.1.2.2 FC Model with Landmarks Integration:

Landmark Addition and Model Enhancement:

To overcome the limitations observed in the original FC model, facial landmarks were added as additional features. This integration provided the model with more nuances of facial expressions, potentially improving its predictive power.

- **Landmark Extraction:** Facial landmarks obtained using the dlib library have been normalized and added to existing grayscale images as additional features.
- **Model Adaptation:** The FC model and its architecture have been modified for these additional attractions. The number of input neurons was adjusted to include orientation coordinates and distances.

Impact on Results:
The augmented FC model, now enriched with facial landmarks, underwent a reevaluation to measure the impact on its predictive performance.

- **Performance Enhancement:** The addition of facial landmarks led to significant improvements, especially in accurately capturing subtle facial cues associated with different emotions.
- **Addressing Imbalances:** The integrated model showed superior performance in correcting class imbalance and improving emotion classification in all seven emotion categories.

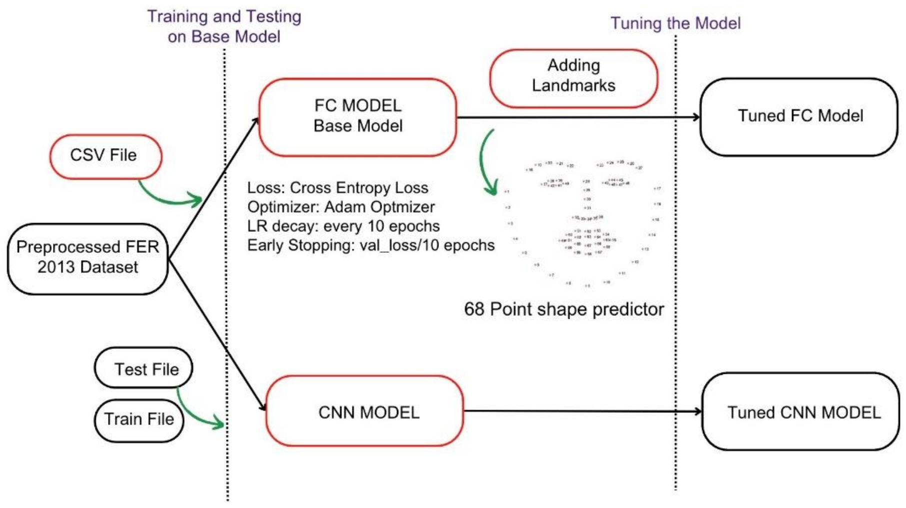
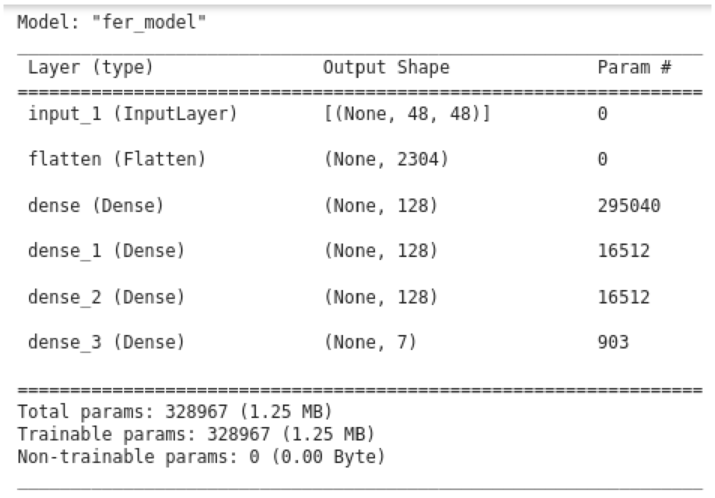


Fig 5: Depicting the working flowchart for Methodology 2

A comparative analysis of the landmark free FC model and the landmark enhanced model shows the importance of including detailed facial features to improve emotion prediction. Adding landmarks helps to understand facial expressions in a more comprehensive way, which improves the model's performance in real-world scenarios.

The reasons for including facial landmarks are explained in the results section, which explores the benefits and detailed improvements obtained by improving the FC model with additional orientation information. Adding landmarks is expected to provide additional information to the model to detect complex patterns and features in the image, which can improve accuracy.



```
Model: "fer_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 48, 48)]          0

 flatten (Flatten)           (None, 2304)              0

 dense (Dense)               (None, 128)               295040

 dense_1 (Dense)             (None, 128)               16512

 dense_2 (Dense)             (None, 128)               16512

 dense_3 (Dense)             (None, 7)                 903

=================================================================
Total params: 328967 (1.25 MB)
Trainable params: 328967 (1.25 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig 6: The summary model of FC

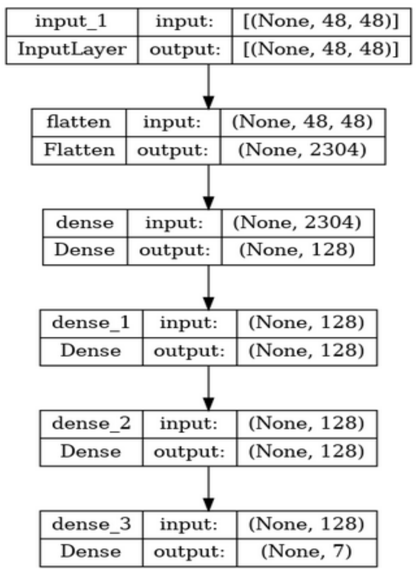

input_1 | input: | [(None, 48, 48)]
InputLayer | output: | [(None, 48, 48)]
flatten | input: | (None, 48, 48)
Flatten | output: | (None, 2304)
dense | input: | (None, 2304)
Dense | output: | (None, 128)
dense_1 | input: | (None, 128)
Dense | output: | (None, 128)
dense_2 | input: | (None, 128)
Dense | output: | (None, 128)
dense_3 | input: | (None, 128)
Dense | output: | (None, 7)

Fig 7: the model layer structure of FC

3.1.3 Methodology 3: CNN (Model 2B)

In this report, we create a sequential CNN model using TensorFlow and the Keras API. The advantage of the Keras API is that it allows for layer-by-layer model building, while TensorFlow is an end-to-end, open machine learning platform that provides a flexible set of tools, libraries, and community resources for building and deploying machine learning applications.

Our sentiment categorization task involves 7 different sentiment categories, each of which corresponds to a sentiment expression. To accomplish this, we define a function create_model, which is used to build a convolutional neural network model. The function accepts a few parameters such as dropout_rate, filters (number of convolutional kernels), kernel_size, and units (number of neurons in the fully connected layer).

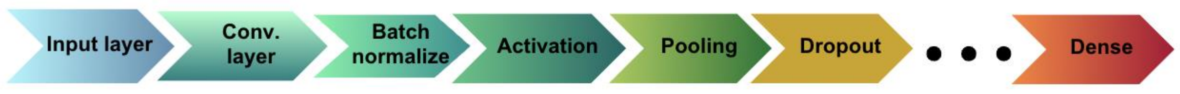
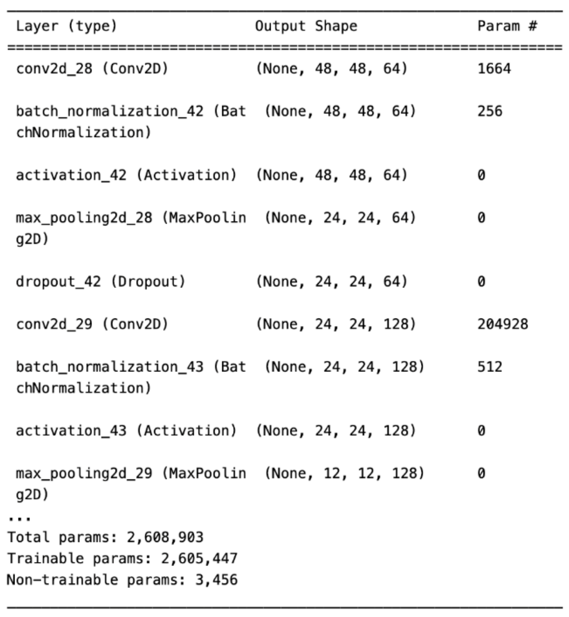


Fig 8: Structure of CNN

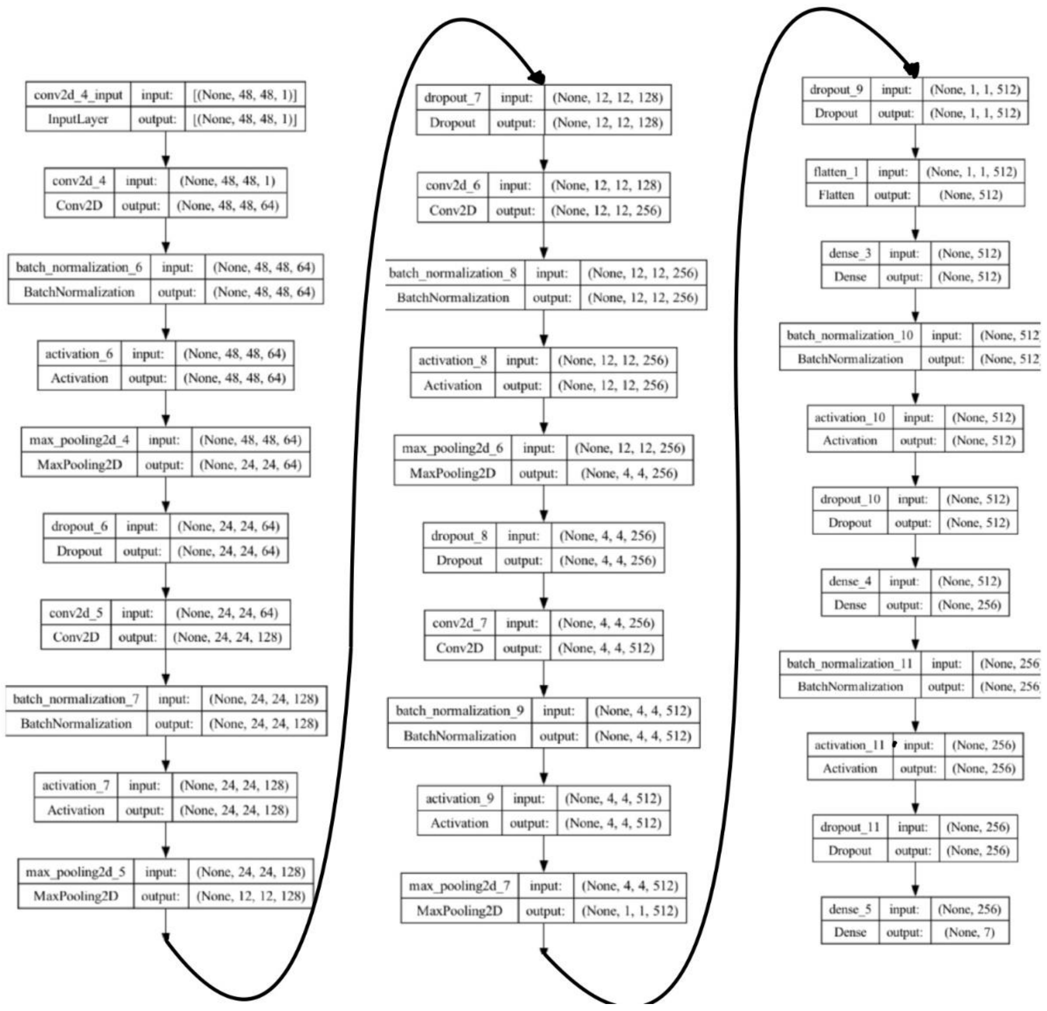
The structure of the model consists of four convolutional layers, each followed by a Batch Normalization layer, a ReLU activation function, and a maximum pooling layer. This helps to extract features from the image and reduce the dimensionality of the features. In addition, the model includes two fully connected layers, each followed by a Batch Normalization layer, ReLU activation function, and Dropout layer to introduce more nonlinearity and prevent overfitting. Finally, the output layer uses a SoftMax activation function for sentiment classification.



```
______________________________________________________________
 Layer (type)                Output Shape              Param #
==============================================================
 conv2d_28 (Conv2D)          (None, 48, 48, 64)        1664

 batch_normalization_42 (Bat  (None, 48, 48, 64)       256
 chNormalization)

 activation_42 (Activation)  (None, 48, 48, 64)        0

 max_pooling2d_28 (MaxPoolin  (None, 24, 24, 64)       0
 g2D)

 dropout_42 (Dropout)        (None, 24, 24, 64)        0

 conv2d_29 (Conv2D)          (None, 24, 24, 128)       204928

 batch_normalization_43 (Bat  (None, 24, 24, 128)      512
 chNormalization)

 activation_43 (Activation)  (None, 24, 24, 128)       0

 max_pooling2d_29 (MaxPoolin  (None, 12, 12, 128)      0
 g2D)
...
Total params: 2,608,903
Trainable params: 2,605,447
Non-trainable params: 3,456
______________________________________________________________
```

Fig 9: the summary model of CNN

The CNN model has five stages. In the first four stages, the size of the input image is reduced at the end of each stage. The first four stages have the same layer structure, starting with a convolution and ending with a flat layer. The combination of convolutional and pooling layers in each stage helps in extracting high-level features of the image step by step. The last stage consists of flat, dense, and output layers for sentiment classification tasks. SoftMax function is applied to the output layer to generate probabilities for each category. This is designed to capture relevant patterns and information from the raw pixel values to provide effective features for subsequent training.



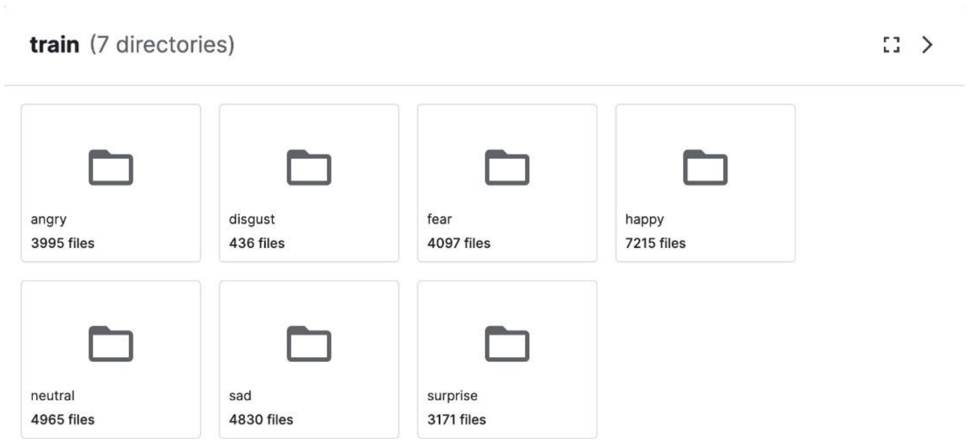
conv2d_4_input | input: | [(None, 48, 48, 1)]
InputLayer | output: | [(None, 48, 48, 1)]
conv2d_4 | input: | (None, 48, 48, 1)
Conv2D | output: | (None, 48, 48, 64)
batch_normalization_6 | input: | (None, 48, 48, 64)
BatchNormalization | output: | (None, 48, 48, 64)
activation_6 | input: | (None, 48, 48, 64)
Activation | output: | (None, 48, 48, 64)
max_pooling2d_4 | input: | (None, 48, 48, 64)
MaxPooling2D | output: | (None, 24, 24, 64)
dropout_6 | input: | (None, 24, 24, 64)
Dropout | output: | (None, 24, 24, 64)
conv2d_5 | input: | (None, 24, 24, 64)
Conv2D | output: | (None, 24, 24, 128)
batch_normalization_7 | input: | (None, 24, 24, 128)
BatchNormalization | output: | (None, 24, 24, 128)
activation_7 | input: | (None, 24, 24, 128)
Activation | output: | (None, 24, 24, 128)
max_pooling2d_5 | input: | (None, 24, 24, 128)
MaxPooling2D | output: | (None, 12, 12, 128)
dropout_7 | input: | (None, 12, 12, 128)
Dropout | output: | (None, 12, 12, 128)
conv2d_6 | input: | (None, 12, 12, 128)
Conv2D | output: | (None, 12, 12, 256)
batch_normalization_8 | input: | (None, 12, 12, 256)
BatchNormalization | output: | (None, 12, 12, 256)
activation_8 | input: | (None, 12, 12, 256)
Activation | output: | (None, 12, 12, 256)
max_pooling2d_6 | input: | (None, 12, 12, 256)
MaxPooling2D | output: | (None, 4, 4, 256)
dropout_8 | input: | (None, 4, 4, 256)
Dropout | output: | (None, 4, 4, 256)
conv2d_7 | input: | (None, 4, 4, 256)
Conv2D | output: | (None, 4, 4, 512)
batch_normalization_9 | input: | (None, 4, 4, 512)
BatchNormalization | output: | (None, 4, 4, 512)
activation_9 | input: | (None, 4, 4, 512)
Activation | output: | (None, 4, 4, 512)
max_pooling2d_7 | input: | (None, 4, 4, 512)
MaxPooling2D | output: | (None, 1, 1, 512)
dropout_9 | input: | (None, 1, 1, 512)
Dropout | output: | (None, 1, 1, 512)
flatten_1 | input: | (None, 1, 1, 512)
Flatten | output: | (None, 512)
dense_3 | input: | (None, 512)
Dense | output: | (None, 512)
batch_normalization_10 | input: | (None, 512
BatchNormalization | output: | (None, 512
activation_10 | input: | (None, 512)
Activation | output: | (None, 512)
dropout_10 | input: | (None, 512)
Dropout | output: | (None, 512)
dense_4 | input: | (None, 512)
Dense | output: | (None, 256)
batch_normalization_11 | input: | (None, 256
BatchNormalization | output: | (None, 256
activation_11 | input: | (None, 256)
Activation | output: | (None, 256)
dropout_11 | input: | (None, 256)
Dropout | output: | (None, 256)
dense_5 | input: | (None, 256)
Dense | output: | (None, 7)

Fig 10: the model layer structure of CNN

3.2 Data preprocessing

3.2.1 Addressing Class Imbalance:

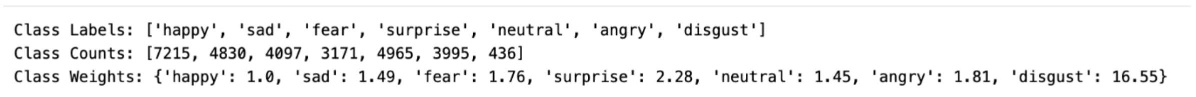
From the training set of our data in the figure below, it can be seen that there is a significant category imbalance in our facial expression recognition dataset, particularly with respect to 'happy'（7215 instances）and 'disgust'（436 instances）. This imbalance has the potential to affect the model's learning process, making categories with more samples easier to learn, while categories with fewer samples may be hindered.


train (7 directories)

angry 3995 files	disgust 436 files	fear 4097 files	happy 7215 files
neutral 4965 files	sad 4830 files	surprise 3171 files	

Fig 11: train set data

Therefore, we performed data preprocessing. First, by traversing the subfolders in the training set folder （each subfolder represents a category）, we obtained the number of samples for each category and stored the category labels with the corresponding number of samples in the ***class_labels*** and ***class_counts*** lists. Next, by taking the ratio of the maximum number of samples to the number of samples per category, the categories with fewer samples have larger weights. These weights were saved in the class_weights dictionary. Eventually, we introduced these class weights into the training process of the model:



```
Class Labels: ['happy', 'sad', 'fear', 'surprise', 'neutral', 'angry', 'disgust']
Class Counts: [7215, 4830, 4097, 3171, 4965, 3995, 436]
Class Weights: {'happy': 1.0, 'sad': 1.49, 'fear': 1.76, 'surprise': 2.28, 'neutral': 1.45, 'angry': 1.81, 'disgust': 16.55}
```

Fig 12: Showing class labels and its contents

Category imbalance may cause the model to be more biased towards learning frequently occurring categories while ignoring rare categories. By introducing category weights, we can achieve balancing the importance of various categories, ensuring that the model pays sufficient attention to each category during training, while improving the model's ability to generalize over unseen and infrequently occurring categories. This makes it more likely that the model will achieve excellent performance for a wide range of situations in real-world applications.

In addition, the introduction of category weights helps to avoid the possibility that when a category has many samples, it may dominate the gradient update due to the computation of the loss function, causing the model to focus more on that category. The introduction of category weights helps to mitigate this effect by making the gradient update more balanced and improving the model's learning over all categories. At the same time, the category weights make the model more sensitive to categories that have fewer samples but may be more important.

3.2.2 Image Normalization:

To make the model independent of the scale of the input pixels, we performed normalization on the image to make the pixel values from the range of [0,255] to [0,1]. This helps the model to converge faster by preventing unnecessary large pixels.

3.2.3 Efficient Data Transformation:

Efficient data conversion: we employ an efficient data conversion strategy to convert the raw data into a more compact format. The strategy consists of preprocessing each pixel value by converting strings to arrays, reshaping the arrays to (48, 48), applying CLAHE (Contrast Constrained Adaptive Histogram Equalization), and obtaining facial marker points. Through this process, we were able to obtain the processed images and organize them into NumPy arrays with the corresponding labels, while excluding images with length 0.

Subsequently, we save the organized training and test set data and related information as an .h5 file. This file contains the list of categories ***list_classes***, the training set images ***train_set_x*** and labels ***train_set_y***, and the test set images ***test_set_x*** and labels ***test_set_y***. The HDF5 format has efficient read and write performance, especially when dealing with substantial amounts of data. The format allows data to be read on demand without having to load the entire dataset.

This choice optimizes data storage for large datasets and improves data querying efficiency while ensuring consistency and repeatability throughout the process. With this approach, we improved the overall efficiency of the project and maintained a user-friendly workflow.

3.3 Feature generation

The experiment involved processing 28,709 training images and 7,178 test images, each belonging to one of seven distinct categories. These seven categories have been pre-categorized in the original dataset, and the feature generation process went through a series of steps including image normalization, facial landmark extraction, and other preprocessing techniques. With this comprehensive feature generation approach, the aim is to capture relevant patterns and information from the raw pixel values to train an effective emotion classification model. The subsequently generated features are used as inputs to a machine learning model that helps in learning discriminative patterns of different emotional expressions in the dataset. The seven categories are anger, disgust, fear, happy, sad, surprise, and neutral ：

```
Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
```

```
Class Labels: ['happy', 'sad', 'fear', 'surprise', 'neutral', 'angry', 'disgust']
Class Counts: [7215, 4830, 4097, 3171, 4965, 3995, 436]
```

Fig 13: Depicting contents of the classes

As shown below, the first 9 images are traversed in each emotion labeled category. Select the image data for the corresponding sentiment label from the DataFrame, convert the string to a NumPy array, and resize the shape to 48x48. Traverse each sentiment label category by looping through it and visualize the top 9 images in it to help understand the image content of the dataset and the differences between the different sentiment labels.

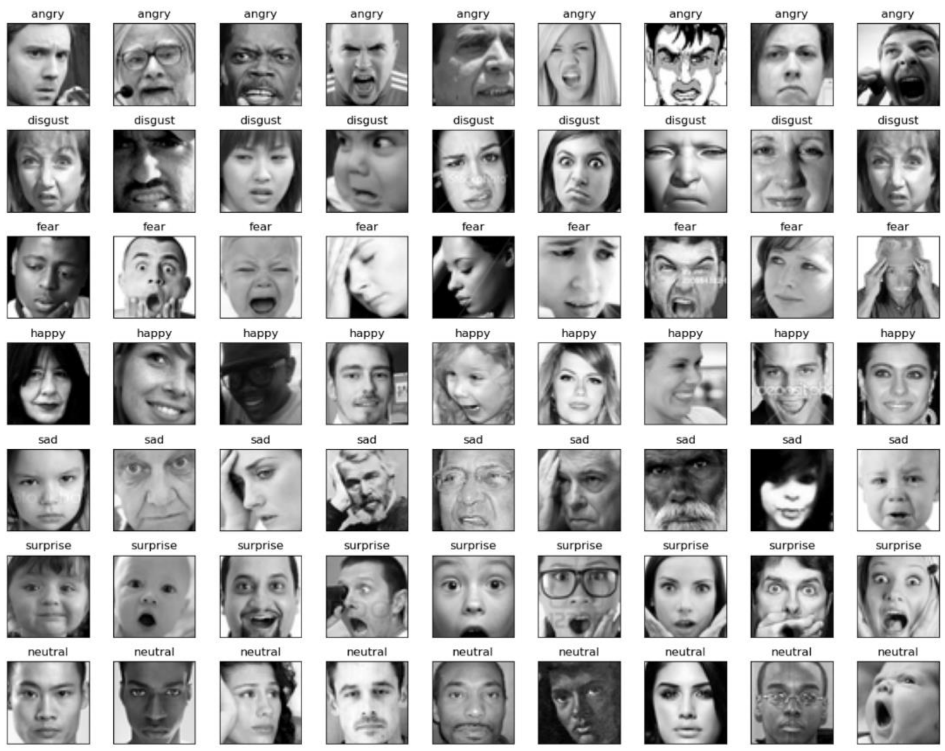


Fig 14: Partial picture display

3.4 Model Evaluation

Once training has begun, the behavior of the model can be evaluated by looking at its performance. Once training starts, the behavior of the models can be evaluated by looking at their performance. We trained the Fully Connected Neural Network （FC） and Convolutional Neural Network （CNN） models by calling the `*fit()*` function and `*fit_generator*` function, respectively. The confusion matrix is computed by using the `c*onfusion_matrix*` function and the models are evaluated and compared with the parameter values of the training results.

For the sentiment classification problem, we focus on the following metrics to measure the model performance:

Accuracy: The accuracy is given by

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Loss: Categorical cross-entropy is used as the loss function and is given by:

$$\text{Loss} = -\sum_{c=1}^{m}\left(y_{o,c}\log(p_{o,c})\right)$$

where y is a binary indicator (0 or 1), p is the predicted probability and m is the number of classes (happy, sad, neutral, fear, angry, disgust, surprise)

Confusion Matrix: The confusion matrix provides values for four combinations of true and predicted values: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). Precision, recall and F-score are calculated using TP, FP, TN, FN. TP is the correct prediction of an emotion, FP is the incorrect prediction of an emotion, TN is the correct prediction of an incorrect emotion and FN is the incorrect prediction of an incorrect emotion.

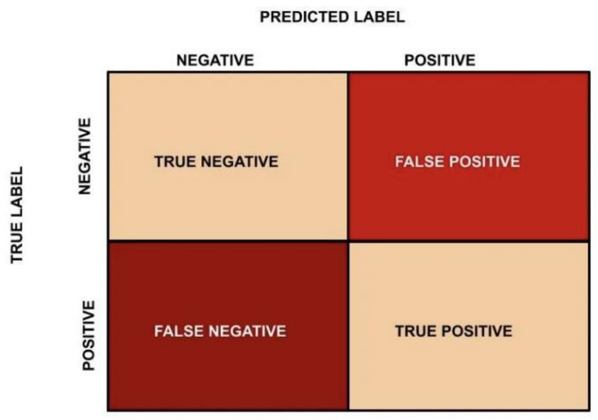


Fig 15: confusion matrix

Consider an image from the happy class. The confusion matrix for this example is shown as follows. The red section has the TP value as the happy image is predicted to be happy. The blue section has FP values as the image is predicted to be sad, angry, neutral, fear, disgust, or surprise. The yellow section has TN values as the image is not sad, angry, neutral, fear, disgust, or surprise but the model predicted this. The green section has FN values as the image is not happy but was predicted to be happy.

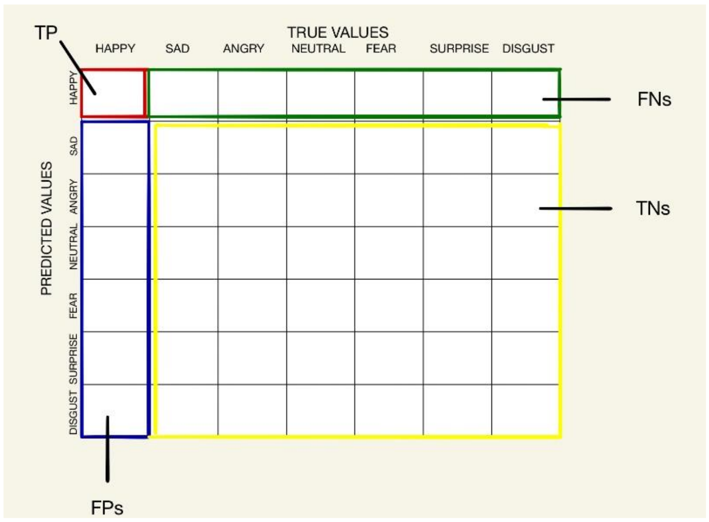


Fig 16: Confusion matrix of 7 emotions

Recall:

High recall is crucial for capturing as many positive instances as possible. In the context of emotion recognition, this means recognizing and classifying all occurrences of specific emotions accurately. High recall ensures that the model does not miss important facial expressions, contributing to a more comprehensive understanding of emotions.

Ideal Value: 1.0 (Captures all positive instances, no false negatives).

Recall is given by

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$$

Fig 18: Evaluation report metric: Recall

Precision:

Precision is a critical metric for the FER models as it emphasizes the accuracy of positive predictions. In the context of emotion recognition, precision ensures that predicted emotions are correct, minimizing the risk of misinterpreting facial expressions. Achieving high precision is vital to avoid false positives, which could lead to inaccurate emotional interpretations.

Ideal Value: 1.0 (All positive predictions are correct, no false positives).

Precision is given

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$$

Fig 19: Evaluation report metric: Precision

F1-score:

The F1 score provides a balanced assessment of precision and recall, offering a unified metric for a holistic evaluation. It is particularly relevant when there is a need to strike a balance between accurate positive predictions and capturing all positive instances. In emotion recognition, achieving a high F1 score ensures that the model performs well across both precision and recall aspects.

Ideal Value: 1.0 (Perfect balance between precision and recall).

F-score is the harmonic mean of recall and precision and is given by

$$\text{F-score} = \frac{2\times\text{Recall}\times\text{Precision}}{\text{Recall+Precision}}$$

Fig 20: Evaluation report metric: F1-Score

Support:

Understanding the distribution of facial expressions is crucial for contextualizing performance metrics. High support values indicate a comprehensive evaluation of model performance across different emotions, providing insights into the model's ability to generalize.

Ideal value: Higher support values for a comprehensive evaluation of model performance.

In this experiment we still use accuracy to evaluate the FC and CNN model. The accuracy ranges from 0 to 1 and reflects the model's classification performance on test data. When the accuracy is 1, it means that the model has completely and correctly classified all samples on the test set, that is, each sample is accurately predicted to its true category. Although this is an ideal situation, it is often rare in practical applications due to the frequent presence of noise and uncertainty in the data set. When the accuracy is 0, it means that the model did not correctly classify any sample on the test set, that is, every sample was predicted incorrectly. This is an extreme situation and usually indicates a significant problem with model performance, which may be due to model selection, data quality, or overfitting. We want the accuracy to fall between 0 and 1, which indicates that the model partially correctly classified the samples on the test set, i.e., the model's predictions are accurate for some samples but may be wrong for others.

4. Evaluation/Results

4.1 Model 1 results

Model 1 processed 40 raw input images with diverse human emotions having some background (noise) where we used the Haar Cascade algorithm, the objective being to detect human faces. The detected faces are then saved in a 48 x 48 grid in grayscale format. The overall outcomes are excellent, benefiting from the effectiveness of the pre-trained Haar Cascade algorithm in face detection. These facial features in a standardized grid format are intended to be input for the next model, that is model 2. On calculation of the output that received we determined that the accuracy of this algorithm is around 95% in our case since it was correctly able to detect facial features for 38 images out of the 40 input images.

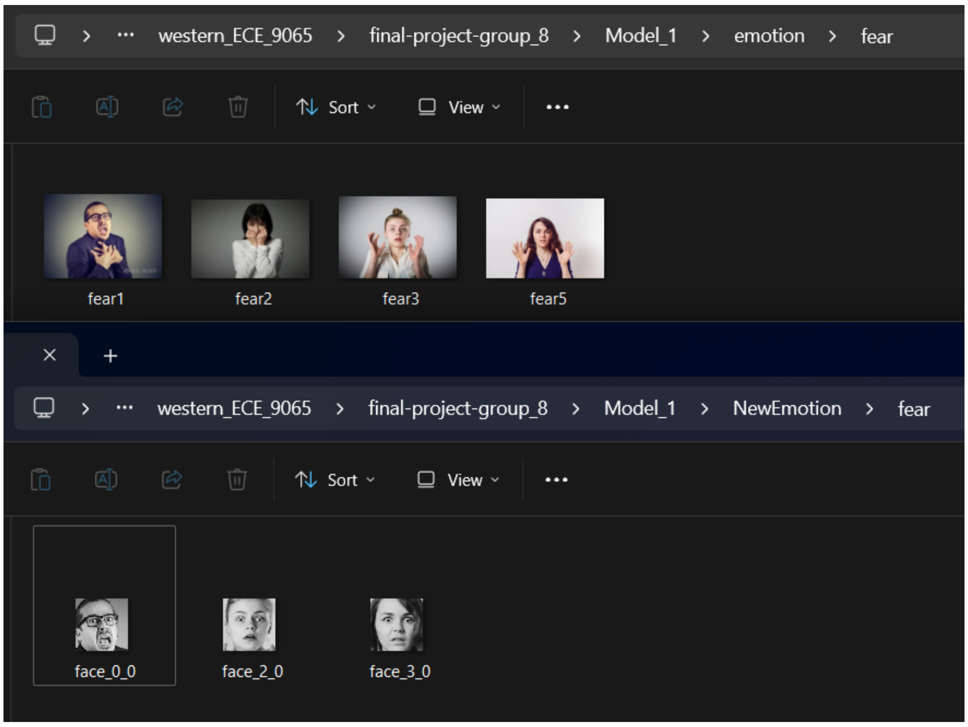

western_ECE_9065 > final-project-group_8 > Model_1 > emotion > fear
Sort
View
fear1
fear2
fear3
fear5
western_ECE_9065 > final-project-group_8 > Model_1 > NewEmotion > fear
Sort
View
face_0_0
face_2_0
face_3_0

Fig 21: Image Showing input and output results for a specific class

In the above image it is shown that for ‘fear’ class, the model does not capture the image where the human face is covered with both hands. Since these cases should not be captured in real world applications which might result in identity crisis, we can confirm that model 1 results are excellent and in accordance with our problem at hand. Just with our given situation that a face should have been detected irrespectively is what has made us determine its accuracy to be 95%.

4.2 Model 2 Results

This section compares the results of the two model architectures discussed in the previous section, namely, Fully Connected (FC) model and CNN model.
For the FC model, the results from 3 iterative models designed are shared below which includes the base model architecture of FC model, the FC model with Landmarks and finally the tuned version of the FC model with landmarks.

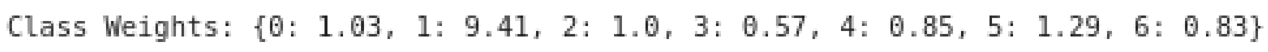
4.2.1 Fully Connected Model (FC Model)

For the Model 2, a fully connected layer based was chosen because of its tendency to capture complex patterns within input data.

4.2.1.1 FC Model without landmarks

The basic fully connected layer-based model (FC Model) consisted of 3 fully connected layers each having 128 neurons and activation function as ‘Relu’ which being a non-linear function can capture complex relationships between images and their emotions.

Class Weights:
To compute the class weights, “compute_class_weight” function was used from the scikit learn library.
Since the dataset used to solve this multi-class classification problem is highly unbalanced, a “balanced” approach was used to assign the class weights which assigns a higher-class weight to the class having lower frequency of items and vice-versa for class having a higher number of images.



```
Class Weights: {0: 1.03, 1: 9.41, 2: 1.0, 3: 0.57, 4: 0.85, 5: 1.29, 6: 0.83}
```

Fig 22: Figure showing the class weights for the model training

The model was trained using a step-decaying learning rate of 0.0001 using the below formula:

$$lr = initial\_lr * math.pow(drop, math.floor((1 + epoch) / epochs\_drop))$$

The learning rate was reduced every 10 epoch cycles.
The validation loss was also monitored during the training process and the best model was saved if the validation loss did not change after 10 epochs.

Finally, with a batch size of 64 and number of epochs of 100, the training of the basic FC model was started using the train_labels from the dataset.
model to generalize well.

4.2.1.1.1 Accuracies and Loss

As observed in the below plot, the model stopped training at the #26 epoch since the validation loss did not change, confirming that early stopping is helping the model to converge quicker and prevent overfitting.
Another point to note is that the validation loss always remains below the training loss. This is because of the class_weights added in the data preprocessing step. Due to this, the training of the model was done on a “balanced” set. The validation set consisted of images from the category which was giving better results. Hence, the addition of class_weights helped to improve the generalization and accuracy of the model.
It can also be observed, the model could converge earlier after using the class weights as shown from the below plots:

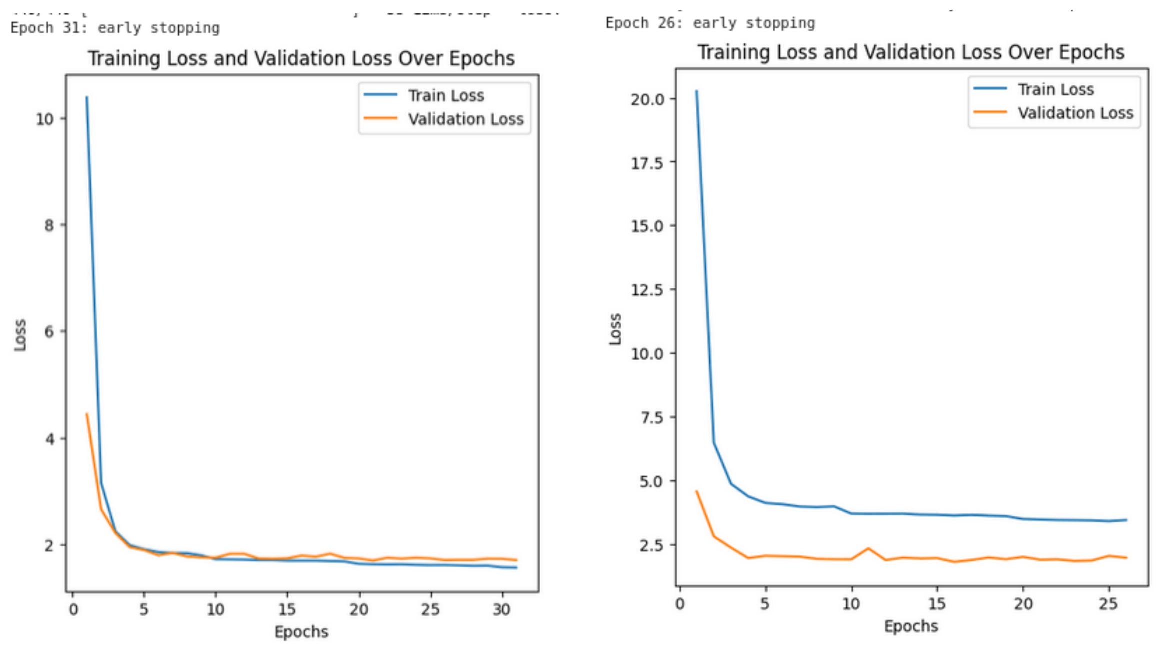

```
Epoch 31: early stopping
```
```
Epoch 26: early stopping
```

Fig 23: Graph showing Training Loss and Validation Loss before and after using class weights for FC Base Model

From the plot below, both the training and validation accuracy curve are increasing, which points us in the direction the model is learning.

Both the training and validation accuracy seems to be around 30%, which tells us there is room for improvement in the model and training techniques. At some instances, the gap between training and validation accuracy is not small, so model is not generalizing well.

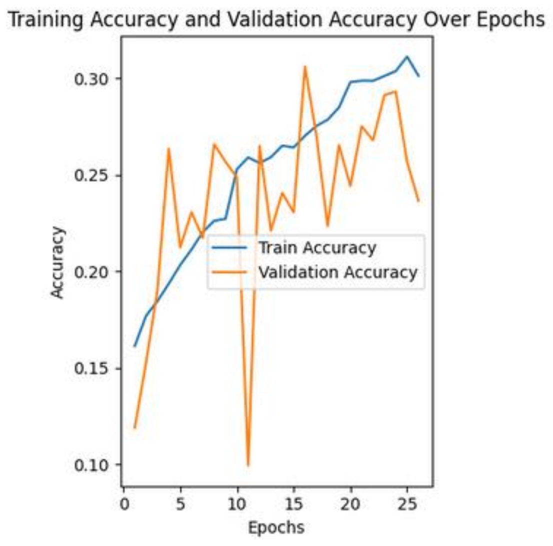
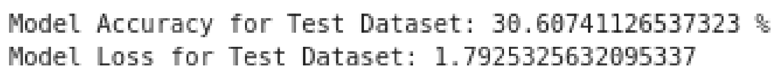


Fig 24: Graph plot between the Training and Validation Accuracy over Epochs for FC Base

Model

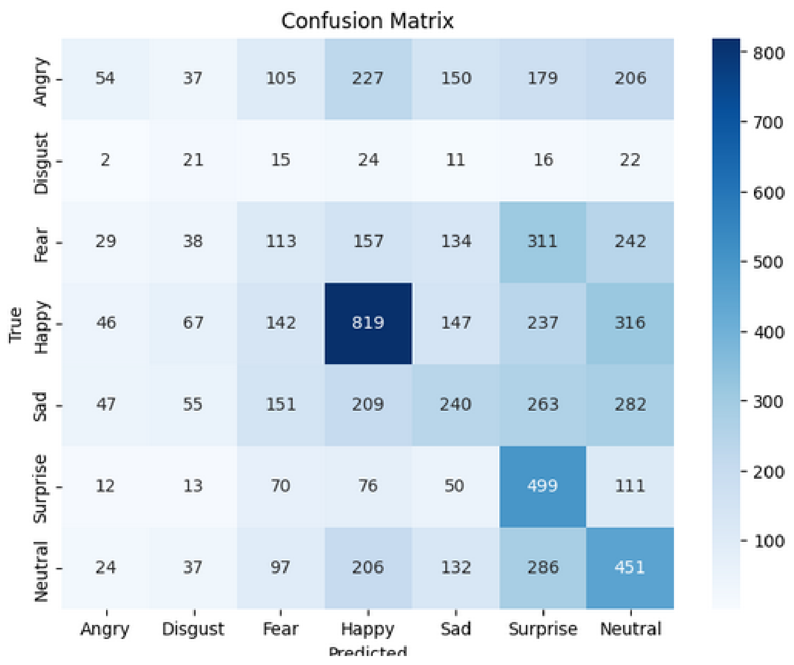


```
Model Accuracy for Test Dataset: 30.60741126537323 %
Model Loss for Test Dataset: 1.7925325632095337
```

Fig 25: Base FC Model results for Accuracy and Loss for the FC Base Model

4.2.1.1.2 Confusion Matrix and Evaluation Report

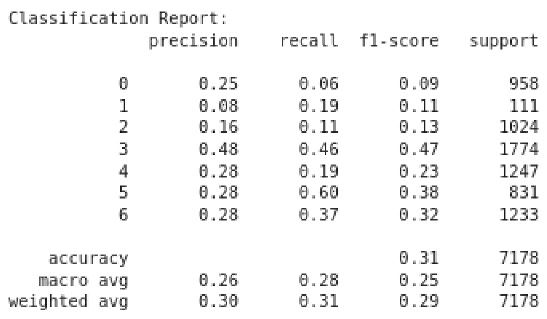
The "happy" category in the confusion matrix shown in the following image, along with the corresponding Precision, Recall, and F1-score data, show that the model has the highest prediction accuracy in this category.



True \ Predicted	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	54	37	105	227	150	179	206
Disgust	2	21	15	24	11	16	22
Fear	29	38	113	157	134	311	242
Happy	46	67	142	819	147	237	316
Sad	47	55	151	209	240	263	282
Surprise	12	13	70	76	50	499	111
Neutral	24	37	97	206	132	286	451

Fig 26: Confusion matrix for the base FC model

With a Precision of 0.48, 48% of the data that the model classified as "happy" genuinely fall into that category. The basic FC model could predict “Sad,” “Surprise” and “Neutral” equally well with a precision of 28% each. The percentage of successful examples the model captured for each category is measured by Recall. Recall for the "surprise" category is 0.60, meaning that 60% of the samples that truly fall into the category were successfully captured by the model.



```
Classification Report:
              precision    recall  f1-score   support

           0       0.25      0.06      0.09       958
           1       0.08      0.19      0.11       111
           2       0.16      0.11      0.13      1024
           3       0.48      0.46      0.47      1774
           4       0.28      0.19      0.23      1247
           5       0.28      0.60      0.38       831
           6       0.28      0.37      0.32      1233

    accuracy                           0.31      7178
   macro avg       0.26      0.28      0.25      7178
weighted avg       0.30      0.31      0.29      7178
```

Fig 27: Classification Report showing evaluation metrics – Precision, Recall, F1-Score and Support for FC Base Model

emotions = [0:"Angry", 1:"Disgust", 2:"Fear", 3:"Happy", 4:"Sad", 5:"Surprise", 6:"Neutral"]

F1-score talks about an imbalance in class distribution. The dataset is highly unbalanced, confirmed by a support value of 1774 for the "happy" class and just "111" for the disgust class.

Below is a visual representation of the above discussed results in a grid plot.

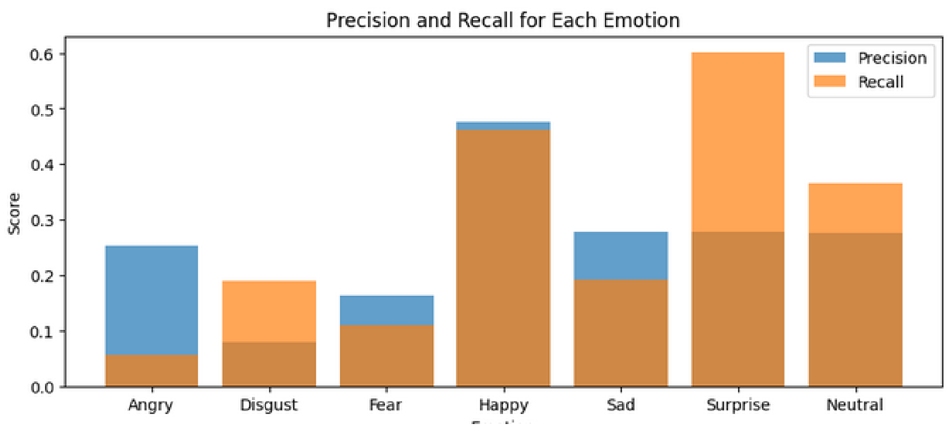


Fig 28: Representation of Precision and Recall for each Emotion for FC Base Model

4.2.1.2 FC Model with landmarks (before tuning)

To increase the accuracy of the model, landmarks were added to the image before the training process.

4.2.1.2.1 Accuracies and Loss

As seen from the plot below, the loss reduced by 10% as compared to loss without landmarks.

In this plot as well, the validation loss is well below the training loss as the class_weights have been used which makes the model training process balanced.

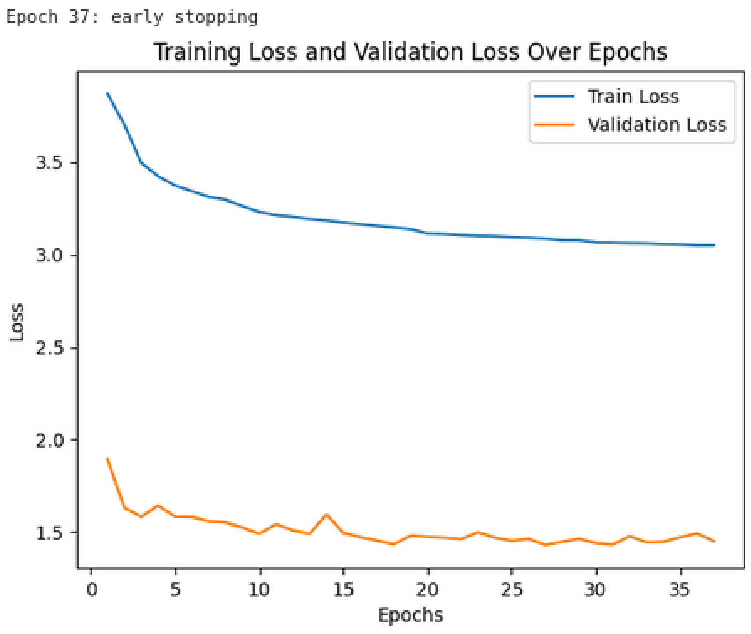


Fig 29: Graph showing Training Loss and Validation Loss over epochs for FC Model with Landmarks

The training accuracy vs the validation accuracy plot provides better results as compared to the model without landmarks. There is a small gap between the training accuracy and the validation accuracy. We can conclude from this that the updated model with landmarks learns and generalized well.

Due to early stopping, the validation accuracy does not decrease as the model converges. Hence, we can also say the there is a prevention of overfitting happening.

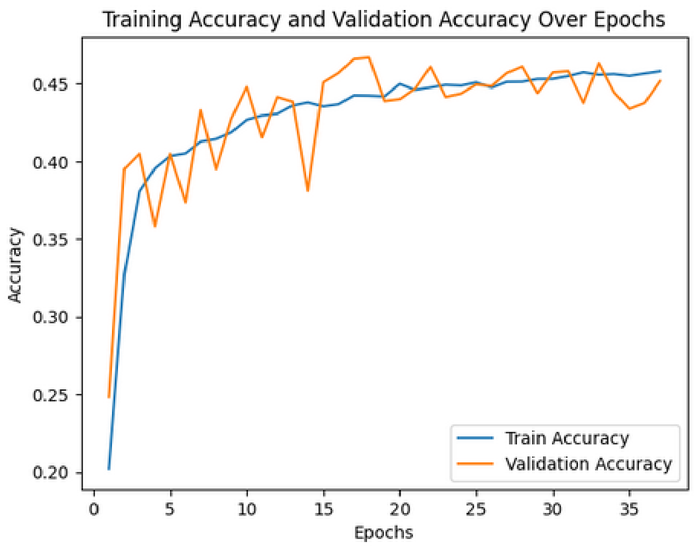
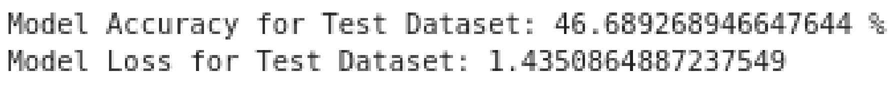


Fig 30: Graph plot between the Training and Validation Accuracy over Epochs for FC Model with Landmarks

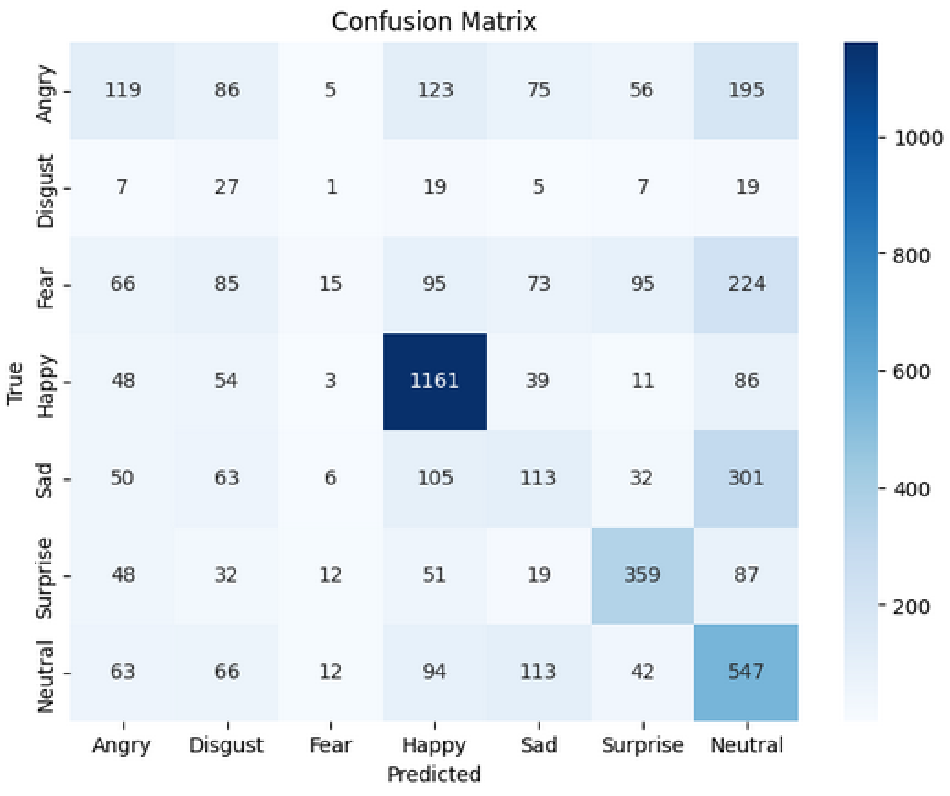


```
Model Accuracy for Test Dataset: 46.689268946647644 %
Model Loss for Test Dataset: 1.4350864887237549
```

Fig 31: Base FC Model results for Accuracy and Loss for FC Model with Landmarks

4.2.1.2.2 Confusion Matrix and Evaluation Report

From the below confusion matrix, the true positive in the “Happy” category increase by 46% and it double for “Angry” category demonstrating the landmarks are helping in detecting the True Positives.

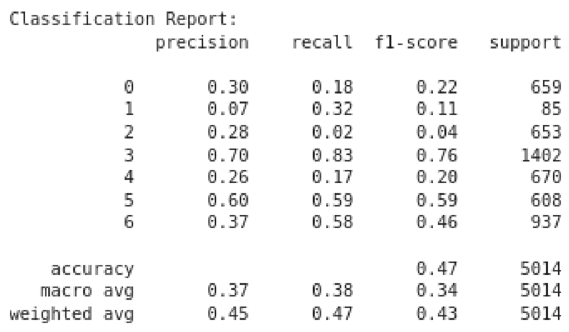


True \ Predicted	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	119	86	5	123	75	56	195
Disgust	7	27	1	19	5	7	19
Fear	66	85	15	95	73	95	224
Happy	48	54	3	1161	39	11	86
Sad	50	63	6	105	113	32	301
Surprise	48	32	12	51	19	359	87
Neutral	63	66	12	94	113	42	547

Fig 32: Confusion matrix for FC Model with Landmarks

The evaluation report shows that the precision or the ability to make positive predictions improved for all emotions category except “disgust” and “sad” where it missed by less than 10%. For the “happy” category, it rose by 45% to provide 70% precision.

Similar scale of improvements was seen for recall and f1-score.



```
Classification Report:
              precision    recall  f1-score   support

           0       0.30      0.18      0.22       659
           1       0.07      0.32      0.11        85
           2       0.28      0.02      0.04       653
           3       0.70      0.83      0.76      1402
           4       0.26      0.17      0.20       670
           5       0.60      0.59      0.59       608
           6       0.37      0.58      0.46       937

    accuracy                           0.47      5014
   macro avg       0.37      0.38      0.34      5014
weighted avg       0.45      0.47      0.43      5014
```

Fig 33: Classification Report showing evaluation metrics – Precision, Recall, F1-Score and Support for FC Model with Landmarks

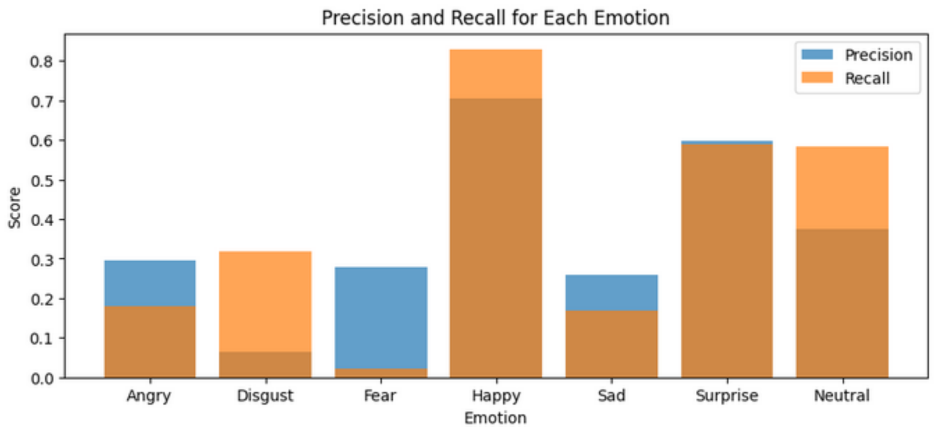
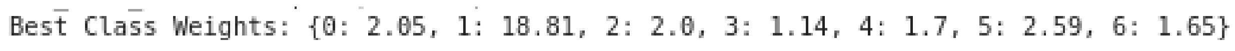


Fig 34: Representation of Precision and Recall for each Emotion for FC Model with Landmarks

4.2.1.3 FC Model with landmarks (after hyper-parameter tuning)

4.2.1.3.1 Tuning the Class Weights

The class weights computed in the initial step were scaled with a factor of [0.1, 0.5, 1.0, 2.0, 5.0, 10.0] to find the scaling factor that gives the best accuracy. In each of the scaling factors, "StratifiedKFold "cross validation technique was used from the sklearn.model_selection library to maintain a similar distribution of classes while training and generalizing the model. A scaling factor of "2.0" was found to have the best accuracy, hence, was used for the subsequent training and fine tuning.

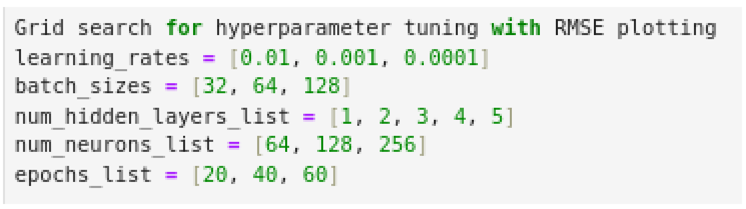


```
Best Class Weights: {0: 2.05, 1: 18.81, 2: 2.0, 3: 1.14, 4: 1.7, 5: 2.59, 6: 1.65}
```

Fig 35: Tuned class weights for the tuned FC Model with Landmarks

4.2.1.3.2 Tuning the Hyper-Parameters:

Based on the below grid search parameters, the model was tuned to find the best hyper parameters:



```
Grid search for hyperparameter tuning with RMSE plotting
learning_rates = [0.01, 0.001, 0.0001]
batch_sizes = [32, 64, 128]
num_hidden_layers_list = [1, 2, 3, 4, 5]
num_neurons_list = [64, 128, 256]
epochs_list = [20, 40, 60]
```

Fig 36: Hyper-parameter tuning using Grid Search

After performing the grid search, the following best parameters were obtained:

Training with learning rate: 0.0001, batch size: 32, hidden layers: 5, neurons: 256, epochs: 60

4.2.1.3.3 Accuracies and Loss

It can be observed that after hyper-parameter tuning, the Validation loss and Training loss tend to merge.

The validation loss maintains the same trend of having lower results than the training loss since in each of the training processes, balanced class weights were used to overcome the class imbalances.

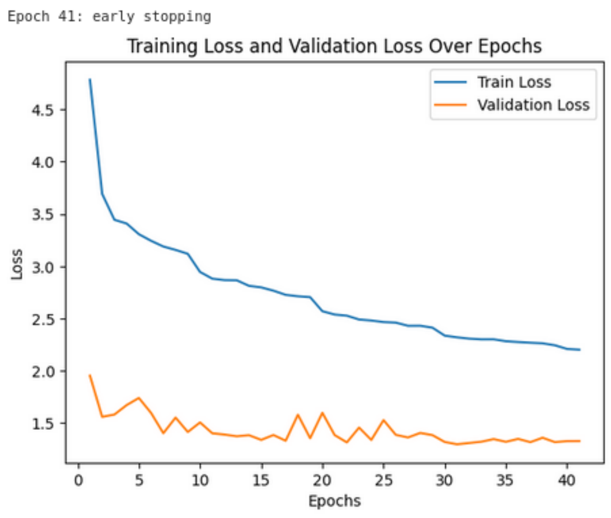

```
Epoch 41: early stopping
```

Fig 37: Graph showing Training Loss and Validation Loss over epochs for tuned FC Model with Landmarks

The model took slightly longer to stop the training process but resulted in a higher accuracy supporting the claim that hyper-parameter tuning is needed to get a better model that gives a higher accuracy and generalizes well. The validation loss seems to decrease after around 30 epochs, which suggests that the model may be over-fitting and total number of epochs of ~32 could be taken for better results.

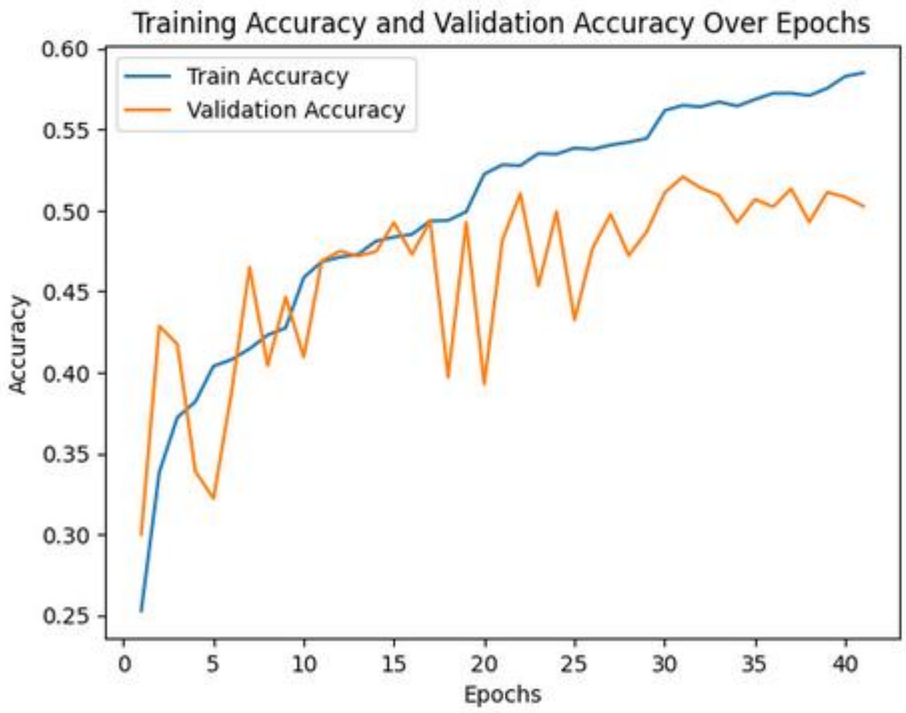


Fig 38: Graph plot between the Training and Validation Accuracy over Epochs for tuned FC Model with Landmarks

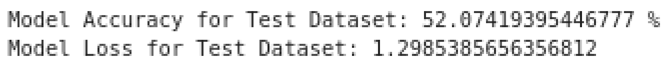

```
Model Accuracy for Test Dataset: 52.07419395446777 %
Model Loss for Test Dataset: 1.2985385656356812
```

Fig 39: Results for Accuracy and Loss for tuned FC Model with Landmarks

4.2.1.3.3 Confusion Matrix and Evaluation Report

Below is the confusion matrix for the FC model with hyper-parameters tuned.

“Sad,” “Fear” and “Disgust” categories should have significant improvements in predicting True Positives as can be seen on the main diagonal of the confusion matrix.

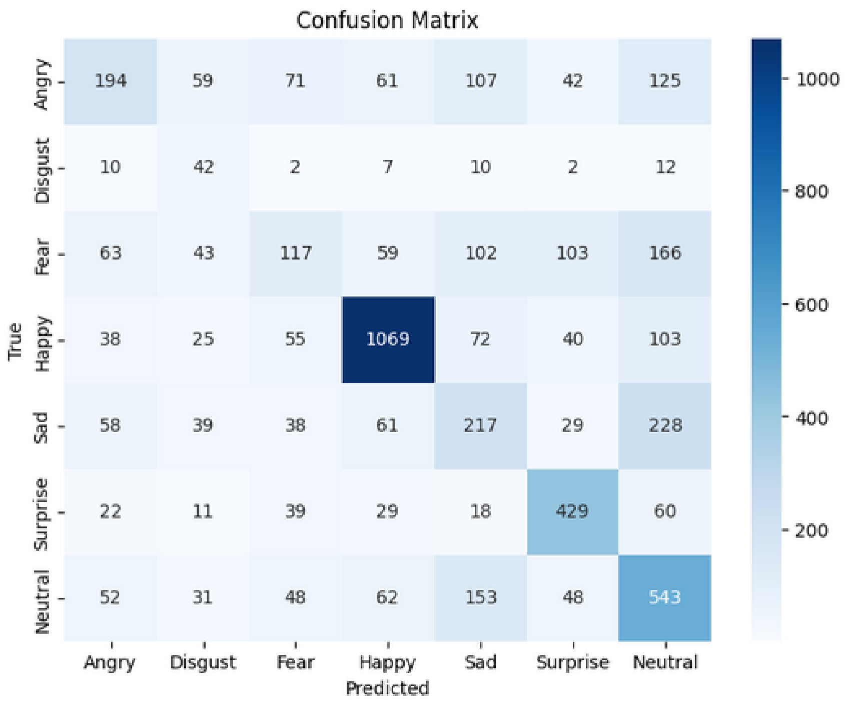


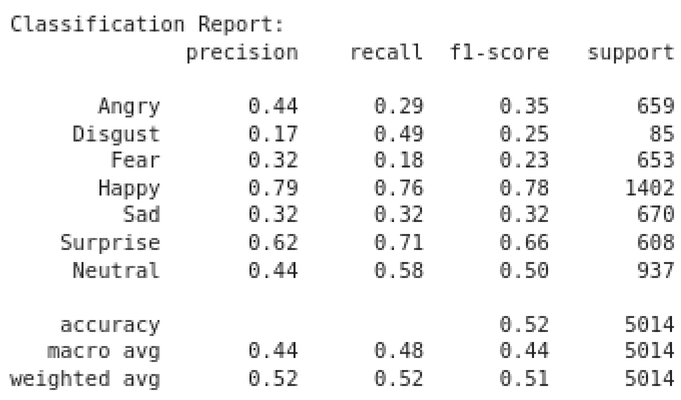
Fig 40: Confusion matrix for tuned FC Model with Landmarks

The tuned model is showing significant improvements in precision, recall and f1-score.

For instance, precision for "disgust" is increased to 17% from 7%. "Angry" category also having better accuracy of positive predictions, improved from 30% to 44% for the tuned model.

The ability to capture all the relevant instances of the positive class rose from just 2% for the "fear" category to 18%, thus, is an indicator that false negatives are minimizing.

"Happy" category shows a high f1-score of 78% and other categories follow similar improvements.



```
Classification Report:
              precision    recall  f1-score   support

       Angry       0.44      0.29      0.35       659
     Disgust       0.17      0.49      0.25        85
        Fear       0.32      0.18      0.23       653
       Happy       0.79      0.76      0.78      1402
         Sad       0.32      0.32      0.32       670
    Surprise       0.62      0.71      0.66       608
     Neutral       0.44      0.58      0.50       937

    accuracy                           0.52      5014
   macro avg       0.44      0.48      0.44      5014
weighted avg       0.52      0.52      0.51      5014
```

Fig 41: Classification Report showing evaluation metrics – Precision, Recall, F1-Score and Support for tuned FC Model with Landmarks

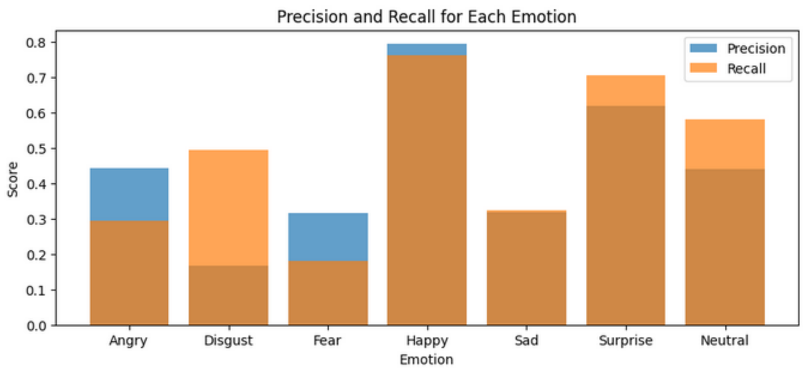


Fig 42: Representation of Precision and Recall for each Emotion for tuned FC Model with Landmarks

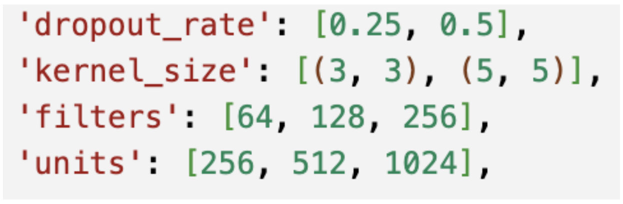
Overall, the Fully Connected tuned model provides a balanced class prediction as compared to the other 2 FC models designed in earlier steps, even though the class distribution is highly imbalanced.

4.2.2 CNN

We constructed a basic CNN network architecture model consisting of five stages. The first four stages consist of a convolutional and pooling layer, followed immediately by a flattening layer, and finally two fully connected layers. To take full advantage of the model's optimal performance in each training cycle, we first called TensorFlow's **ModelCheckpoint** callback function, which is used to save the model's weights, with the path set to ". /**Saved_model.h5**".

At the end of each training cycle, the model weights with the highest accuracy on the validation set are saved by monitoring the validation set accuracy. and calling the Early Stopping callback function to monitor the loss on the validation set and stop the training early if the loss is not reduced within the specified training cycle. Setting ***PATIENCE=3*** means to stop training if the loss is not reduced in 3 consecutive training cycles. Finally call the ReduceLROnPlateau callback function to monitor the loss on the validation set and reduce the learning rate if the loss is no longer decreasing. Helps the model converge better. Next, we used the fit_generator function for model training, setting parameters such as the number of training cycles （***epochs***） , the number of batches per cycle (***steps_per_epoch)***, and the number of batches on the validation set (***validation_steps***). During the training process, the list of callback functions is utilized, including callback functions for model saving, early stopping, and learning rate adjustment. To address the problem of category imbalance in the dataset, we introduced category weights (***class_weight=index_to_weight)*** for training to ensure that the model pays enough attention to each category.

We also Manually adjusted the learning rate, batch size, convolutional kernel size, number of filters, dropout rate, and number of neurons in the fully connected layers.



```
'dropout_rate': [0.25, 0.5],
'kernel_size': [(3, 3), (5, 5)],
'filters': [64, 128, 256],
'units': [256, 512, 1024],
```

Fig 43: Different hyper-parameter tuning for the CNN Model

Best Hyper-parameters: #Training with learning rate: 0.00001, batch size: 256, kernel size: 5*5, filters: 64, dropout rate 0.25, neurons: 256.

4.2.2.1 Accuracy & Loss

The figure below compares the accuracy and loss of the CNN model before and after parameter adjustment. It can be observed that after parameter adjustment, the accuracy of the model on the training set increased from 0.48 to 0.64, and the accuracy on the test set also increased from 0.46 to 0.56. This shows that the adjusted model achieved significant performance improvements in the training and testing stages.

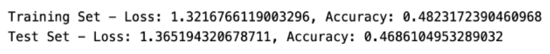
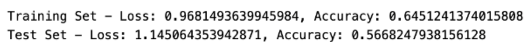
Before Tuning	After Tuning
 Training Set - Loss: 1.3216766119003296, Accuracy: 0.4823172390460968 Test Set - Loss: 1.365194320678711, Accuracy: 0.4686104953289032	 Training Set - Loss: 0.9681493639945984, Accuracy: 0.6451241374015808 Test Set - Loss: 1.145064353942871, Accuracy: 0.5668247938156128

Fig 44: Accuracy Loss for the training and test set for CNN model before and after tuning of hyper-parameters

Before Tuning

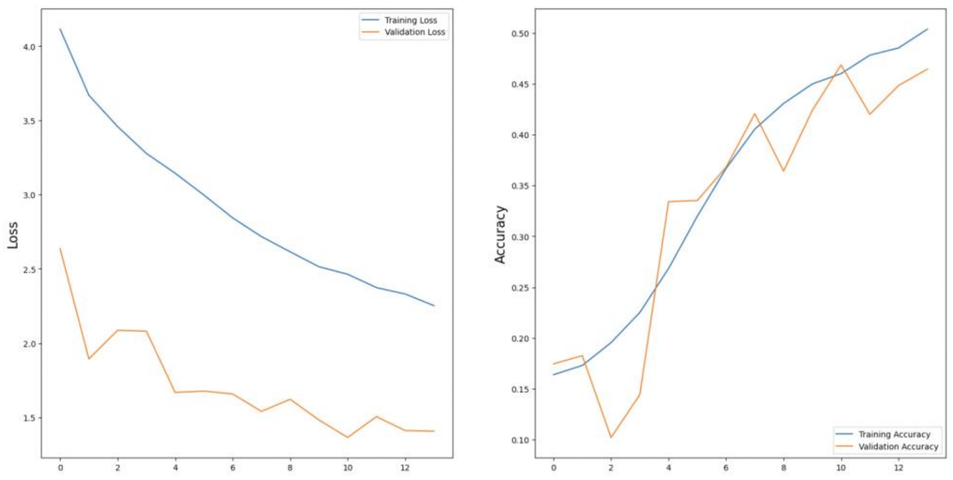


Fig 45: Results for Accuracy and Loss over epochs for CNN Model before hyper-parameter tuning

After Tuning

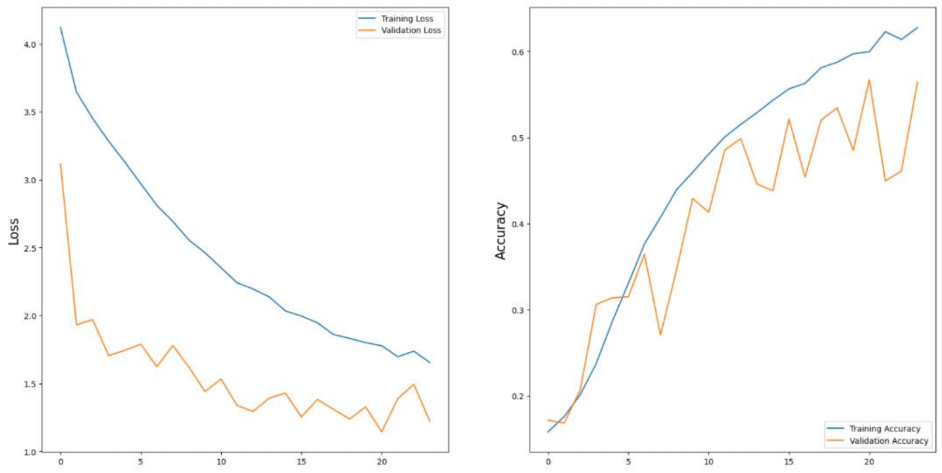


Fig 46: Results for Accuracy and Loss over epochs for CNN Model after hyper-parameter tuning with class weights

The above figure shows the accuracy and loss graphs of the CNN model after tuning. From the graph, it can be observed that the loss in the validation set is still lower than the training set after parameterization. We have troubleshot the code and found that the probable reason is due to the introduction of weight balancing or regularization for category imbalance.

This situation suggests that the tuned model generalizes better on the validation set and overfits the training data less. The accuracy and loss curves of the CNN model after tuning are shown in the following figure, where we removed the code for category weights:

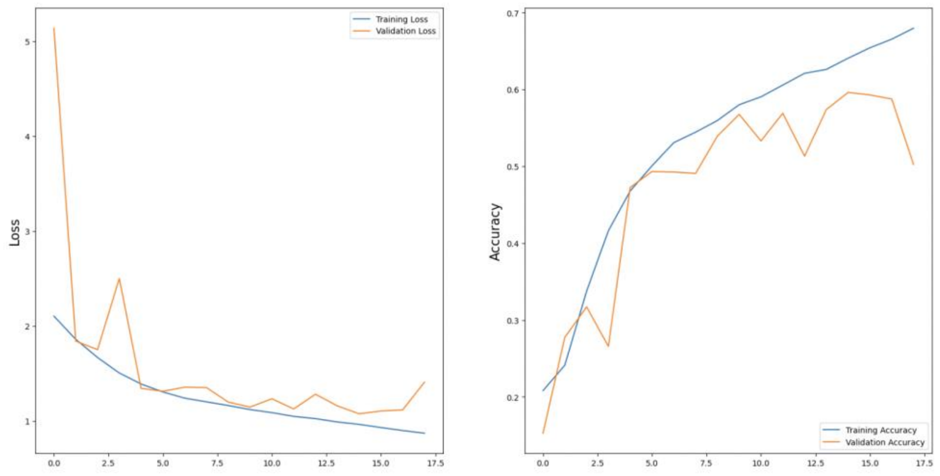
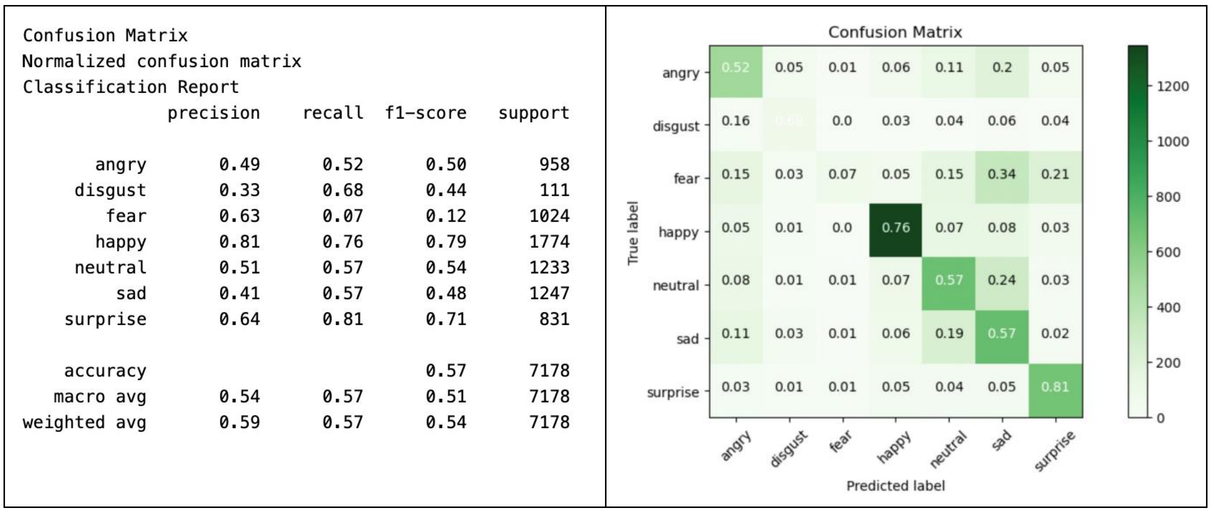


Fig 47: Results for Accuracy and Loss over epochs for CNN Model after hyper-parameter tuning without class weights

4.2.2.2 Confusion Matrix

From the confusion matrix provided in the figure below and the associated Precision, Recall, and F1-score information, we can see that the model's highest prediction accuracy for each category is in the "happy" category. The Precision is 0.81, which means that 81% of the samples predicted by the model as "happy" belong to the "happy" category. Recall measures the proportion of positive examples that the model successfully captured for each category. In the "surprise" category, the Recall is 0.81, indicating that the model successfully captured 81% of the samples that belong to the "surprise" category.

F1-score (F1 value) takes Precision and Recall into consideration and is the harmonic mean of Precision and Recall. The higher the F1-score, the better the balance between precision and recall the model has achieved. The highest f1 value is "happy", which is 0.79, followed by "surprise", which is 0.71. Support expresses the support of each category, that is, the number of samples of this category in the actual data. For example, in the "happy" category, there are 1774 samples.



```
Confusion Matrix
Normalized confusion matrix
Classification Report
              precision    recall  f1-score   support

       angry       0.49      0.52      0.50       958
     disgust       0.33      0.68      0.44       111
        fear       0.63      0.07      0.12      1024
       happy       0.81      0.76      0.79      1774
     neutral       0.51      0.57      0.54      1233
         sad       0.41      0.57      0.48      1247
    surprise       0.64      0.81      0.71       831

    accuracy                           0.57      7178
   macro avg       0.54      0.57      0.51      7178
weighted avg       0.59      0.57      0.54      7178
```

Confusion Matrix

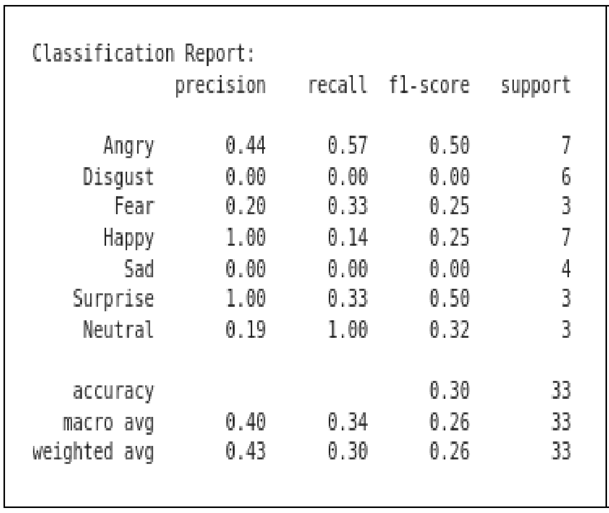
True label \ Predicted label	angry	disgust	fear	happy	neutral	sad	surprise
angry	0.52	0.05	0.01	0.06	0.11	0.2	0.05
disgust	0.16	0.68	0.0	0.03	0.04	0.06	0.04
fear	0.15	0.03	0.07	0.05	0.15	0.34	0.21
happy	0.05	0.01	0.0	0.76	0.07	0.08	0.03
neutral	0.08	0.01	0.01	0.07	0.57	0.24	0.03
sad	0.11	0.03	0.01	0.06	0.19	0.57	0.02
surprise	0.03	0.01	0.01	0.05	0.04	0.05	0.81

Fig 48: Confusion matrix and Classification Report showing evaluation metrics – Precision, Recall, F1-Score and Support for tuned CNN Model

4.3 Results of Combine Model 1 and Model 2

We utilize real images generated by Model 1 that have been processed through Haar cascading for sentiment analysis and pass these images to the fully connected model and the CNN model, respectively, to test their performance in sentiment detection and evaluate the results. By scrutinizing their performance, we can gain more insight into which model performs more accurately in sentiment classification.

For the tuned FC Model:
The tuned FC could predict the "Happy" and "Surprise" with a precision of 100% but lagged the "Disgust" and "Sad" emotion categories which had less data due to class imbalances. Overall, the model showed close to 50.50% accuracy in determining the emotion from the 40 completely unseen images.



```
Classification Report:
              precision    recall  f1-score   support

       Angry       0.44      0.57      0.50         7
     Disgust       0.00      0.00      0.00         6
        Fear       0.20      0.33      0.25         3
       Happy       1.00      0.14      0.25         7
         Sad       0.00      0.00      0.00         4
    Surprise       1.00      0.33      0.50         3
     Neutral       0.19      1.00      0.32         3

    accuracy                           0.30        33
   macro avg       0.40      0.34      0.26        33
weighted avg       0.43      0.30      0.26        33
```

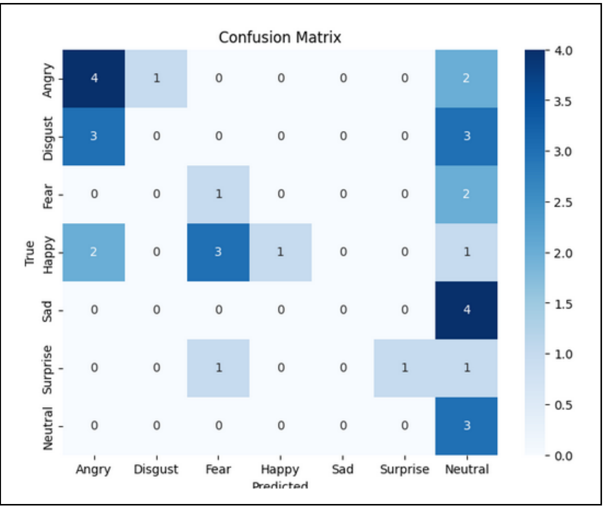
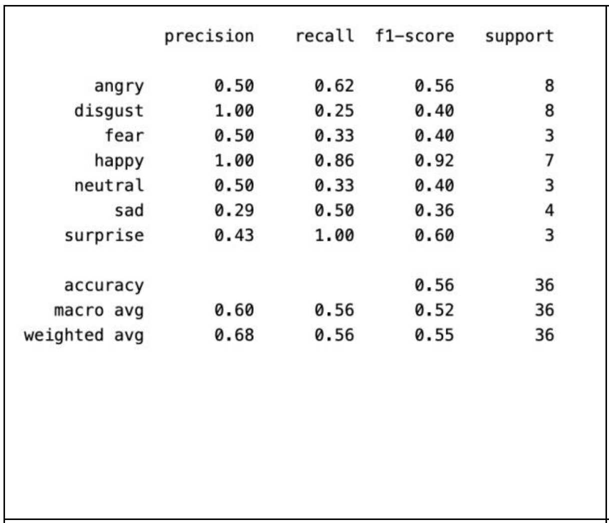


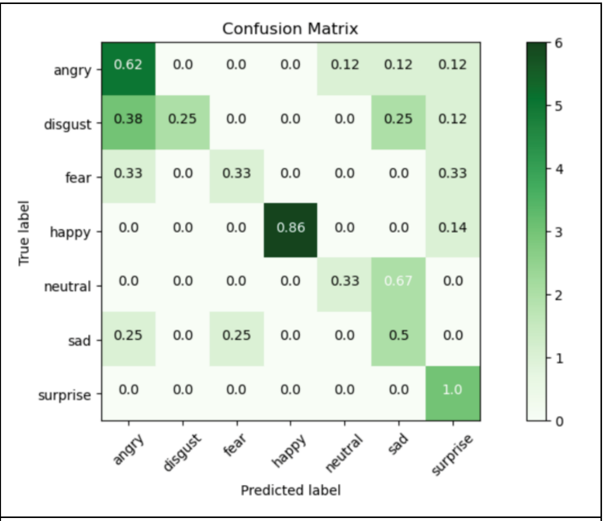
Fig 49: Confusion matrix and Classification Report showing evaluation metrics – Precision, Recall, F1-Score and Support after combining Model 1 and Model 2 (tuned FC Model)

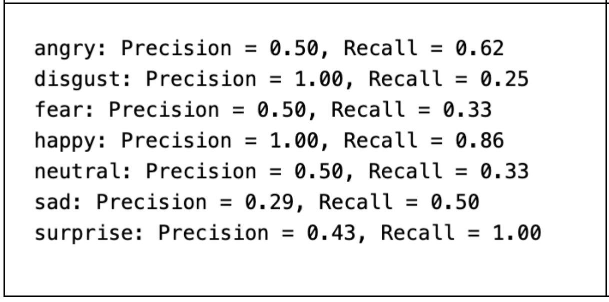
For the CNN model of Model 2, we trained and saved the best model in an H5 file. We then read the new test set and preprocessed it. Then, we obtained the true labels of the new test set and made predictions to calculate the accuracy of the model on the test set. This process further validates the performance of our model and ensures that it can accurately perform sentiment classification in real applications.

The following figure shows the confusion matrix on the new test set：



```
              precision    recall  f1-score   support

       angry       0.50      0.62      0.56         8
     disgust       1.00      0.25      0.40         8
        fear       0.50      0.33      0.40         3
       happy       1.00      0.86      0.92         7
     neutral       0.50      0.33      0.40         3
         sad       0.29      0.50      0.36         4
    surprise       0.43      1.00      0.60         3

    accuracy                           0.56        36
   macro avg       0.60      0.56      0.52        36
weighted avg       0.68      0.56      0.55        36
```





```
angry: Precision = 0.50, Recall = 0.62
disgust: Precision = 1.00, Recall = 0.25
fear: Precision = 0.50, Recall = 0.33
happy: Precision = 1.00, Recall = 0.86
neutral: Precision = 0.50, Recall = 0.33
sad: Precision = 0.29, Recall = 0.50
surprise: Precision = 0.43, Recall = 1.00
```

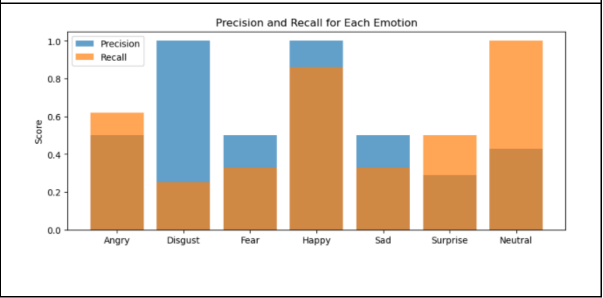


Fig 50: Confusion matrix and Classification Report showing evaluation metrics – Precision, Recall, F1-Score and Support after combining Model 1 and Model 2 (tuned FC Model)

The figure shows that the model performs well in the "Happy" and "Surprise" categories, with high Precision and Recall of 1.00 and 0.86 ("Happy") and 0.43 and 1.00 ("Surprise"), respectively. The category "Disgust" has a higher Precision (1.00) but a lower Recall (0.25), suggesting that the model may have some missed detections in this category. The category "Sad" has a low Precision and Recall of 0.29 and 0.50, respectively. There is also room for improvement in the performance of the "Angry" and "Fear" categories. The overall accuracy of the model on the new test set is 56%, showing some ability to categorize sentiment.

5. Conclusion

In conclusion, our project revolves around utilizing advanced technology, particularly through the analysis of facial expressions in images, to understand and monitor people's emotions and mental well-being. The significance of our endeavor lies in the early detection of potential mental health issues, envisioning a future where technology becomes a supportive tool in fostering better mental health.

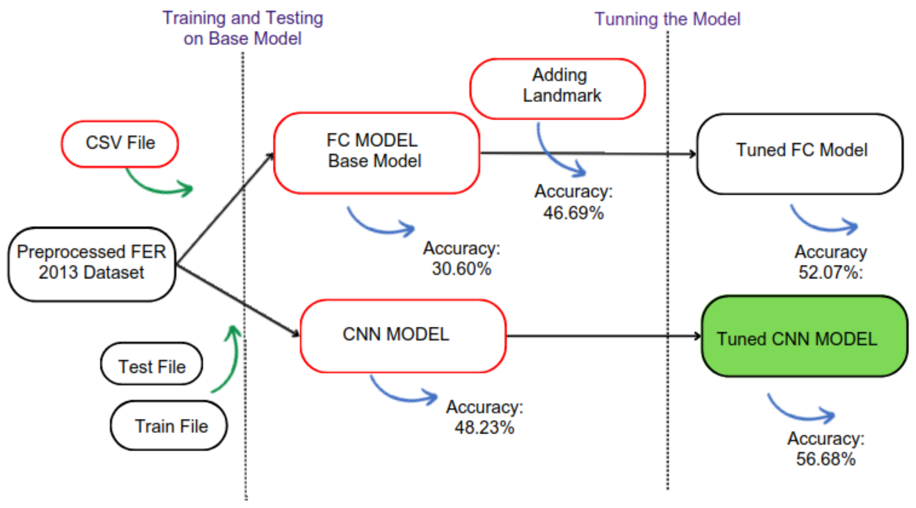


Fig 51: Summary of the results of Model 2 (FC Model and CNN Model)

Our initial hypothesis was that the Fully Connected (FC model) would perform better since we did extensive tuning on FC model, improving upon the initial Base model which had an accuracy of 30.60% , followed by adding landmarks to it which improved the accuracy to 46.69% and finally performing hyper parameter tuning it reached 52.07% but it was observed that the tuned CNN model after hyperparameter tuning had better results having an accuracy of 58.33 %.

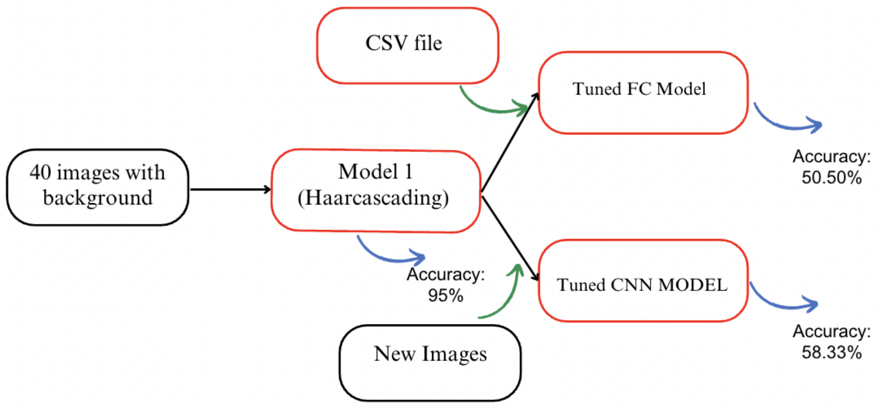


Fig 52: Summary of the results after combination of Model 1 (Haar Cascading) and Model 2 (FC Model and CNN Model)

The flowchart below illustrates the steps taken to conduct manual testing on our proposed architecture, aimed at reinforcing and validating our hypothesis for solving human sentiment analysis through face detection. We initiated the process with 40 raw input images containing various human emotions with background (noise), which were then fed into Model 1. The resulting output, as previously discussed, was subsequently directed to our two tuned models: the Tuned FC Model and the Tuned CNN Model. Notably, the tuned CNN model demonstrated superior performance, achieving an accuracy of 58.33%. This observation leads to the conclusion that further refinement of the CNN model, incorporating elements like landmarks and leveraging pre-trained models such as ResNet, has the potential to significantly enhance its accuracy. Such enhancements would contribute to a more effective solution for the emotion detection analysis challenge at hand.

6. References

[1] K. Kaulard, D. W. Cunningham, H. H. Bulthoff, C. Wallraven, "The MPI facial expression database: A validated database of emotional and conversational facial expressions," *PLoS One*, vol. 7, no. 3, art. e32321, 2012.

[2] S. Shah, "A comprehensive guide to convolutional neural networks," available online: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-theeli5-way-3bd2b1164a53, (2018)

[3] OpenCV, "Cascade Classifier," available online: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html.

[4] S. Shah, "Face detection with Haar Cascade," available online: https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08.

[5] Medium, "Haar Cascade Classifiers," available online: https://medium.datadriveninvestor.com/haar-cascade-classifiers-237c9193746b.

[6] NVIDIA, "Fully Connected Layer," available online: https://docs.nvidia.com/deeplearning/performance/dl-performance-fully-connected/index.html.

[7] V. Sharma, "Fully Connected Layer," available online: https://medium.com/@vaibhav1403/fully-connected-layer-f13275337c7c.

[8] BuiltIn, "Understanding Fully Connected Layer in Neural Networks," available online: https://builtin.com/machine-learning/fully-connected-layer.

[9] Analytics Vidhya, "Convolutional Neural Networks (CNN)," available online: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/.

[10] Towards Data Science, "Convolutional Neural Networks Explained," available online: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939.

[11] GeeksforGeeks, "Introduction to Convolution Neural Network," available online: https://www.geeksforgeeks.org/introduction-convolution-neural-network/.

[12] Aitude, "8 Important Evaluation Metrics for Classification Models," available online: https://www.aitude.com/8-important-evaluation-metrics-for-classification-models/.